

BỘ GIÁO DỤC
VÀ ĐÀO TẠO

VIỆN HÀN LÂM KHOA HỌC
VÀ CÔNG NGHỆ VIỆT NAM

HỌC VIỆN KHOA HỌC VÀ CÔNG NGHỆ



Nguyễn Thị Huệ

**NGHIÊN CỨU THUẬT TOÁN XẤP XỈ CHO
MỘT SỐ BÀI TOÁN TỐI ƯU SUBMODULAR
VỚI RÀNG BUỘC CHI PHÍ**

LUẬN ÁN TIẾN SĨ MÁY TÍNH

Hà Nội - 2026

BỘ GIÁO DỤC
VÀ ĐÀO TẠO

VIỆN HÀN LÂM KHOA HỌC
VÀ CÔNG NGHỆ VIỆT NAM

HỌC VIỆN KHOA HỌC VÀ CÔNG NGHỆ

Nguyễn Thị Huệ

**NGHIÊN CỨU THUẬT TOÁN XẤP XỈ CHO
MỘT SỐ BÀI TOÁN TỐI ƯU SUBMODULAR
VỚI RÀNG BUỘC CHI PHÍ**

LUẬN ÁN TIẾN SĨ MÁY TÍNH

Ngành: Khoa học máy tính

Mã số: 9 48 01 01

Xác nhận của Học viện
Khoa học và Công nghệ

**KT. GIÁM ĐỐC
PHÓ GIÁM ĐỐC**



Nguyễn Thị Trung

Người hướng dẫn 1
(Ký, ghi rõ họ tên)

[Signature]
Phạm Văn Cảnh

Hà Nội - 2026

Người hướng dẫn 2
(Ký, ghi rõ họ tên)

[Signature]
Nguyễn Long Cương

LỜI CAM ĐOAN

Tôi xin cam đoan rằng nội dung nghiên cứu trình bày trong luận án là kết quả nghiên cứu do tôi thực hiện trên cơ sở tìm hiểu, phân tích các tài liệu và dữ liệu có liên quan. Các kết quả, nhận xét và đánh giá trong luận án được trình bày trung thực, chính xác. Những kết quả nghiên cứu đã được công bố được liệt kê trong mục “Danh mục công trình công bố liên quan đến luận án”. Các kết quả này chưa được sử dụng để bảo vệ một học vị hoặc học hàm trong bất kỳ công trình nào khác.

Tôi hoàn toàn chịu trách nhiệm trước pháp luật về tính chính xác, trung thực của các số liệu, kết quả và nội dung được trình bày trong luận án này.

Hà Nội, ngày 14 tháng 10 năm 2026

Tác giả luận án



Nguyễn Thị Huệ

LỜI CẢM ƠN

Trước hết, tôi xin bày tỏ lòng biết ơn chân thành và sâu sắc nhất tới tập thể hướng dẫn khoa học của tôi, PGS. TS. Phạm Văn Cảnh và PGS. TS. Nguyễn Long Giang. Trong suốt quá trình thực hiện học tập và nghiên cứu, các thầy đã luôn tận tình chỉ bảo, định hướng nghiên cứu và dành nhiều thời gian trao đổi học thuật quý báu, giúp tôi từng bước hoàn thiện công trình này.

Tôi xin chân thành cảm ơn các Thầy, Cô tại Viện Công nghệ thông tin, Ban Lãnh đạo, Phòng Đào tạo cùng các phòng chức năng của Học viện Khoa học và công nghệ, Viện Hàn lâm Khoa học và Công nghệ Việt Nam, những người đã giảng dạy, tạo điều kiện thuận lợi và cung cấp môi trường nghiên cứu học thuật nghiêm túc trong suốt quá trình tôi học tập và thực hiện luận án.

Tôi xin trân trọng gửi lời cảm ơn tới Ban Lãnh đạo Trường Đại học Kiến trúc Hà Nội đã luôn quan tâm, tạo điều kiện thuận lợi trong suốt quá trình học tập, nghiên cứu và thực hiện luận án. Tôi cũng xin chân thành cảm ơn Ban Chủ nhiệm Khoa Công nghệ thông tin cùng các đồng nghiệp tại Trường Đại học Kiến trúc Hà Nội đã luôn động viên, hỗ trợ và chia sẻ nhiều kinh nghiệm quý báu trong quá trình công tác và nghiên cứu.

Tôi xin gửi lời cảm ơn tới các nhà khoa học tại Phòng thí nghiệm Tối ưu hóa các hệ thống lớn (ORLab), Đại học Phenikaa, đã dành nhiều thời gian trao đổi học thuật, đóng góp những ý kiến quý báu và hỗ trợ tôi hoàn thiện các ý tưởng nghiên cứu, góp phần quan trọng vào việc hoàn thành luận án này.

Tôi đặc biệt cảm ơn gia đình, chồng, con và bạn bè của tôi đã luôn động viên và giúp đỡ tôi trong suốt quá trình học tập, nghiên cứu, tham gia các hội nghị trong và ngoài nước cũng như thực hiện luận án này.

Hà Nội, ngày 14 tháng 10 năm 2026

Tác giả luận án



Nguyễn Thị Huệ

MỤC LỤC

Lời cam đoan	i
Lời cảm ơn	ii
Danh mục các từ viết tắt	v
Danh mục bảng	vii
Danh mục các hình vẽ	viii
MỞ ĐẦU	1
Chương 1. Tổng quan về các bài toán tối ưu submodular	9
1.1. Bài toán tối ưu tổ hợp	9
1.2. Bài toán tối ưu submodular	11
1.2.1. Định nghĩa bài toán	12
1.2.2. Một số bài toán cụ thể	16
1.2.3. Hàm k -submodular: mở rộng của hàm submodular	20
1.3. Thuật toán xấp xỉ	21
1.3.1. Định nghĩa	22
1.3.2. Ưu điểm và nhược điểm của thuật toán xấp xỉ	23
1.3.3. Thuật toán xấp xỉ tiêu chuẩn kép	24
1.4. Một số phương pháp thiết kế TXXX hiện đại	25
1.4.1. Thuật toán tham lam	25
1.4.2. Thuật toán luồng	27
1.4.3. Thuật toán song song	28
1.5. Tổng quan về bài toán tối ưu submodular	32
1.5.1. Bài toán tối đa hàm submodular	32
1.5.2. Bài toán phủ submodular với chi phí tối thiểu	33
1.5.3. Các thách thức trong nghiên cứu bài toán SO	34
1.6. Kết luận chương	35
Chương 2. Bài toán Phủ Submodular với chi phí tối thiểu	36
2.1. Phát biểu bài toán	36
2.2. Các nghiên cứu liên quan	37
2.3. Thuật toán song song AdaptMCSC	41
2.3.1. Thuật toán ngưỡng tuần tự THRESHSEQ	42
2.3.2. Thuật toán AdaptMCSC	48
2.4. Thực nghiệm và kết quả	61
2.4.1. Thiết lập thực nghiệm	62

2.4.2. Kết quả và thảo luận	64
2.5. Kết luận Chương	68
Chương 3. Bài toán Phủ k-Submodular với chi phí tối thiểu	71
3.1. Đặt vấn đề và phát biểu bài toán	72
3.1.1. Hàm k -submodular	72
3.1.2. Bài toán Phủ k -submodular	75
3.2. Các nghiên cứu liên quan và đóng góp của Luận án	78
3.3. Thuật toán cho trường hợp chi phí đồng nhất	81
3.3.1. Thuật toán FBiUOpt (biết ước lượng tối ưu)	81
3.3.2. Thuật toán FBiU	85
3.3.3. Thực nghiệm	87
3.4. Thuật toán cho bài toán Phủ k -submodular tổng quát	91
3.4.1. Thuật toán tham lam mở rộng	91
3.4.2. Thuật toán xấp xỉ nhanh FBi cho hàm đơn điệu	98
3.4.3. Thực nghiệm	109
3.5. Kết luận chương	117
Chương 4. Bài toán Tối đa hàm submodular với ràng buộc chi phí công bằng	118
4.1. Phát biểu bài toán	118
4.2. Các nghiên cứu liên quan	122
4.3. Thuật toán đề xuất	124
4.3.1. Thuật toán cho trường hợp đặc biệt khi chi phí đồng nhất	124
4.3.2. Thuật toán cho trường hợp tổng quát	126
4.4. Thực nghiệm và kết quả	130
4.4.1. Thiết lập thực nghiệm	131
4.4.2. Kết quả thực nghiệm	132
4.5. Kết luận chương	136
KẾT LUẬN VÀ KIẾN NGHỊ	138
DANH MỤC CÔNG TRÌNH CÔNG BỐ LIÊN QUAN ĐẾN LUẬN ÁN	140
DANH MỤC TÀI LIỆU THAM KHẢO	141

DANH MỤC CÁC TỪ VIẾT TẮT

Từ viết tắt	Tiếng Anh	Tiếng Việt
BA	Barabási–Albert Model	Mô hình mạng Barabási–Albert
CO	Combinatorial Optimization	Tối ưu tổ hợp
CREW	Concurrent Read Exclusive Write	Đọc đồng thời, ghi độc quyền
CRCW	Concurrent Read Concurrent Write	Đọc đồng thời, ghi đồng thời
ER	Erdos–Rényi Graph	Đồ thị ngẫu nhiên Erdos–Rényi
EREW	Exclusive Read Exclusive Write	Đọc độc quyền, ghi độc quyền
GP	Gaussian Process	Quá trình Gauss
HPC	High Performance Computing	Tính toán hiệu năng cao
IC	Independent Cascade	Mô hình lan truyền độc lập
IM	Influence Maximization	Tối đa ảnh hưởng
ITk	Influence Threshold with k top-ics	Ảnh hưởng ngưỡng với k chủ đề
kSC	Minimum Cost k -Submodular Cover	Phủ k -submodular với chi phí tối thiểu
LT	Linear Threshold	Mô hình ngưỡng tuyến tính
Max-Cut	Maximum Cut	Lát cắt cực đại
MC	Maximum Coverage	Phủ cực đại
MCSC	Minimum Cost Submodular Cover	Phủ submodular với chi phí tối thiểu
NP	Nondeterministic Polynomial Time	Lớp bài toán không đơn định thời gian đa thức
OSN	Online Social Network	Mạng xã hội trực tuyến
P	Polynomial Time	Lớp bài toán giải trong thời gian đa thức
PRAM	Parallel Random Access Machine	Mô hình máy truy cập ngẫu nhiên song song
RM	Revenue Maximization	Tối đa doanh thu
RT	Revenue Threshold	Ngưỡng doanh thu
SAT	Boolean Satisfiability	Thỏa mãn công thức logic
SC	Set Cover	Tập phủ

Từ viết tắt	Tiếng Anh	Tiếng Việt
SMK	Submodular Maximization with Knapsack Constraint	Tối đa hóa submodular với ràng buộc knapsack
SM	Submodular Maximization	Tối đa submodular
SMFC	Submodular Maximization with Fair Cost Constraints	Tối đa hàm submodular với ràng buộc chi phí công bằng
SNAP	Stanford Large Network Dataset Collection	Bộ dữ liệu mạng lớn của Stanford
TI	Threshold Influence	Ảnh hưởng ngưỡng
TTXX	Approximation Algorithm	Thuật toán xấp xỉ
WSC	Weighted Set Cover	Tập phủ có trọng số

DANH MỤC BẢNG

1.1	So sánh độ phức tạp thời gian và độ phức tạp truy vấn.	12
1.2	Minh họa các vòng truy vấn song song với $n = 1000$ và $k = 5$	31
2.1	Bảng so sánh các thuật toán hiện có cho bài toán MCSC và SC. Ký hiệu (a, b) tương ứng với TTX tiêu chuẩn kép (a, b)	40
2.2	Tập người dùng bị ảnh hưởng bởi mỗi người dùng.	56
2.3	Bảng thống kê các dữ liệu được sử dụng	63
2.4	Thiết lập các tham số thực nghiệm.	63
2.5	So sánh tỷ lệ cải thiện của AdaptMCSC so với các thuật toán cơ sở, trong đó $\% \text{ cải thiện} = \frac{\text{Cost}(\text{Baseline}) - \text{Cost}(\text{Adapt})}{\text{Cost}(\text{Baseline})} \times 100$	66
3.1	Các thuật toán cho bài toán kSC. Cặp (x, y) biểu diễn tỷ lệ xấp xỉ tiêu chuẩn kép (x, y) của một thuật toán, tức là thuật toán trả về nghiệm s sao cho $ supp(s) \leq x \cdot \text{opt}$ và $g(s) \geq y \cdot T$, với $x, y > 0$	80
3.2	Các thuật toán cho bài toán kSC. Bộ (x, y) biểu diễn tỷ lệ xấp xỉ tiêu chuẩn kép (x, y)	80
3.3	Các tham số sử dụng trong thực nghiệm	110
3.4	Các bộ dữ liệu sử dụng trong thực nghiệm	110
3.5	Thống kê số so sánh theo trường hợp của thuật toán đề xuất với các thuật toán theo tổng số 15 trường hợp của T	111
4.1	Tóm tắt các kết quả chính của luận án cho bài toán SMFC. $n = V $ là kích thước tập nền, k là ngân sách trong trường hợp chi phí đồng nhất, $\alpha \in (0, 1)$ là tỷ lệ công bằng giữa các nhóm, và $\epsilon > 0$ là tham số đánh đổi giữa chất lượng nghiệm và thời gian chạy của thuật toán.	124
4.2	Các tập dữ liệu	132

DANH MỤC CÁC HÌNH VẼ

0.1	Các dạng của bài toán SO và đóng góp của LA.	6
2.1	So sánh tổng chi phí của các thuật toán tham lam, MinSMC và AdaptMCSC theo các giá trị ngưỡng T	65
2.2	Số vòng lặp tuần tự của các thuật toán GREEDY, MinSMC và AdaptMCSC trên ba bộ dữ liệu Email EU, Facebook và NetHEPT với các giá trị ngưỡng $T \in \{0.1g(V), 0.2g(V), \dots, 0.5g(V)\}$. Trục tung được biểu diễn theo thang logarit.	67
2.3	Thời gian chạy của các thuật toán GREEDY, MinSMC và AdaptMCSC. . .	69
3.1	So sánh hiệu năng (chi phí, số lượng truy vấn và kích thước nghiệm) giữa FBIU và các phương pháp GREEDY và luồng trên bộ dữ liệu ER. . .	89
3.2	So sánh hiệu năng (giá trị hàm mục tiêu, số lượng truy vấn và kích thước nghiệm) giữa FBIU và các phương pháp GREEDY và luồng trên bộ dữ liệu Email.	90
3.3	So sánh hiệu năng của các thuật toán trên mạng Barabási–Albert (BA) theo ngưỡng T với ứng dụng IT.	112
3.4	So sánh hiệu năng của các thuật toán theo ngưỡng T với ứng dụng IT: (a) chi phí của lời giải; (b) số lượng truy vấn; (c) thời gian chạy.	113
3.5	So sánh hiệu năng của các thuật toán trên bộ dữ liệu Email-Eu đối với ứng dụng RT theo ngưỡng T : (a) chi phí của lời giải; (b) số lượng truy vấn; (c) thời gian chạy.	114
3.6	So sánh hiệu năng của các thuật toán trên bộ dữ liệu Email-Eu đối với ứng dụng RT theo các giá trị ngưỡng T : (a) chi phí của lời giải; (b) số lượng truy vấn; (c) thời gian chạy.	115
4.1	Giá trị hàm f (ảnh hưởng) của các thuật toán với trường hợp chi phí đồng nhất trên dữ liệu (a) Facebook, (b) Google Plus.	132
4.2	Thời gian chạy của các thuật toán với trường hợp chi phí đồng nhất trên dữ liệu (a) Facebook, (b) Google Plus.	133
4.3	Giá trị hàm f (ảnh hưởng) của các thuật toán với trường hợp chi phí tổng quát trên các bộ dữ liệu: (a) Musae-Github, (b) Gemsec-Deezer, (c) Gemsec-Facebook và (d) Google Plus.	134

4.4	Thời gian chạy của các thuật toán với trường hợp chi phí tổng quát trên các bộ dữ liệu: (a) Musae-Github, (b) Gemsec-Deezer, (c) Gemsec-Facebook và (d) Google Plus.	135
-----	--	-----

MỞ ĐẦU

Các bài toán tối ưu tổ hợp (Combinatorial Optimization - CO) hình thành từ nhu cầu ra quyết định trong nhiều hoạt động kinh tế, kỹ thuật và đời sống xã hội. Trong thực tế, người ra quyết định thường phải lựa chọn một phương án tối ưu từ tập các phương án hợp lệ, đồng thời phải thỏa mãn nhiều điều kiện và ràng buộc khác nhau. Những tình huống như xác định tuyến đường vận chuyển hiệu quả, xây dựng lịch làm việc, phân công nhiệm vụ hay phân bổ nguồn lực đều có thể được mô hình hóa thành các bài toán CO. Nhờ khả năng biểu diễn đa dạng các bài toán thực tiễn dưới một khuôn khổ toán học thống nhất, CO đã trở thành một lĩnh vực nghiên cứu quan trọng trong khoa học máy tính, toán học ứng dụng và nghiên cứu tác nghiệp. Các mô hình và phương pháp của lĩnh vực này không chỉ giúp phân tích và giải quyết hiệu quả các bài toán có cấu trúc rời rạc mà còn đóng vai trò nền tảng trong việc phát triển các thuật toán xử lý dữ liệu quy mô lớn và các hệ thống hỗ trợ ra quyết định hiện đại.

Bước sang kỷ nguyên trí tuệ nhân tạo, các bài toán CO ngày càng trở nên cấp thiết khi đóng vai trò trung tâm trong nhiều hệ thống hiện đại của khoa học dữ liệu, như tối đa ảnh hưởng trên các mạng xã hội [1, 2], tóm tắt thông tin [3, 4, 5], phân tích đặc trưng người dùng, hệ thống khuyến nghị, hay bài toán tối đa hóa doanh thu trong các chiến dịch tiếp thị lan truyền [6, 3, 7]. Việc tăng nhanh quy mô dữ liệu dẫn tới các yêu cầu xử lý hiệu quả trong thời gian ngắn đặt ra nhu cầu cấp bách về các thuật toán tối ưu có khả năng xử lý các bài toán quy mô lớn. Một đặc điểm quan trọng của lớp bài toán này là chúng có thể được biểu diễn dưới dạng các bài toán CO trên tập hợp, trong đó hàm mục tiêu hoặc ràng buộc thường mang tính *submodular*, còn được gọi là tính chất *lợi nhuận biên giảm dần*. Tính chất này cho phép thiết kế các thuật toán hiệu quả với các đảm bảo lý thuyết về chất lượng lời giải, đồng thời phù hợp với các ứng dụng xử lý dữ liệu quy mô lớn. Những bài toán này được các nhà nghiên cứu gọi chung là tối ưu submodular (Submodular Optimization-SO).

Dưới dạng tổng quát nhất, bài toán SO được phát biểu như sau: Cho tập nền hữu hạn $V = \{e_1, e_2, \dots, e_n\}$, với hàm mục tiêu $f : 2^V \mapsto \mathbb{R}_+^1$ có tính chất submodular, nghĩa là

$$f(A \cup \{u\}) - f(A) \geq f(B \cup \{u\}) - f(B),$$

với mọi $A \subseteq B \subseteq V$ và $u \in V \setminus B$. Bài toán SO yêu cầu tìm tập con $S \subseteq V$ thỏa mãn ràng buộc $\mathcal{C}$ cho trước sao cho giá trị của f là cực đại hoặc cực tiểu.

Bài toán SO thuộc lớp bài toán NP-khó, tức là không tồn tại thuật toán tìm lời giải

¹ 2^V : Tập tất cả các tập con của V bao gồm tập rỗng.

chính xác chạy trong thời gian đa thức, trừ khi $P = NP$. Do đó, trong thực tế, các hướng tiếp cận chủ yếu tập trung vào việc thiết kế các thuật toán gần đúng nhằm đạt được lời giải tốt trong thời gian tính toán chấp nhận được. Trong hơn một thập niên trở lại đây, việc nghiên cứu các bài toán SO liên tục được mở rộng đầu tư khi các nhà khoa học tìm thấy ứng dụng của các bài toán trên trong nhiều lĩnh vực trong CO, trí tuệ nhân tạo, học máy. Nhiều phương pháp hiện đại liên tục được công bố và trình bày trên các tạp chí uy tín và hàng đầu về tối ưu hóa như *Mathematics of Operations Research*, *Mathematical Programming*, *Computer & Operations Research*, *SIAM Journal on Computing*, *European Journal of Operational Research*, v.v. cũng như tại các hội nghị hàng đầu (A^*) về học máy và trí tuệ nhân tạo như NeurIPS, ICML, AAAI, IJCAI, ICLR, UAI, v.v. cho thấy vai trò ngày càng quan trọng của lớp bài toán này trong cả nghiên cứu lý thuyết lẫn các ứng dụng liên quan đến khoa học dữ liệu với quy mô lớn.

Lịch sử nghiên cứu về các bài toán SO từ rất sớm, xuất phát từ các yêu cầu thiết yếu trong vấn đề kinh tế nơi bài toán lựa chọn tập phương án tối ưu dưới các giới hạn nguồn lực được nghiên cứu rộng rãi [8, 9, 10, 6, 11, 12, 13]. Nemhauser và cộng sự [14] là những nhà nghiên cứu tiên phong tìm thấy ứng dụng to lớn của bài toán này trong kinh tế học và phát triển các thuật toán đạt tỷ lệ xấp xỉ $1 - 1/e$ cho các bài toán với ràng buộc kích thước, đặt nền móng cho việc khai thác tính chất submodular trong thiết kế thuật toán xấp xỉ (TTXX). Những năm gần đây, SO đã được mở rộng nghiên cứu dưới nhiều dạng ràng buộc khác nhau nhằm phản ánh đa dạng các yêu cầu thực tế. Các dạng ràng buộc phổ biến bao gồm: ràng buộc kích thước lời giải (cardinality constraints) [15, 16, 8], ràng buộc tổng chi phí của lời giải (knapsack constraints) [13, 17, 18], tìm lời giải với chi phí tối thiểu với ràng buộc cho trước [16, 8, 19], cũng như các ràng buộc cấu trúc tổng quát hơn như matroid hoặc giao của nhiều matroid [3]. Những mở rộng này cho phép mô hình hóa các bài toán thực tế phức tạp hơn, nơi lời giải không chỉ cần đạt giá trị mục tiêu cao mà còn phải tuân thủ các cấu trúc hay giới hạn tài nguyên đa dạng. Vì lớp bài toán SO rất rộng các nhà nghiên cứu chia bài toán thành dạng cụ thể như sau:

- **Dạng 1 (Cực tiểu) - Bài toán phủ submodular với chi phí tối thiểu (MCSC)**: Trong bài toán này, mỗi phần tử $e \in V$ có một chi phí $c(e) > 0$. Với ngưỡng giá trị T cho trước và một hàm đánh giá $g : 2^V \mapsto \mathbb{R}_+$ là submodular, bài toán MCSC yêu cầu tìm tập con $S \subseteq V$ có tổng chi phí $\sum_{e \in S} c(e)$ là nhỏ nhất sao cho $g(S) \geq T$. Dạng bài toán này phổ biến trong các ứng dụng có thể kể đến như: tối đa ảnh hưởng ngưỡng [20], tóm tắt dữ liệu ảnh [21, 3, 10, 13, 5], tìm lát cắt với chi phí tối thiểu [22], v.v.

- **Dạng 2 (Cực đại) - Bài toán tối đa hàm submodular (SM):** Với hàm mục tiêu nhận giá trị thực không âm $f : 2^V \mapsto \mathbb{R}_+$ là submodular, bài toán SM yêu cầu tìm tập con $S \subseteq V$ thỏa mãn ràng buộc $\mathcal{C}$ cho trước sao cho $f(S)$ đạt giá trị lớn nhất.

Dạng bài toán này bao trùm nhiều bài toán điển hình trong khoa học dữ liệu và học máy, chẳng hạn như tối đa hóa ảnh hưởng trong mạng xã hội [1], lựa chọn đặc trưng trong học máy [5, 10], tối đa hóa doanh thu trong mạng xã hội [23], bài toán đặt cảm biến [6, 12], và lựa chọn tập đại diện nhằm tối đa hóa lượng thông tin thu được dưới các ràng buộc tài nguyên [10, 24], v.v. Các ràng buộc trong $\mathcal{C}$ thường gặp trong các bài toán CO bao gồm: ràng buộc kích thước (cardinality constraint), ràng buộc ba lô (knapsack constraint), ràng buộc matroid (matroid constraint), v.v.

Trong LA, ký hiệu SM được dùng cho bài toán tối đa hàm submodular tổng quát với ràng buộc $\mathcal{C}$ bất kỳ. Các trường hợp riêng ứng với từng dạng ràng buộc cụ thể được ký hiệu riêng: SMC là bài toán SM với ràng buộc kích thước, SMK là bài toán SM với ràng buộc chi phí (knapsack), và SMFC là bài toán SM với ràng buộc chi phí công bằng được LA đề xuất nghiên cứu tại Chương 4.

Tận dụng các tính chất quan trọng của hàm submodular đối với bài toán SO, nhiều nghiên cứu đã tập trung phát triển các TTXH hiệu quả theo hai hướng chính.

Thứ nhất, một số công trình tập trung nâng cao chất lượng nghiệm thu được thông qua việc cải thiện tỷ lệ xấp xỉ [25, 26, 27, 28]. Đối với các bài toán CO NP-khó như SO, việc tìm nghiệm tối ưu trong thời gian đa thức thường là không khả thi. Vì vậy, các TTXH được thiết kế nhằm bảo đảm nghiệm thu được đạt ít nhất một tỷ lệ nhất định so với nghiệm tối ưu. Tỷ lệ xấp xỉ càng lớn thì nghiệm của thuật toán càng gần với nghiệm tối ưu. Nhờ khai thác các tính chất đặc trưng của hàm submodular như tính lợi ích giảm dần (*diminishing returns*) và khả năng xây dựng nghiệm hiệu quả bằng chiến lược tham lam, nhiều nghiên cứu đã đề xuất các thuật toán với cận xấp xỉ ngày càng chặt hơn, đồng thời vẫn duy trì được độ phức tạp tính toán hợp lý.

Thứ hai, nhiều nghiên cứu hướng tới cải thiện thời gian tính toán và bộ nhớ sử dụng nhằm mở rộng khả năng áp dụng cho dữ liệu quy mô lớn. Trong thực tế, các bài toán tối ưu submodular thường phát sinh trên các hệ thống dữ liệu lớn như mạng xã hội, hệ thống gợi ý hoặc các ứng dụng trong khoa học dữ liệu hiện đại, nơi kích thước dữ liệu có thể lên tới hàng triệu hoặc hàng tỷ phần tử. Điều này khiến các thuật toán truyền thống khó đáp ứng được yêu cầu về tốc độ xử lý và tài nguyên tính toán. Vì vậy, nhiều hướng tiếp cận đã được đề xuất nhằm giảm chi phí tính toán nhưng vẫn duy trì được chất lượng nghiệm tốt. Các hướng tiếp cận tiêu biểu, cho hướng này bao gồm:

(1) *Phát triển các thuật toán có độ phức tạp truy vấn thấp, tức giảm số lần gọi hàm*

mục tiêu submodular [28, 13, 24, 10]. Trong nhiều bài toán thực tế, việc đánh giá giá trị của hàm mục tiêu thường là bước tốn kém nhất do phải xử lý lượng dữ liệu rất lớn hoặc thực hiện các phép mô phỏng phức tạp. Vì vậy, giảm số lần truy vấn hàm mục tiêu có ý nghĩa đặc biệt quan trọng trong việc tăng tốc thuật toán. Các nghiên cứu gần đây tập trung vào các kỹ thuật như chiến lược ngưỡng, lấy mẫu hoặc đánh giá xấp xỉ nhằm giảm đáng kể số lượng truy vấn nhưng vẫn bảo đảm được tỷ lệ xấp xỉ lý thuyết;

(2) *Khai thác mô hình tính toán song song và phân tán để liên tục rút ngắn thời gian tìm lời giải* [29, 30, 31, 32, 12, 19]. Mục tiêu chính của hướng tiếp cận này là phân chia khối lượng tính toán thành nhiều tác vụ có thể xử lý đồng thời trên nhiều bộ xử lý hoặc nhiều máy tính khác nhau, từ đó rút ngắn đáng kể thời gian thực thi. Các thuật toán trực tuyến cho phép xử lý dữ liệu ngay khi dữ liệu xuất hiện mà không cần biết trước toàn bộ dữ liệu đầu vào. Tuy nhiên, việc thiết kế các thuật toán song song hiệu quả cho tối ưu submodular thường phải cân bằng giữa tốc độ xử lý, số vòng thích nghi (*adaptivity*) và chất lượng nghiệm, do đó đây vẫn là một hướng nghiên cứu đầy thách thức.

(3) *Xử lý dữ liệu dạng dòng (streaming) với yêu cầu giới hạn về bộ nhớ và số lần duyệt toàn bộ dữ liệu.* [23, 33] Trong mô hình này, dữ liệu xuất hiện liên tục theo thời gian và thuật toán chỉ được phép truy cập tuần tự từng phần tử với số lần quét dữ liệu rất hạn chế. Điều này đặc biệt phù hợp với các ứng dụng thời gian thực hoặc các hệ thống có dữ liệu vượt quá khả năng lưu trữ của bộ nhớ chính. Vì vậy, các thuật toán streaming thường được thiết kế với yêu cầu sử dụng bộ nhớ nhỏ, tốc độ xử lý nhanh nhưng vẫn duy trì được các bảo đảm xấp xỉ tốt cho nghiệm thu được.

Mặc dù nhiều kết quả quan trọng đã đạt được hiện tại, lớp các bài toán SO vẫn còn nhiều thách thức cần phải giải quyết cả về phương diện mô hình hóa lẫn thiết kế thuật toán.

Thứ nhất, sự gia tăng nhanh chóng về quy mô dữ liệu đặt ra yêu cầu cân bằng giữa chất lượng lời giải và hiệu quả tính toán của các thuật toán tối ưu submodular [7]. Trong nhiều ứng dụng, việc đánh giá hàm mục tiêu là công đoạn tốn kém nhất. Chẳng hạn, đối với bài toán tối đa ảnh hưởng, giá trị ảnh hưởng kỳ vọng thường phải được ước lượng thông qua nhiều lần mô phỏng quá trình lan truyền trên mạng [1]. Vì vậy, nhiều nghiên cứu tập trung vào việc giảm độ phức tạp truy vấn và số vòng tính toán phụ thuộc tuần tự bằng các kỹ thuật lấy mẫu và xử lý song song [34, 7]. Việc xây dựng các thuật toán có độ phức tạp truy vấn và độ phức tạp song song thấp, đồng thời vẫn duy trì các bảo đảm lý thuyết về chất lượng nghiệm, do đó trở thành một hướng nghiên cứu quan trọng đối với các bài toán tối ưu submodular trên dữ liệu quy mô lớn.

Thứ hai, nhiều biến thể có ý nghĩa thực tiễn của các bài toán tối ưu submodular

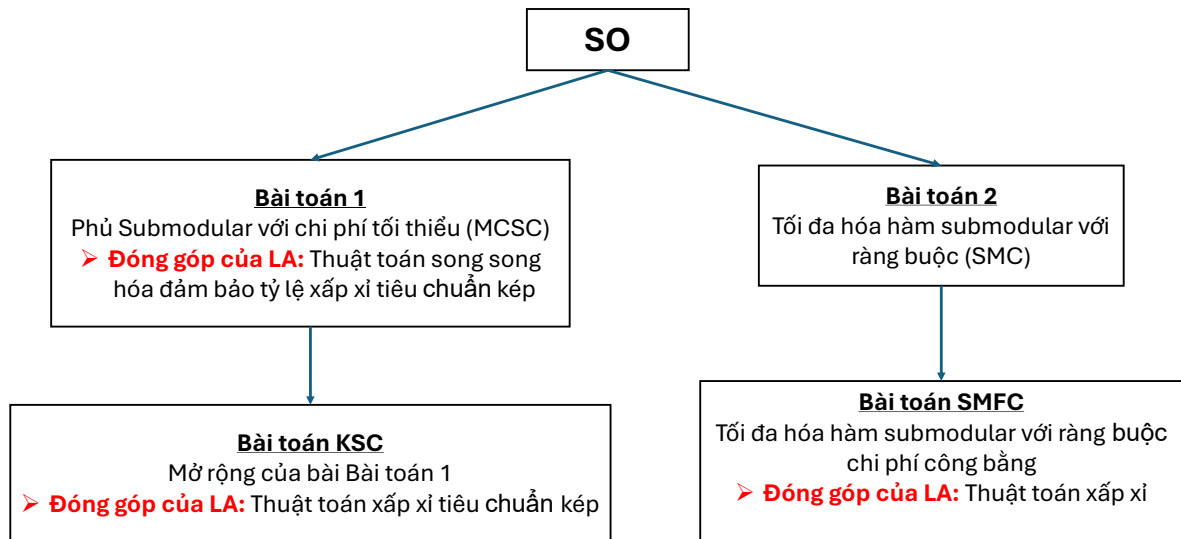
vấn cần tiếp tục được nghiên cứu và phát triển. Các nghiên cứu gần đây đã xem xét yêu cầu công bằng giữa các nhóm thông qua những ràng buộc về mức độ đại diện, số lượng phần tử được lựa chọn hoặc chi phí phân bổ cho từng nhóm [35, 36]. Bên cạnh đó, trong nhiều ứng dụng, bài toán còn phải đồng thời thỏa mãn các giới hạn về tài nguyên và các ràng buộc cấu trúc phức tạp như knapsack hoặc matroid [37, 36]. Những yêu cầu này khó được mô tả đầy đủ bởi các mô hình truyền thống chỉ xét một ràng buộc lực lượng hoặc ngân sách đơn lẻ. Vì vậy, việc xây dựng các mô hình và thuật toán có khả năng xử lý đồng thời các ràng buộc phức tạp, đồng thời duy trì bảo đảm lý thuyết về chất lượng nghiệm, vẫn là một hướng nghiên cứu nhận được nhiều sự quan tâm [37, 35, 36].

Xuất phát từ những thách thức đã nêu, luận án (LA) hướng đến việc phát triển các thuật toán xấp xỉ nhằm nâng cao hiệu quả giải quyết các bài toán tối ưu trong khoa học dữ liệu quy mô lớn, có xét đến yếu tố chi phí và yêu cầu công bằng. Với định hướng đó, LA được thực hiện với đề tài **Nghiên cứu thuật toán xấp xỉ cho một số bài toán tối ưu submodular với ràng buộc chi phí**, với các mục tiêu sau:

1. Nghiên cứu các bài toán tối ưu submodular (SO) và mở rộng k -submodular gắn với yêu cầu tối ưu chi phí và ràng buộc công bằng. Cụ thể, LA tập trung vào bài toán phủ submodular với chi phí tối thiểu (MCSC), bài toán phủ k -submodular (kSC) trên không gian lời giải đa tập và bài toán tối đa hóa hàm submodular với ràng buộc chi phí công bằng (SMFC). Các bài toán này mô hình hóa yêu cầu đạt ngưỡng giá trị mục tiêu với chi phí nhỏ nhất hoặc tối đa hóa giá trị mục tiêu dưới các giới hạn ngân sách tổng và ngân sách riêng cho từng nhóm.
2. Đề xuất và phân tích các thuật toán xấp xỉ (TTXX) cho các bài toán trên nhằm bảo đảm chất lượng lời giải và hiệu quả tính toán trong bối cảnh việc tìm nghiệm tối ưu chính xác thường không khả thi. Các thuật toán được thiết kế với những bảo đảm lý thuyết về tỷ lệ xấp xỉ và độ phức tạp tính toán, đồng thời chú trọng giảm số lượng truy vấn hàm mục tiêu và khai thác khả năng song song đối với các bài toán phù hợp. Hiệu quả của các thuật toán được đánh giá thông qua phân tích lý thuyết và thực nghiệm, hướng đến khả năng ứng dụng trên dữ liệu quy mô lớn.

Với mục tiêu trên, LA tập trung nghiên cứu ba bài toán thuộc hai dạng cơ bản của SO. Thuộc Dạng 1, LA nghiên cứu bài toán MCSC và bài toán phủ k -submodular (kSC), một mở rộng của MCSC trên không gian lời giải đa tập. Thuộc Dạng 2, LA nghiên cứu bài toán SMFC, một biến thể của bài toán SM tổng quát trong đó ràng buộc $\mathcal{C}$ gồm ràng buộc ngân sách tổng và ràng buộc chi phí công bằng giữa các nhóm. Hai lớp bài toán MCSC và SM là nền tảng của nhiều mô hình tối ưu trong khai phá dữ liệu, học máy và trí tuệ nhân tạo hiện đại. Trong khi bài toán phủ submodular hướng tới việc đạt được một

mức lợi ích mong muốn với chi phí nhỏ nhất, thì bài toán tối đa submodular tập trung tìm kiếm tập nghiệm có giá trị lớn nhất dưới các ràng buộc về tài nguyên hoặc cấu trúc. Mặc dù có bản chất tối ưu khác nhau, cả hai đều khai thác sâu sắc các tính chất của hàm submodular và có mối liên hệ chặt chẽ về mặt lý thuyết cũng như kỹ thuật thiết kế thuật toán. Mối liên hệ giữa các bài toán này được minh họa trong Hình 0.1. Trên cơ sở đó, LA



Hình 0.1: Các dạng của bài toán SO và đóng góp của LA.

tập trung nghiên cứu định nghĩa bài toán, tính chất lý thuyết và đề xuất các thuật toán hiệu quả cho từng lớp bài toán tương ứng. Các kết quả đạt được như sau:

1. LA nghiên cứu bài toán MCSC và đề xuất thuật toán có khả năng song song hiệu quả đồng thời đạt được tỷ lệ tiệm cận với TTXS tốt nhất hiện nay [19] trong khi giảm rất đáng kể độ phức tạp song song cả về phương diện lý thuyết lẫn thực nghiệm. Các kết quả nghiên cứu về bài toán này được công bố trên tạp chí chuyên ngành về tối ưu hóa và vận trù học **Asia-Pacific Journal of Operational Research (SCIE/Q3)** năm 2024.

2. LA nghiên cứu bài toán kSC, một mở rộng của bài toán MCSC trên không gian lời giải đa tập

$$(k+1)^V = \{(V_1, V_2, \dots, V_k) \mid V_i \subseteq V, \forall i \in [k], V_i \cap V_j = \emptyset, \forall i \neq j\}.$$

Đối với bài toán này, LA đề xuất các TTXS tiêu chuẩn kép cho hai trường hợp chi phí đồng nhất và chi phí tùy ý. Về mặt lý thuyết, các thuật toán đề xuất đạt được các bảo

đảm xấp xỉ tốt hơn so với những kết quả đã được công bố trước đó. Bên cạnh đó, LA xây dựng hệ thống thực nghiệm và đánh giá các thuật toán trên nhiều bộ dữ liệu ứng dụng thực tế. Kết quả thực nghiệm cho thấy các phương pháp đề xuất không chỉ đạt chất lượng lời giải cao hơn mà còn có chi phí tính toán thấp hơn so với các phương pháp hiện có.

Các kết quả liên quan đến bài toán kSC đã được công bố trên tạp chí **Asia-Pacific Journal of Operational Research (SCIE/Q3)** năm 2026 và tại hội nghị **SoICT** năm 2025.

3. LA nghiên cứu bài toán tối đa hàm submodular với ràng buộc chi phí công bằng (SMFC), một biến thể của bài toán SM tổng quát (Dạng 2). Cụ thể, cho một hàm mục tiêu submodular đơn điệu $f : 2^V \rightarrow \mathbb{R}_+$, tập nền V được phân hoạch thành K nhóm $C_1, C_2, \dots, C_K$, một ngân sách tổng B và tham số công bằng $\alpha \in (0, 1)$. Mỗi phần tử $u \in V$ có chi phí dương $c(u)$. Mục tiêu là tìm tập con $S \subseteq V$ có giá trị $f(S)$ lớn nhất, đồng thời thỏa mãn các ràng buộc chi phí tổng là giới hạn $c(S) \leq B$, và ngân sách của từng nhóm không vượt quá αB , tức là $c(S \cap C_i) \leq \alpha B, \quad \forall i \in [K]$.

Đối với bài toán này, LA đề xuất các TTX cho cả trường hợp chi phí đồng nhất và chi phí tổng quát. Các kết quả lý thuyết chứng minh rằng các thuật toán đề xuất đạt được bảo đảm xấp xỉ hằng số đối với bài toán. Bên cạnh đó, LA tiến hành đánh giá thực nghiệm trên nhiều bộ dữ liệu mạng xã hội thực tế. Kết quả thu được cho thấy các phương pháp đề xuất đạt chất lượng lời giải cạnh tranh hoặc vượt trội so với các phương pháp hiện có, đồng thời duy trì hiệu quả tính toán cao.

Các kết quả liên quan đến bài toán SMFC đã được công bố tại hội nghị **Asian Conference on Intelligent Information and Database (ACIIDS), Rank B** năm 2025 và tại hội nghị **Symposium on Information and Communication Technology (SOICT)** năm 2024.

Các kết quả và đóng góp của LA đã được công bố thông qua **02** bài báo trên tạp chí SCIE, **03** bài báo đăng trong kỷ yếu các hội nghị quốc tế có phản biện thuộc danh mục SCOPUS, cùng với **01** bài báo trong kỷ yếu hội nghị quốc gia có phản biện.

Ngoài phần Mở đầu và Kết luận, nội dung chính của luận án được tổ chức thành bốn chương như sau:

Chương 1 trình bày cơ sở lý thuyết và tổng quan các nghiên cứu liên quan đến CO và tối ưu submodular, làm nền tảng cho các nội dung nghiên cứu của luận án.

Chương 2 nghiên cứu bài toán MCSC, đề xuất các thuật toán song song mới cùng các phân tích lý thuyết và thực nghiệm nhằm đánh giá chất lượng lời giải, độ phức tạp và khả

năng mở rộng.

Chương 3 nghiên cứu bài toán kSC, đề xuất các TTXX tiêu chuẩn kép cho cả trường hợp chi phí đồng nhất và chi phí tổng quát, kèm theo các phân tích lý thuyết và đánh giá thực nghiệm.

Chương 4 nghiên cứu bài toán SMFC, đề xuất các TTXX cho các mô hình chi phí khác nhau, đồng thời phân tích bảo đảm lý thuyết và kiểm chứng hiệu quả trên các bộ dữ liệu thực tế.

CHƯƠNG 1

TỔNG QUAN VỀ CÁC BÀI TOÁN TỐI ƯU SUBMODULAR

Chương này trình bày tổng quan về các bài toán Tối ưu tổ hợp (CO) và tối ưu submodular (SO), cùng với một số ứng dụng tiêu biểu như tóm tắt thông tin, tối đa hóa ảnh hưởng trên OSN và lựa chọn đặc trưng. Thông qua các ứng dụng này, chương làm rõ cách thức mô hình hóa các vấn đề thực tiễn dưới dạng các bài toán CO.

Tiếp theo, chương giới thiệu các khái niệm nền tảng, các mô hình thuật toán cơ bản và tổng quan các nghiên cứu tiêu biểu đối với lớp bài toán SO. Đồng thời, các đặc điểm quan trọng của hàm submodular và những dạng ràng buộc thường gặp cũng được trình bày.

Cuối cùng, chương tổng hợp các khái niệm về độ phức tạp tính toán và các kỹ thuật thiết kế thuật toán hiện đại cho SO. Những cơ sở lý thuyết này tạo nền tảng cho việc mô hình hóa, phát triển và phân tích các thuật toán được nghiên cứu trong các chương tiếp theo.

1.1. BÀI TOÁN TỐI ƯU TỔ HỢP

Các bài toán CO bắt nguồn từ những nhu cầu thực tiễn của con người trong sản xuất và đời sống. Trong nhiều tình huống khác nhau của xã hội, con người thường phải lựa chọn trong số nhiều khả năng các phương án tối ưu nhất, dưới các ràng buộc và điều kiện cụ thể. Nhu cầu này hiện diện rộng rãi trong nhiều lĩnh vực kinh tế – xã hội, bao gồm sản xuất, kinh doanh, quản lý, cũng như trong các quyết định thường nhật.

Nhằm mô hình hóa và xử lý các vấn đề nêu trên, nhiều bài toán kinh điển trong CO đã được phát triển. Tiêu biểu có thể kể đến các bài toán như xác định tuyến vận chuyển tối ưu về chi phí, xây dựng kế hoạch và lịch làm việc trong tổ chức, cũng như phân công nhiệm vụ và phân bổ nguồn lực. Thậm chí, những tình huống quen thuộc trong đời sống, chẳng hạn việc tìm đường đi ngắn nhất từ nhà đến nơi làm việc, cũng có thể được biểu diễn dưới dạng các mô hình CO. Nhờ khả năng mô tả linh hoạt các bài toán thực tiễn cùng tính khái quát cao, CO đã trở thành một lĩnh vực nền tảng trong khoa học máy tính.

Một cách tổng quát, có thể định nghĩa bài toán CO cụ thể như sau

Định nghĩa 1.1. Một bài toán CO được xác định bởi các thành phần sau.

Input:

- Một không gian nghiệm rời rạc hữu hạn Ω , bao gồm các lời giải hợp lệ của bài toán.
- Một hàm mục tiêu nhận giá trị thực không âm $f : \Omega \rightarrow \mathbb{R}_+$, trong đó $f(S)$ đo

lượng chất lượng hoặc lợi ích của nghiệm $S \in \Omega$.

Output:

- Một nghiệm $O \in \Omega$ sao cho $f(O)$ đạt giá trị tối ưu (tối đa hóa hoặc tối thiểu).

Bài toán cũng có thể được phát biểu dưới dạng

$$\max_{S \in \Omega} f(S) \quad (\text{hoặc } \min_{S \in \Omega} f(S)). \quad (1.1)$$

Dưới đây là một số khái niệm quan trọng về bài toán CO:

- *Lời giải hợp lệ (feasible solution)*: Một tập con $S \in \Omega$ được gọi là một lời giải chấp nhận được.

- *Lời giải tối ưu (optimal solution)*: Một lời giải chấp nhận được $O \in \Omega$ được gọi là lời giải tối ưu nếu $O \in \arg \max_{S \in \Omega} f(S)$ (hoặc $O \in \arg \min_{S \in \Omega} f(S)$).

- *Tập nền V* : Trong nhiều bài toán CO, không gian nghiệm Ω thường được xác định là tập các lời giải hợp lệ được xây dựng từ một tập nền V . Cụ thể, mỗi nghiệm là một tập con của V , do đó ta có: $\Omega \subseteq 2^V$, trong đó 2^V là tập lũy thừa của V , bao gồm tất cả các tập con của V .

Trong những năm gần đây, sự phát triển nhanh chóng không ngừng nghỉ của trí tuệ nhân tạo và khoa học dữ liệu đã làm gia tăng đáng kể vai trò của các bài toán CO, đặc biệt trong những bài toán gắn liền với dữ liệu quy mô lớn. Các nền tảng trực tuyến, OSN, hệ thống cảm biến và nhiều ứng dụng số khác liên tục sản sinh số lượng lớn dữ liệu cần được xử lý và phân tích, kéo theo những thách thức mới không chỉ về kích thước mà còn về mức độ phức tạp của bài toán. Trong bối cảnh đó, các phương pháp tối ưu không còn chỉ hướng đến việc tìm nghiệm tốt, mà còn phải đảm bảo hiệu quả tính toán, khả năng mở rộng và xử lý lượng lớn truy vấn dữ liệu trong thời gian hợp lý. Chính vì vậy, CO ngày càng giữ vai trò quan trọng, không chỉ trong các ứng dụng truyền thống mà còn ở vị trí trung tâm của nhiều bài toán then chốt trong phân tích dữ liệu và học máy hiện đại.

Khi nghiên cứu các thuật toán CO, một trong những khía cạnh được xem xét đầu tiên là độ phức tạp về thời gian thực thi của thuật toán.

Định nghĩa 1.2 (Độ phức tạp thời gian của thuật toán [38]). Xét một thuật toán $\mathcal{A}$ giải một bài toán với dữ liệu đầu vào có kích thước n . Độ phức tạp thời gian của thuật toán $\mathcal{A}$, ký hiệu $T_{\mathcal{A}}(n)$, được định nghĩa là số bước tính toán cơ bản nhiều nhất mà thuật toán cần thực hiện để xử lý mọi dữ liệu đầu vào có kích thước không vượt quá n .

Trong nhiều bài toán CO trong phân tích dữ liệu, giá trị của hàm mục tiêu f thường có thể được tính toán trong thời gian đa thức theo kích thước đầu vào. Ở một số trường

hợp thuận lợi, đặc biệt khi dữ liệu có cấu trúc đơn giản hoặc đã được tiền xử lý thích hợp, việc đánh giá f thậm chí có thể đạt độ phức tạp hằng số. Tuy vậy, đối với các ứng dụng thực tế ở quy mô lớn, chi phí dành cho việc tính toán hàm mục tiêu vẫn thường chiếm tỷ trọng đáng kể trong tổng thời gian thực thi của thuật toán.

Chẳng hạn, trong các bài toán tối đa hóa ảnh hưởng, việc tính số lượng người dùng bị ảnh hưởng bởi một tập hạt giống đòi hỏi mô phỏng quá trình lan truyền trên mạng, vốn là một bước tính toán rất tốn kém [1]. Mô hình đo độ phức tạp dựa trên số lần truy vấn (query complexity) có nguồn gốc từ mô hình cây quyết định trong lý thuyết độ phức tạp tính toán từ những năm 1950, và sau đó được phát triển thành mô hình oracle trong tối ưu hóa bởi Nemirovsky và Yudin (1983) [15]. Hiện nay, mô hình này được sử dụng rộng rãi trong các thuật toán tối ưu quy mô lớn, đặc biệt trong tối ưu hóa submodular, nơi chi phí chủ yếu nằm ở việc đánh giá hàm mục tiêu.

Định nghĩa 1.3 (Độ phức tạp truy vấn (Query complexity) [7, 32, 38]). Xét một thuật toán $\mathcal{A}$ giải một bài toán với dữ liệu đầu vào có kích thước n . Giả sử hàm mục tiêu f chỉ được truy cập thông qua một mô hình truy vấn cho trước (hộp đen, black-box), trong đó mỗi truy vấn với tập $S \subseteq V$ trả về giá trị $f(S)$. Độ phức tạp truy vấn của thuật toán $\mathcal{A}$, ký hiệu $Q_{\mathcal{A}}(n)$, là số lần tối đa thuật toán gọi hàm mục tiêu f thông qua mô hình truy vấn đã cho, xét trên mọi dữ liệu đầu vào có kích thước không vượt quá n .

Ta có thể so sánh đặc điểm của độ phức tạp thời gian và độ phức tạp truy vấn qua Bảng 1.1.

Khi quy mô bài toán tăng, số lượng truy vấn hàm mục tiêu mà thuật toán phải thực hiện thường tăng theo, đồng thời mỗi truy vấn cũng trở nên tốn kém hơn do phải xử lý nhiều dữ liệu hơn. Sự gia tăng đồng thời này có thể khiến tổng thời gian tính toán tăng mạnh, làm hạn chế khả năng áp dụng thuật toán trên dữ liệu quy mô lớn. Vì vậy, giảm số lượng truy vấn hàm mục tiêu là một hướng quan trọng để nâng cao hiệu quả tính toán của thuật toán.

Trong thực tế, việc xác định chính xác hàm độ phức tạp thời gian, truy vấn $T_{\mathcal{A}}(n)$ và $Q_{\mathcal{A}}(n)$ của một thuật toán thường rất khó do phụ thuộc vào nhiều yếu tố như cấu trúc dữ liệu, chi tiết cài đặt và môi trường thực thi. Vì vậy, trong phân tích thuật toán, người ta thường sử dụng các ký hiệu tiệm cận như $O(\cdot)$, $\Omega(\cdot)$ và $\Theta(\cdot)$ để mô tả tốc độ tăng của thời gian thực thi theo kích thước dữ liệu đầu vào [38].

1.2. BÀI TOÁN TỐI ƯU SUBMODULAR

Các bài toán CO trong thực tế rất đa dạng. Trong phạm vi đề tài, LA tập trung vào các bài toán phát sinh trong các lĩnh vực khoa học dữ liệu, OSN, học máy và khai phá dữ

Bảng 1.1: So sánh độ phức tạp thời gian và độ phức tạp truy vấn.

Tiêu chí	Độ phức tạp thời gian	Độ phức tạp truy vấn
Ký hiệu	$T_A(n)$	$Q_A(n)$
Đại lượng đo	Số bước tính toán cơ bản nhiều nhất mà thuật toán thực hiện.	Số lần truy vấn hàm mục tiêu f nhiều nhất mà thuật toán thực hiện.
Phạm vi tính toán	Bao gồm toàn bộ các phép tính của thuật toán, kể cả chi phí đánh giá hàm mục tiêu f .	Chỉ đếm số lần gọi hàm mục tiêu f thông qua hộp đen; không tính chi phí bên trong mỗi lần gọi.
Các yếu tố ảnh hưởng	Kích thước đầu vào, cách triển khai và chi phí của các phép tính.	Kích thước đầu vào, chiến lược truy vấn và mô hình truy cập hàm mục tiêu.
Ý nghĩa	Phản ánh tổng chi phí tính toán để thuật toán hoàn thành.	Phản ánh số lần thuật toán cần thu thập thông tin từ hàm mục tiêu.
Trường hợp sử dụng	Khi cần phân tích tổng chi phí tính toán của thuật toán.	Khi việc đánh giá hàm mục tiêu tốn kém hoặc hàm mục tiêu được truy cập qua hộp đen.

liệu, đặc biệt trong bối cảnh dữ liệu quy mô lớn với hàm mục tiêu hoặc hàm ràng buộc có tính chất submodular. Lớp bài toán này thường được gọi chung là các bài toán tối ưu submodular (SO).

Một số ứng dụng tiêu biểu của SO bao gồm tối đa hóa ảnh hưởng trên OSN nhằm thúc đẩy quá trình lan truyền thông tin hoặc quảng bá sản phẩm [1, 2]; tối đa hóa doanh thu trong các chiến dịch tiếp thị lan truyền [6, 3, 4]; tóm tắt dữ liệu và nội dung đa phương tiện nhằm giảm kích thước dữ liệu nhưng vẫn duy trì các thông tin quan trọng [3, 4, 5], cũng như lựa chọn tập đặc trưng để nâng cao hiệu quả của các mô hình học máy [39, 40].

1.2.1. Định nghĩa bài toán

Dưới góc độ mô hình hóa toán học, bài toán SO thường được định nghĩa dựa trên các khái niệm sau:

- *Tập nền* $V = \{e_1, e_2, \dots, e_n\}$: tập hữu hạn gồm n phần tử hữu hạn, trong đó mỗi phần tử e_i đại diện cho một đối tượng ứng viên cần được lựa chọn. Tùy theo từng bài toán cụ thể, các phần tử này có thể tương ứng với người dùng trong OSN, sản phẩm trong hệ thống gợi ý, các điểm dữ liệu, các đặc trưng trong học máy, hoặc các vị trí tiềm năng để đặt cảm biến.
- *Ràng buộc* C : Nhằm mô hình hóa các yêu cầu và hạn chế trong thực tế. Các ràng buộc này có thể phản ánh những giới hạn về tài nguyên, chi phí hoặc cấu trúc, và thường xuất hiện dưới các dạng quen thuộc như ràng buộc lực lượng, ràng buộc

chi phí hoặc ngân sách, hay các ràng buộc phức tạp hơn như ràng buộc công bằng, ràng buộc phân hoạch hoặc ràng buộc matroid.

- *Họ các tập con* 2^V : Tập hợp tất cả các tập con có thể của tập V .
- *Không gian lời giải hợp lệ*: $\Omega = \{S \subseteq V : S \text{ thỏa mãn } \mathcal{C}\}$.
- *Hàm mục tiêu f , đánh giá g* : Trên không gian này, ta xác định hàm mục tiêu, ràng buộc nhận giá trị thực không âm.

$$f, g : 2^V \rightarrow \mathbb{R}_+,$$

trong đó f đo lường chất lượng hoặc lợi ích của mỗi tập con và mức thỏa mãn ràng buộc $\mathcal{C}$ của lời giải.

Một đặc điểm chung của nhiều bài toán tối ưu trong phân tích dữ liệu là hàm mục tiêu hoặc hàm ràng buộc có tính chất *submodular* [1, 13, 41]. Tính chất này xuất hiện tự nhiên trong nhiều ứng dụng như đặt cảm biến, tối đa hóa ảnh hưởng trên OSN, tìm lát cắt cực đại và lựa chọn đặc trưng trong học máy. Nhờ khai thác cấu trúc submodular, nhiều bài toán NP-khó có thể được giải bằng các thuật toán hiệu quả với bảo đảm xấp xỉ chặt chẽ.

Hàm submodular là một lớp hàm tập hợp quan trọng trong CO, đặc trưng bởi tính chất *lợi ích biên giảm dần*: lợi ích khi bổ sung một phần tử vào tập nhỏ thường không nhỏ hơn lợi ích khi bổ sung phần tử đó vào một tập lớn hơn. Tính chất này phản ánh nhiều hiện tượng thực tế, chẳng hạn lợi ích của việc bổ sung cảm biến mới hoặc lựa chọn thêm người dùng hạt giống trong các bài toán lan truyền thông tin sẽ giảm dần khi quy mô hệ thống tăng lên. Do đó, hàm submodular đã trở thành một công cụ mô hình hóa nền tảng trong khoa học dữ liệu, học máy và khoa học mạng lưới.

Định nghĩa 1.4 (Hàm submodular [42, 15]). Cho một tập hữu hạn V , một hàm tập hợp $f : 2^V \rightarrow \mathbb{R}$ được gọi là *submodular* nếu với mọi cặp tập con $A \subseteq B \subseteq V$ và với mọi phần tử $e \in V \setminus B$, điều kiện sau được thỏa mãn:

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

Bất đẳng thức trên thể hiện rõ tính chất lợi ích biên giảm dần của hàm f .

Một cách phát biểu tương đương của tính chất submodular là:

$$f(A) + f(B) \geq f(A \cup B) + f(A \cap B), \quad \forall A, B \subseteq V.$$

Hai cách định nghĩa này là tương đương và thường được sử dụng luân phiên trong phân tích lý thuyết. Ngoài ra, một hàm submodular f được gọi là *đơn điệu* nếu

$$f(A) \leq f(B), \quad \forall A \subseteq B \subseteq V.$$

Tính đơn điệu thường xuất hiện trong các bài toán đo lường thông tin hoặc mức độ bao phủ, trong đó việc mở rộng tập lựa chọn không làm giảm giá trị hàm mục tiêu.

Trên cơ sở các thành phần đã nêu ở trên, một lớp lớn các bài toán CO trong phân tích dữ liệu có thể được phát biểu một cách tổng quát như sau.

Định nghĩa 1.5 (Bài toán SO). Một bài toán CO được xác định bởi các thành phần sau.

Input:

- Một tập nền hữu hạn $V = \{e_1, e_2, \dots, e_n\}$, gồm n phần tử
- Ràng buộc $\mathcal{C}$, với hàm đánh giá ràng buộc g là submodular. Không gian lời giải hợp lệ là $\Omega_{\mathcal{C}} = \{S \subseteq V, S \text{ thỏa mãn } \mathcal{C}\}$.
- Một hàm mục tiêu $f : 2^V \rightarrow \mathbb{R}$, là submodular.

Output: Một tập con $S \subseteq V$ sao cho S thỏa mãn $\mathcal{C}$ và $f(S)$ đạt giá trị tối ưu theo yêu cầu của bài toán (tối đa hóa hoặc tối thiểu hóa).

Bài toán có thể được phát biểu dưới dạng:

$$\max_{S \subseteq V} f(S) \quad (\text{hoặc } \min_{S \subseteq V} f(S)) \quad (1.2)$$

$$\text{thỏa mãn: } S \in \Omega_{\mathcal{C}}. \quad (1.3)$$

Từ khuôn khổ tổng quát của bài toán SO , bài toán này thể được phân thành hai dạng cơ bản tùy theo mục tiêu tối ưu, bao gồm dạng tối thiểu hóa chi phí để đạt một mức chất lượng yêu cầu và dạng tối đa hóa chất lượng dưới các ràng buộc tài nguyên.

Dạng 1 (tối thiểu): Bài toán phủ Submodular với chi phí tối thiểu (MCSC))

Input:

- Một tập nền hữu hạn $V = \{e_1, e_2, \dots, e_n\}$, gồm n phần, mỗi phần tử $e_i \in V$ có một chi phí $c(e_i) > 0$.
- Với tập con $S \subseteq V$, hàm mục tiêu là tổng chi phí $f(S) = \sum_{e \in S} c(e)$.
- Hàm ràng buộc $g : 2^V \mapsto \mathbb{R}_+$ là submodular, đánh giá giá trị của tập con S .
- Ngưỡng giá trị T .

Output:

- Một tập con $S \subseteq V$ sao cho $g(S) \geq T$.

Bài toán thường được phát biểu dưới dạng ngắn gọn

$$\min_{S \subseteq V} f(S) \quad (1.4)$$

$$\text{thỏa mãn: } g(S) \geq T \quad (1.5)$$

Dạng bài toán này thường xuất hiện trong các ứng dụng như tóm tắt dữ liệu, lựa chọn tập đại diện, hoặc đảm bảo mức bao phủ thông tin tối thiểu.

Trong dạng thứ hai, mục tiêu là tìm một tập con hợp lệ sao cho giá trị hàm mục tiêu đạt lớn nhất dưới các ràng buộc tài nguyên hoặc kích thước.

Dạng 2 (tối đa): Bài toán tối đa hóa hàm submodular với ràng buộc (SM).

Input:

- Một tập nền hữu hạn $V = \{e_1, e_2, \dots, e_n\}$, gồm n phần.
- Với tập con $S \subseteq V$, hàm mục tiêu $f(\cdot) : 2^V \mapsto \mathbb{R}_+$.
- Ràng buộc $\mathcal{C}$.

Output:

- Một tập con $S \subseteq \Omega_{\mathcal{C}}$ sao cho $f(S)$ đạt giá trị lớn nhất.

Bài toán thường được phát biểu dưới dạng ngắn gọn

$$\max_{S \subseteq V} f(S) \quad (1.6)$$

$$\text{thỏa mãn: } S \in \Omega_{\mathcal{C}} \quad (1.7)$$

Trong thực tế, ràng buộc $\mathcal{C}$ thường biểu diễn giới hạn về kích thước tập chọn, chi phí, hoặc các điều kiện công bằng giữa các nhóm dữ liệu. Dạng bài toán này bao gồm nhiều bài toán điển hình trong phân tích dữ liệu như tối đa hóa ảnh hưởng trong OSN, lựa chọn đặc trưng trong học máy, và lựa chọn tập đại diện nhằm tối đa hóa thông tin thu được dưới giới hạn tài nguyên.

Bài toán trên có thể xuất hiện dưới hai dạng cơ bản là bài toán tối đa hóa hoặc bài toán tối thiểu hóa hàm mục tiêu, tùy thuộc vào bối cảnh ứng dụng cụ thể.

Một số dạng ràng buộc $\mathcal{C}$ thường gặp trong các bài toán CO bao gồm:

- *Ràng buộc kích thước (Cardinality constraint)*: Lời giải bị giới hạn về số lượng phần tử được chọn, tức là $|S| \leq k$, với k là số phần tử tối đa được phép chọn.
- *Ràng buộc Chi phí (Knapsack)*: Mỗi phần tử $e \in V$ có một chi phí $c(e) > 0$, và tổng chi phí của các phần tử được chọn không vượt quá ngân sách B , tức là $\sum_{e \in S} c(e) \leq B$.

- *Ràng buộc matroid*: Một matroid được định nghĩa bởi một cặp $\mathcal{M} = (V, \mathcal{I})$, trong đó V là tập nền hữu hạn và $\mathcal{I} \subseteq 2^V$ là họ các tập độc lập (các nghiệm hợp lệ), thỏa mãn hai tính chất cơ bản sau:

- (i) *Tính chất di truyền (hereditary property)*: Nếu $A \in \mathcal{I}$ và $B \subseteq A$ thì $B \in \mathcal{I}$.
- (ii) *Tính chất trao đổi (exchange property)*: Nếu $A, B \in \mathcal{I}$ và $|A| < |B|$, thì tồn tại một phần tử $e \in B \setminus A$ sao cho $A \cup \{e\} \in \mathcal{I}$.

1.2.2. Một số bài toán cụ thể

Dưới đây là một số bài toán tiêu biểu thuộc lớp các bài toán SO thường gặp trong thực tế.

1.2.2.1. Bài toán IM trên OSN

Bài toán tối đa hóa ảnh hưởng (IM) là một hướng nghiên cứu quan trọng trong lĩnh vực phân tích OSN và khai phá dữ liệu. Vấn đề này được ứng dụng rộng rãi trong nhiều bối cảnh thực tiễn như tiếp thị lan truyền, xây dựng hệ thống khuyến nghị, mô hình hóa sự lan tỏa thông tin trên OSN, cũng như kiểm soát sự lây lan của dịch bệnh. Mục tiêu cốt lõi của bài toán là xác định một tập nhỏ các nút ban đầu sao cho mức độ lan truyền ảnh hưởng từ tập này đạt giá trị lớn nhất theo một mô hình lan truyền đã được xác định trước.

Trong số các mô hình lan truyền đã được đề xuất, hai mô hình tiêu biểu và được sử dụng phổ biến là mô hình lan truyền độc lập (Independent Cascade – IC) và mô hình ngưỡng tuyến tính (Linear Threshold – LT) [43, 1]. Trong nghiên cứu này, trọng tâm được đặt vào mô hình IC do đặc tính xác suất tự nhiên cùng với tính chất submodular rõ ràng của hàm ảnh hưởng. Từ đó, bài toán tối đa hóa ảnh hưởng được phát biểu chính thức như sau:

Định nghĩa 1.6 (Bài toán IM [43, 1]). Cho một đồ thị có hướng $G = (V, E)$ biểu diễn OSN cùng với một mô hình lan truyền thông tin cố định $\mathcal{M}$. Mục tiêu của bài toán là tìm một tập nút khởi tạo (hay tập hạt giống) $S \subseteq V$ sao cho mức độ ảnh hưởng lan truyền từ S lên toàn bộ mạng, được đo bằng hàm ảnh hưởng $f(S)$, được tối đa hóa.

Bài toán IM thuộc lớp bài toán SM. Trong thực tế, mô hình lan truyền $\mathcal{M}$ thường được giả định thuộc một trong hai khuôn khổ kinh điển là mô hình Bậc độc lập (Independent Cascade-IC) hoặc Ngưỡng tuyến tính (Linear Threshold-LT), vốn được giới thiệu và nghiên cứu một cách hệ thống trong công trình của Kempe và cộng sự [1].

Mô hình IC. Trong mô hình IC, mỗi cạnh có hướng $(u, v) \in E$ được gán một xác suất

lan truyền $p_{u,v} \in [0, 1]$, biểu thị khả năng mà nút u có thể kích hoạt nút v . Quá trình lan truyền diễn ra theo từng bước rời rạc. Ở mỗi bước, nếu một nút u vừa được kích hoạt ở bước trước đó, nó chỉ có một cơ hội duy nhất để kích hoạt từng nút kề chưa được kích hoạt v , với xác suất $p_{u,v}$. Nếu lần thử này không thành công, cạnh (u, v) sẽ không được xem xét lại trong các bước tiếp theo. Quá trình tiếp diễn cho đến khi không còn nút nào có thể kích hoạt thêm nút mới.

Với một tập hạt giống $S \subseteq V$, ký hiệu $A_{\mathcal{M}}(S)$ là tập các nút được kích hoạt khi quá trình lan truyền khởi phát từ S dưới mô hình $\mathcal{M} \in \{\text{IC}, \text{LT}\}$. Hàm ảnh hưởng được định nghĩa bởi

$$f(S) := \mathbb{E}[|A_{\mathcal{M}}(S)|], \quad (1.8)$$

trong đó kỳ vọng được lấy theo tính ngẫu nhiên của mô hình lan truyền (xác suất kích hoạt trong IC hoặc ngưỡng trong LT). Trong thực tế, $f(S)$ thường được ước lượng bằng mô phỏng Monte-Carlo:

$$\widehat{f}(S) := \frac{1}{R} \sum_{r=1}^R |A_{\mathcal{M}}^{(r)}(S)|, \quad (1.9)$$

với $A_{\mathcal{M}}^{(r)}(S)$ là tập nút kích hoạt ở lần mô phỏng thứ r và R là số lần mô phỏng.

Mô hình LT Trong mô hình LT, mỗi cạnh $(u, v) \in E$ được gán một trọng số $w_{u,v} \geq 0$, thỏa mãn $\sum_{u:(u,v) \in E} w_{u,v} \leq 1$ với mọi nút $v \in V$. Mỗi nút v được gán một ngưỡng kích hoạt θ_v , trong đó θ_v là một biến ngẫu nhiên được chọn độc lập và phân bố đều trong khoảng $[0, 1]$. Một nút chưa được kích hoạt sẽ trở nên kích hoạt khi tổng trọng số từ các nút lân cận đã được kích hoạt vượt quá ngưỡng θ_v . Quá trình lan truyền tiếp tục cho đến khi không còn nút nào thỏa mãn điều kiện kích hoạt.

Trong cả hai mô hình IC và LT, hàm ảnh hưởng $f(S)$ được định nghĩa là kỳ vọng số lượng nút được kích hoạt khi quá trình lan truyền bắt đầu từ tập hạt giống S . Kempe và cộng sự đã chỉ ra rằng hàm $f(S)$ trong hai mô hình này đều là hàm đơn điệu và submodular, tạo cơ sở lý thuyết quan trọng cho việc thiết kế các TTX hiệu quả cho bài toán IM [1].

Với mỗi nút v , chọn ngẫu nhiên một *tập kích hoạt* $T(v) \subseteq N^-(v)$ từ một phân phối phù hợp với các trọng số $\{w_{u,v}\}_{u \in N^-(v)}$. Một cách cài đặt phổ biến là: chọn nhiều nhất một nút $u \in N^-(v)$ làm “tác nhân kích hoạt” cho v , trong đó v chọn u với xác suất $w_{u,v}$ và chọn “không ai” với xác suất $1 - \sum_{u \in N^-(v)} w_{u,v}$. Từ đó thu được một đồ thị ngẫu nhiên G^{LT} gồm các cạnh (u, v) sao cho $u \in T(v)$. Với một hiện thực G^{LT} , quá trình lan truyền kích hoạt đúng các nút có thể đi tới từ S , tức là $\text{Reach}_{G^{\text{LT}}}(S)$, và vì thế

$$f(S) = \mathbb{E}_{G^{\text{LT}}} [|\text{Reach}_{G^{\text{LT}}}(S)|]. \quad (1.10)$$

Các nghiên cứu đã chứng minh rằng, dưới hai mô hình lan truyền LT và IC, hàm ảnh hưởng (f) là hàm submodular và đơn điệu tăng [1].

1.2.2.2. Bài toán Ảnh hưởng ngưỡng

Bài toán *Ảnh hưởng ngưỡng* (Threshold Influence–TI) là một biến thể quan trọng của bài toán tối đa hóa ảnh hưởng trong OSN, trong đó mục tiêu không chỉ là lan truyền thông tin đến càng nhiều nút càng tốt, mà còn yêu cầu mức độ ảnh hưởng đạt đến hoặc vượt qua một ngưỡng xác định. Mô hình này phù hợp với nhiều kịch bản thực tế, chẳng hạn như chiến dịch tiếp thị cần đạt được một mức độ nhận biết tối thiểu, hoặc các ứng dụng lan truyền thông tin đòi hỏi một tỷ lệ chấp nhận đủ lớn trong mạng [20].

Xét một đồ thị có hướng $G = (V, E)$ biểu diễn OSN, cùng với một mô hình lan truyền thông tin $\mathcal{M}$. Với mỗi tập hạt giống $S \subseteq V$, ký hiệu $f(S)$ là kỳ vọng số lượng nút được kích hoạt khi quá trình lan truyền kết thúc, hay nói cách khác là giá trị ảnh hưởng của S .

Định nghĩa 1.7 (Bài toán IT). Cho một OSN $G = (V, E)$, một mô hình lan truyền thông tin $\mathcal{M}$, và một ngưỡng T . Mục tiêu là tìm một tập hạt giống $S \subseteq V$ sao cho

$$f(S) \geq T$$

với lượng ngân sách là tối thiểu, chẳng hạn như số lượng tập hạt giống $|S|$ hoặc tổng chi phí bỏ ra $\sum_{v \in S} c(v)$.

Bài toán IT là một trường hợp của bài toán MCSC, có nhiều ứng dụng trong đầu tư, tiếp thị lan truyền và tóm tắt thông tin, khi mục tiêu là đạt được một mức độ ảnh hưởng hoặc lượng thông tin tối thiểu đã định trước.

1.2.2.3. Bài toán Tối đa doanh thu

Bài toán tối đa doanh thu (*Revenue Maximization–RM*) trong các chiến lược tiếp thị lan truyền trên OSN không chỉ hướng tới mục tiêu mở rộng phạm vi lan tỏa thông tin, mà còn nhằm tối ưu hóa tổng doanh thu thu được từ chiến dịch. Khác với bài toán IM, vốn chỉ đo lường số lượng người dùng được kích hoạt, bài toán tối đa doanh thu còn xét đến mức độ đóng góp kinh tế của từng người dùng sau khi bị ảnh hưởng. Điều này thường được mô hình hóa thông qua các hàm doanh thu phi tuyến, cho phép phản ánh hiện tượng bão hòa trong hành vi tiếp nhận và tiêu dùng sản phẩm của người dùng.

Bài toán được mô hình hóa trên OSN biểu diễn dưới dạng đồ thị như sau.

Định nghĩa 1.8 (Bài toán RM [44, 45]). Cho một OSN được mô hình hóa bởi đồ thị có hướng $G = (V, E)$, trong đó mỗi cạnh $(u, v) \in E$ được gán một trọng số không âm $w_{u,v}$,

thể hiện mức độ ảnh hưởng của người dùng u lên người dùng v . Với mỗi nút $u \in V$, giả sử tồn tại một hàm doanh thu $R_u(S)$ phụ thuộc vào tập người dùng $S \subseteq V$ đã được kích hoạt hoặc mua sản phẩm. Tổng doanh thu thu được từ tập S được xác định bởi

$$f(S) = \sum_{u \in V} R_u(S). \quad (1.11)$$

Bài toán RM yêu cầu xác định một tập hạt giống $S \subseteq V$ có kích thước không vượt quá k nhằm tối đa hóa tổng doanh thu thu được.

Bài toán RM thuộc nhóm SM và có nhiều ứng dụng trong các hệ thống gợi ý, tóm tắt dữ liệu, lựa chọn đặc trưng và phân bổ tài nguyên, khi mục tiêu là lựa chọn một tập phần tử có giá trị lớn nhất dưới các ràng buộc về nguồn lực hoặc ngân sách. Trong bài toán này hàm mục tiêu có thể không đơn điệu.

1.2.2.4. Bài toán tóm tắt thông tin ảnh

Sự phổ biến của các nền tảng chia sẻ trực tuyến như TikTok, Facebook và Instagram đã kéo theo sự gia tăng nhanh chóng của dữ liệu hình ảnh. Điều này đặt ra nhu cầu ngày càng rõ rệt đối với các phương pháp tổ chức và tóm lược những bộ sưu tập ảnh có quy mô lớn. Trong bối cảnh đó, bài toán tóm tắt hình ảnh hướng đến việc lựa chọn một tập con có tính đại diện cao từ tập dữ liệu ban đầu, sao cho vẫn phản ánh được cả nội dung chính lẫn sự đa dạng của toàn bộ bộ ảnh.

Bên cạnh việc hỗ trợ lưu trữ và chia sẻ cá nhân, các phương pháp tóm tắt hình ảnh còn đóng vai trò quan trọng trong nhiều ứng dụng như tóm tắt video, hệ thống giám sát an ninh và nhận diện cảnh. Một bản tóm tắt hiệu quả cần đồng thời đáp ứng hai tiêu chí quan trọng: mức độ đại diện (fidelity) và tính đa dạng (diversity) của các phần tử được chọn [21, 3, 10, 13, 5]. Từ góc độ tối ưu hóa, bài toán Tóm tắt thông tin ảnh (Image Summarization-IS) này thường được mô hình hóa như một bài toán lựa chọn tập con có ràng buộc về kích thước

Định nghĩa 1.9 (Bài toán IS [21, 3, 10, 13, 5]). Xét một tập ảnh ban đầu V cùng với một ngưỡng kích thước k cho trước. Bài toán đặt ra là xác định một tập con $S \subseteq V$ thỏa mãn $|S| \leq k$ nhằm tối đa hóa một hàm đo lường mức độ thông tin của tập ảnh được chọn, tức là

$$\max_{S \subseteq V, |S| \leq k} f(S), \quad (1.12)$$

trong đó $f : 2^V \rightarrow \mathbb{R}_+$ là hàm đánh giá chất lượng tóm tắt.

Một dạng cụ thể của hàm f thường được sử dụng trong bài toán tóm tắt hình ảnh

được xác định như sau:

$$f(S) = \sum_{u \in V} \max_{v \in S} w_{u,v} - \frac{1}{|V|} \sum_{u \in V} \sum_{v \in S} w_{u,v}. \quad (1.13)$$

Ở đây, $w_{u,v}$ biểu diễn mức độ tương đồng giữa hai ảnh u và v . Giá trị này được suy ra từ độ tương phản (*contrast*) của các ảnh tương ứng theo công thức

$$w_{u,v} = 1 - \frac{|\text{RMS}(u) - \text{RMS}(v)|}{\max\{\text{RMS}(u), \text{RMS}(v)\}}. \quad (1.14)$$

Độ tương phản của một ảnh u được đo thông qua chỉ số $\text{RMS}(u)$, được tính bằng phương sai của cường độ điểm ảnh:

$$\text{RMS}(u) = \sqrt{\frac{1}{N} \sum_{i=1}^N (I_i - \mu)^2}, \quad (1.15)$$

trong đó I_i là cường độ của pixel thứ i trong ảnh u , N là tổng số pixel, và μ ký hiệu cho giá trị cường độ trung bình của ảnh. Hàm mục tiêu F trong mô hình trên thường có tính chất *submodular*, phản ánh hiện tượng lợi ích biên giảm dần khi bổ sung thêm ảnh mới vào tập tóm tắt. Bên cạnh các ứng dụng đã nêu bài toán SO còn có nhiều ứng dụng khác như tìm lát cắt cực đại [46, 13], đặt cảm biến tối ưu [41].

1.2.3. Hàm k -submodular: mở rộng của hàm submodular

Trong nhiều bài toán thực tiễn, hàm submodular được mở rộng thành hàm k -submodular nhằm mô hình hóa các vấn đề có hai đặc trưng chính: (i) tồn tại k loại hoặc k phương án khác nhau; và (ii) mỗi phần tử, nếu được lựa chọn, phải được gán vào đúng một trong k loại đó.

Cụ thể, cho tập nền $V = \{e_1, e_2, \dots, e_n\}$ và một số nguyên dương k . Ký hiệu $[k] = \{1, 2, \dots, k\}$ và đặt

$$(k+1)^V = \{(V_1, V_2, \dots, V_k) \mid V_i \subseteq V, \forall i \in [k], \quad V_i \cap V_j = \emptyset, \forall i \neq j\}.$$

Mỗi phần tử của $(k+1)^V$ là một bộ gồm k tập con đôi một rời nhau của V và được gọi là một **k -tập** (k -set). Với $\mathbf{x} = (X_1, X_2, \dots, X_k) \in (k+1)^V$, ta định nghĩa $\text{supp}_i(\mathbf{x}) = X_i$ và $\text{supp}(\mathbf{x}) = \bigcup_{i \in [k]} X_i$. Tập X_i được gọi là **tập thành phần thứ i** của $\mathbf{x}$, còn $\text{supp}(\mathbf{x})$ là tập tất cả các phần tử đã được gán vào một trong k vị trí. Ngoài ra, ký hiệu $\mathbf{0} = (\emptyset, \dots, \emptyset)$ được dùng để chỉ k -tập rỗng.

Nếu $e \in X_i$, ta viết $\mathbf{x}(e) = i$ và nói rằng i là **vị trí** của phần tử e trong $\mathbf{x}$. Ngược lại, nếu $e \notin \text{supp}(\mathbf{x})$ thì ta quy ước $\mathbf{x}(e) = 0$, nghĩa là phần tử e chưa được lựa chọn.

Với $e \notin \text{supp}(\mathbf{x})$, việc thêm phần tử e vào tập thành phần X_i được ký hiệu bởi $\mathbf{x} \sqcup (e, i)$. Cụ thể, $\mathbf{x} \sqcup (e, i) = (X_1, \dots, X_{i-1}, X_i \cup \{e\}, X_{i+1}, \dots, X_k)$. Ta cũng có thể biểu diễn một k -tập dưới dạng $\mathbf{x} = \{(e_1, i_1), (e_2, i_2), \dots, (e_t, i_t)\}$, trong đó $e_j \in \text{supp}(\mathbf{x})$ và $i_j = \mathbf{x}(e_j)$ với mọi $1 \leq j \leq t$. Trong trường hợp $X_i = \{e\}$ và $X_j = \emptyset$ với mọi $j \neq i$, k -tập tương ứng được ký hiệu gọn là (e, i) .

Với hai k -tập $\mathbf{x} = (X_1, X_2, \dots, X_k)$ và $\mathbf{y} = (Y_1, Y_2, \dots, Y_k)$ thuộc $(k+1)^V$, ta viết $\mathbf{x} \sqsubseteq \mathbf{y}$ nếu và chỉ nếu $X_i \subseteq Y_i, \forall i \in [k]$. Trong bối cảnh nghiên cứu ta giả sử hàm g là không âm, tức là $g(\mathbf{x}) \geq 0, \forall \mathbf{x} \in (k+1)^V$, và được chuẩn hóa, nghĩa là $g(\mathbf{0}) = 0$. Một hàm $g : (k+1)^V \rightarrow \mathbb{R}_+$ được gọi là **k -submodular** nếu với mọi $\mathbf{x} = (X_1, X_2, \dots, X_k), \mathbf{y} = (Y_1, Y_2, \dots, Y_k)$ thuộc $(k+1)^V$, ta có

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

trong đó phép gặp $\sqcap$ được định nghĩa bởi

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

và phép hợp $\sqcup$ được định nghĩa bởi

$$\mathbf{x} \sqcup \mathbf{y} = (Z_1, \dots, Z_k),$$

với $Z_i = (X_i \cup Y_i) \setminus \bigcup_{\substack{j \in [k] \\ j \neq i}} (X_j \cup Y_j), \forall i \in [k]$. Trong phép hợp $\mathbf{x} \sqcup \mathbf{y}$, một phần tử được giữ lại ở vị trí i nếu nó được gán vào vị trí i trong ít nhất một trong hai k -tập và không được gán vào bất kỳ vị trí nào khác. Nếu một phần tử được gán vào hai vị trí khác nhau trong $\mathbf{x}$ và $\mathbf{y}$, phần tử đó bị loại khỏi kết quả do xảy ra xung đột. Đặc biệt, khi $k = 1$, khái niệm k -submodular trở thành khái niệm submodular thông thường.

Ví dụ, trong bài toán tối đa hóa ảnh hưởng với nhiều chiến dịch, tập nền V biểu diễn tập người dùng trong mạng xã hội, còn k biểu diễn số sản phẩm, chủ đề hoặc chiến dịch truyền thông. Mỗi người dùng được lựa chọn làm hạt giống chỉ có thể được gán cho một chiến dịch. Khi đó, X_i là tập người dùng hạt giống của chiến dịch thứ i , và các tập $X_1, X_2, \dots, X_k$ phải đôi một rời nhau. Hàm mục tiêu $g(X_1, X_2, \dots, X_k)$ có thể biểu diễn tổng số người dùng kỳ vọng được tác động bởi các chiến dịch. Trong nhiều mô hình lan truyền, hàm mục tiêu này thể hiện tính chất lợi ích biên giảm dần và có thể được mô hình hóa dưới dạng một hàm k -submodular.

1.3. THUẬT TOÁN XẤP XỈ

Thuật toán xấp xỉ (TTXX) là một hướng tiếp cận quan trọng để giải quyết các bài toán CO NP-khó, trong đó mục tiêu là tìm nghiệm gần tối ưu cùng với các bảo đảm lý

thuyết về chất lượng nghiệm. Đặc biệt, trong bối cảnh dữ liệu quy mô lớn và các bài toán có cấu trúc phức tạp, các bảo đảm này đóng vai trò quan trọng trong việc đánh giá và so sánh hiệu quả của thuật toán.

Trong LA, NCS tập trung nghiên cứu và phát triển các TTX cho các bài toán CO trong phân tích dữ liệu quy mô lớn, hướng tới việc cân bằng giữa chất lượng nghiệm theo lý thuyết và hiệu quả tính toán trong thực tế.

1.3.1. Định nghĩa

Xét một bài toán tối ưu trong đó mỗi nghiệm hợp lệ đều có một giá trị chi phí (hoặc giá trị mục tiêu) dương. Tùy theo bài toán, mục tiêu có thể là: (i) tìm nghiệm có giá trị lớn nhất (bài toán tối đa hóa), hoặc (ii) tìm nghiệm có giá trị nhỏ nhất (bài toán tối thiểu hóa).

Định nghĩa 1.10 (TTX [38]). Cho bài toán tối ưu $\mathcal{P}$ với hàm mục tiêu f , với kích thước dữ liệu đầu vào là n , trong đó mục tiêu có thể là tìm giá trị cực đại hoặc cực tiểu. Một thuật toán $\mathcal{A}$ được gọi là TTX cho bài toán $\mathcal{P}$ với hệ số xấp xỉ $\rho(n)$ nếu:

- (i) Thuật toán $\mathcal{A}$ chạy trong thời gian đa thức theo kích thước đầu vào n ;
- (ii) Luôn trả về lời giải hợp lệ cho bài toán, gọi lời giải là S . Gọi S^* lời giải tối ưu, ta có:
 - Đối với bài toán tìm cực đại: $f(S) \geq \rho(n) \cdot f(S^*)$, ($0 < \rho \leq 1$).
 - Đối với bài toán tìm cực tiểu: $f(S) \leq \rho(n) \cdot f(S^*)$, ($\rho \geq 1$).

Một thuật toán có tỷ lệ xấp xỉ $\rho(n)$ được gọi là thuật toán $\rho(n)$ -xấp xỉ.

Nói cách khác, nghiệm do thuật toán tìm được không sai khác quá một hệ số $\rho(n)$ so với nghiệm tối ưu.

Khái niệm tỷ lệ xấp xỉ được xây dựng sao cho có thể áp dụng thống nhất cho cả hai loại bài toán tối ưu. Cụ thể:

- Với bài toán tối thiểu hóa, ta luôn có $C^* \leq C$, trong đó $C^* = f(S^*)$ là giá trị tối ưu và $C = f(S)$ là giá trị nghiệm do thuật toán trả về; khi đó tỷ số C/C^* phản ánh mức độ sai lệch của nghiệm thu được so với nghiệm tối ưu.
- Với bài toán tối đa hóa, ta có $C \leq C^*$; khi đó tỷ số C^*/C biểu thị mức độ nghiệm tối ưu lớn hơn nghiệm do thuật toán tìm được.

Nhờ cách định nghĩa này, một khuôn khổ phân tích chung có thể được sử dụng cho cả hai lớp bài toán mà không cần thay đổi bản chất của khái niệm TTX.

TTX trong CO thường có một số đặc điểm cơ bản sau.

Một tiêu chí quan trọng để phân biệt TTX với các phương pháp tìm kiếm nghiệm gần đúng khác là yêu cầu thuật toán phải chạy trong thời gian đa thức theo kích thước dữ liệu đầu vào. Điều này đảm bảo thuật toán có khả năng áp dụng trên các bài toán quy mô lớn trong thực tế. Ngược lại, các phương pháp không đảm bảo thời gian chạy đa thức thường có chi phí tính toán rất cao và khó hợp lệ khi kích thước dữ liệu tăng lên.

Trong nhiều bài toán, tỷ lệ xấp xỉ $\rho(n)$ về mặt tổng quát là một hàm số theo n . Nếu tỷ lệ xấp xỉ $\rho(n)$ không phụ thuộc vào kích thước đầu vào n mà chỉ là một hằng số, thuật toán tương ứng được gọi là *TTX hằng số* (constant-factor approximation algorithm). Tổng quát hơn, nếu một thuật toán đạt được tỷ lệ xấp xỉ $\rho(n)$ thì nó được gọi là *thuật toán $\rho(n)$ -xấp xỉ*. Các thuật toán đạt hệ số xấp xỉ hằng số thường được xem là có giá trị thực tiễn cao vì chất lượng nghiệm được đảm bảo ổn định khi quy mô dữ liệu tăng.

1.3.2. Ưu điểm và nhược điểm của thuật toán xấp xỉ

TTX mang lại nhiều lợi thế quan trọng trong việc giải các bài toán CO khó, đặc biệt khi việc tìm nghiệm tối ưu chính xác là không hợp lệ về mặt tính toán.

TTX cung cấp các bảo đảm lý thuyết rõ ràng thông qua hệ số xấp xỉ nhưng vẫn đồng thời đảm bảo khả năng triển khai hiệu quả trong thực nghiệm trên dữ liệu quy mô lớn. Hệ số xấp xỉ đóng vai trò như một tiêu chuẩn để đánh giá chất lượng nghiệm thu được, cho phép so sánh mức độ gần tối ưu của các phương pháp khác nhau một cách định lượng và có cơ sở lý thuyết. Các TTX thường có cấu trúc đơn giản, dễ mở rộng và có thể sử dụng như thành phần nền cho các phương pháp tối ưu hoặc heuristic nâng cao hơn.

Bên cạnh những ưu điểm trên, việc thiết kế và phân tích TTX cũng đặt ra nhiều thách thức.

Thứ nhất, việc chứng minh tỷ lệ xấp xỉ thường khó khăn, đòi hỏi phải khai thác sâu cấu trúc của bài toán cũng như sử dụng các kỹ thuật phân tích toán học chặt chẽ. Trong nhiều trường hợp, cần xây dựng các cận trên và cận dưới tinh vi để liên hệ nghiệm thuật toán với nghiệm tối ưu, khiến quá trình phân tích trở nên phức tạp.

Thứ hai, khi bài toán xuất hiện thêm các ràng buộc phức tạp (như knapsack, fairness, hoặc nhiều ràng buộc đồng thời), việc duy trì hệ số xấp xỉ tốt đồng thời đảm bảo hiệu quả tính toán trở nên đặc biệt thách thức.

1.3.3. Thuật toán xấp xỉ tiêu chuẩn kép

Trong nhiều bài toán CO thực tế, việc thiết kế TTXX theo tiêu chuẩn cổ điển thường gặp khó khăn do cấu trúc bài toán phức tạp hoặc do sự xuất hiện đồng thời của nhiều loại ràng buộc. Ví dụ các bài toán: người chào hàng tổng quát, bài toán tập phủ đều là những bài toán NP-khó không tồn tại TTXX hằng số [47, 48].

Trong các trường hợp này, rất khó để duy trì đồng thời cả nghiệm hợp lệ tuyệt đối và hệ số xấp xỉ tốt cho hàm mục tiêu đôi khi trở nên quá hạn chế.

Vì vậy, một hướng tiếp cận thường được sử dụng là nói lỏng một phần các yêu cầu của TTXX, cho phép nghiệm thu được có thể vi phạm nhẹ một số ràng buộc, trong khi vẫn đảm bảo chất lượng nghiệm theo nghĩa lý thuyết. Cách tiếp cận này dẫn đến khái niệm *TTXX tiêu chuẩn kép* (bicriteria approximation algorithms), trong đó thuật toán đồng thời kiểm soát hai yếu tố: chất lượng của hàm mục tiêu và mức độ vi phạm ràng buộc.

Cụ thể, thay vì yêu cầu nghiệm phải hoàn toàn thỏa mãn các ràng buộc ban đầu, thuật toán có thể cho phép nghiệm vi phạm các ràng buộc trong một hệ số cho phép, đồng thời vẫn đảm bảo giá trị mục tiêu đạt một tỷ lệ xấp xỉ nhất định so với nghiệm tối ưu. Cách tiếp cận này giúp mở rộng khả năng thiết kế thuật toán cho nhiều lớp bài toán khó, trong khi vẫn giữ được các bảo đảm lý thuyết cần thiết.

Đối với bài MCSC, TTXX tiêu chuẩn kép được định nghĩa như sau:

Định nghĩa 1.11 (TTXX tiêu chuẩn kép [49]). Xét một bài toán tối ưu với hai tiêu chí gồm hàm mục tiêu f và hàm ràng buộc g . Bài toán có dạng

$$\min_{S \in \Omega} f(S) \quad (1.16)$$

$$\text{thỏa mãn: } g(S) \geq C \quad (1.17)$$

Một thuật toán được gọi là *TTXX tiêu chuẩn kép*, hay thuật toán (a, b) -xấp xỉ, nếu trong thời gian đa thức theo kích thước dữ liệu đầu vào n , nó tìm được một nghiệm S sao cho giá trị của hàm ràng buộc chỉ vi phạm nhiều nhất một hệ số a , và đạt được tỷ lệ xấp xỉ b , tức là:

$$\begin{cases} g(S) \geq a \cdot C, \\ f(S) \leq b \cdot f(S^*), \end{cases} \quad (1.18)$$

với $0 < a < 1, b > 1$ và S^* là nghiệm tối ưu.

TTXX tiêu chuẩn kép có thể được xem như một phiên bản nói lỏng của TTXX truyền thống trong các bài toán CO có ràng buộc. Thay vì yêu cầu nghiệm tìm được phải

hoàn toàn thỏa mãn mọi ràng buộc của bài toán, cách tiếp cận này cho phép nghiệm có thể vi phạm ràng buộc ở một mức độ kiểm soát được, đồng thời vẫn đảm bảo một chặn lý thuyết rõ ràng đối với chất lượng của hàm mục tiêu so với nghiệm tối ưu.

Do đó, lời giải thu được có thể không hoàn toàn hợp lệ theo nghĩa chặt chẽ, nhưng mức độ vi phạm ràng buộc và độ suy giảm chất lượng nghiệm đều được khống chế bởi các hệ số xấp xỉ xác định trước. Quan trọng hơn, phương pháp này cho phép điều chỉnh sự đánh đổi giữa chất lượng hàm mục tiêu và mức độ vi phạm ràng buộc thông qua việc lựa chọn các tham số xấp xỉ phù hợp với từng ứng dụng cụ thể.

Trong thực nghiệm, các TTX tiêu chuẩn kép thường cho nghiệm có chất lượng tốt, mức vi phạm ràng buộc nhỏ và hiệu quả tính toán cao, do đó trở thành một hướng tiếp cận thực tế và hữu ích cho nhiều bài toán quy mô lớn hiện nay.

1.4. MỘT SỐ PHƯƠNG PHÁP THIẾT KẾ TTX HIỆN ĐẠI

Trong phần này, luận án giới thiệu một số cách tiếp cận thiết kế TTX hiện đại dành cho các bài toán tối ưu quy mô lớn, trong đó mục tiêu là tìm được nghiệm có chất lượng được đảm bảo bởi các chặn lý thuyết so với nghiệm tối ưu, đồng thời duy trì độ phức tạp tính toán và lượng dữ liệu lưu trữ ở mức hợp lệ trong thực tế.

Khi quy mô dữ liệu tiếp tục gia tăng và độ phức tạp của bài toán ngày càng lớn, việc tìm nghiệm tối ưu chính xác thường trở nên không thực tế do chi phí tính toán quá cao. Trong bối cảnh đó, các phương pháp xấp xỉ hiện đại hướng đến việc dung hòa giữa chất lượng nghiệm, thời gian xử lý và yêu cầu về bộ nhớ, qua đó cho phép triển khai hiệu quả trên các hệ thống dữ liệu lớn cũng như trong những môi trường tính toán có hạn chế về tài nguyên. Phần tiếp theo sẽ giới thiệu một số hướng tiếp cận tiêu biểu đang được áp dụng rộng rãi trong cả nghiên cứu và thực tiễn.

1.4.1. Thuật toán tham lam

Thuật toán tham lam là một phương pháp kinh điển trong thiết kế các TTX, nổi bật nhờ cấu trúc đơn giản nhưng vẫn đạt hiệu quả tốt cả về mặt lý thuyết lẫn thực nghiệm. Ý tưởng cơ bản của phương pháp này là xây dựng lời giải một cách tuần tự qua nhiều bước; tại mỗi bước, thuật toán lựa chọn phần tử hoặc hành động được xem là tốt nhất theo một tiêu chí đánh giá cục bộ, thường dựa trên mức cải thiện của hàm mục tiêu hoặc mức độ phù hợp với các ràng buộc hiện có.

Mặc dù quyết định tại mỗi bước chỉ dựa trên thông tin cục bộ, trong nhiều lớp bài toán quan trọng, đặc biệt là các bài toán liên quan đến hàm submodular hoặc các bài toán bao phủ, phương pháp tham lam vẫn đạt được các hệ số xấp xỉ mạnh với chi phí tính toán thấp. Bên cạnh đó, tư tưởng tham lam còn đóng vai trò nền tảng cho nhiều kỹ thuật hiện

đại hơn, chẳng hạn các thuật toán song song, thuật toán xử lý dữ liệu dạng luồng, hay các biến thể ngẫu nhiên và tham lam ngưỡng, qua đó cho phép mở rộng phương pháp này để xử lý các bài toán quy mô rất lớn trong thực tế. Thuật toán tham lam cơ bản (thuật toán 1.1) được mô tả như sau:

Thuật toán 1.1: Thuật toán tham lam [38, 47]

Input: Tập phần tử V , hàm mục tiêu f , tập ràng buộc hợp lệ

Output: Lời giải S

```

1 Khởi tạo nghiệm rỗng:  $S \leftarrow \emptyset$ ;
2 while điều kiện dừng chưa thoả mãn do
3     Xác định tập các phần tử  $C \subseteq V \setminus S$  thoả mãn các ràng buộc
4     Chọn phần tử tốt nhất  $c \in C$  theo tiêu chí tham lam (dựa trên mức cải thiện
       của  $f$  hoặc một tiêu chí đánh giá khác);
5     if việc thêm  $c$  vào  $S$  vẫn thoả mãn ràng buộc then
6          $S \leftarrow S \cup \{c\}$ ;
7     end
8     Loại bỏ  $c$  khỏi tập ứng viên:  $C \leftarrow C \setminus \{c\}$ ;
9 end
10 return  $S$ 

```

Phương pháp tham lam sở hữu nhiều ưu điểm khiến nó trở thành một trong những công cụ phổ biến nhất trong thiết kế TTX. Trước hết, cấu trúc của thuật toán thường đơn giản, dễ cài đặt và dễ mở rộng cho các bài toán khác nhau. Mỗi bước chỉ cần đánh giá lợi ích cục bộ của việc thêm một phần tử vào lời giải hiện tại, do đó chi phí tính toán thường thấp hơn đáng kể so với các phương pháp tối ưu chính xác. Ngoài ra, trong nhiều lớp bài toán quan trọng như tối đa hóa hàm submodular, bài toán bao phủ hay các bài toán lựa chọn tập con dưới ràng buộc kích thước, thuật toán tham lam đạt được các tỷ lệ xấp xỉ mạnh với đảm bảo lý thuyết rõ ràng. Một ưu điểm khác là phương pháp này có thể được điều chỉnh để hoạt động hiệu quả trong các môi trường dữ liệu lớn, chẳng hạn như các biến thể song song, phân tán hoặc xử lý dữ liệu dạng luồng.

Tuy nhiên, thuật toán tham lam cũng tồn tại một số hạn chế. Do mỗi quyết định được đưa ra dựa trên thông tin cục bộ tại từng bước, lời giải cuối cùng không phải lúc nào cũng gần với nghiệm tối ưu trong các bài toán tổng quát, đặc biệt khi hàm mục tiêu không có cấu trúc thuận lợi như tính submodular. Ngoài ra, các quyết định được thực hiện ở giai đoạn đầu có thể làm mất đi cơ hội xây dựng những lời giải tốt hơn ở các bước sau, do thuật toán thường không có cơ chế quay lui hay điều chỉnh lại lựa chọn trước đó.

Trong các bài toán có nhiều ràng buộc phức tạp hoặc cấu trúc tương tác mạnh giữa các phần tử, hiệu quả của phương pháp tham lam có thể suy giảm đáng kể.

Trong nhiều trường hợp, thuật toán tham lam được sử dụng như một thành phần nền tảng hoặc một bước khởi tạo cho các phương pháp nâng cao hơn. Các hướng tiếp cận này thường kết hợp thêm những kỹ thuật như ngẫu nhiên hóa, tối ưu cục bộ hoặc các quy trình nhiều giai đoạn nhằm cải thiện chất lượng lời giải.

1.4.2. Thuật toán luồng

Trong mô hình dữ liệu luồng (data streaming), dữ liệu đầu vào không được cung cấp toàn bộ ngay từ đầu mà xuất hiện dần theo thời gian dưới dạng một chuỗi các cập nhật liên tiếp. Cụ thể, ta có thể xem dữ liệu như một tín hiệu một chiều $A : [1..n] \rightarrow \mathbb{R}$, trong đó trạng thái của A được hình thành thông qua một chuỗi các phép cập nhật

$$a_1, a_2, \dots$$

đến theo thứ tự thời gian. Mỗi phép cập nhật làm thay đổi giá trị tại một vị trí nào đó của tín hiệu A .

Tùy theo cách diễn giải các phép cập nhật, người ta thường xét ba mô hình luồng dữ liệu tiêu biểu sau:

Mô hình chuỗi thời gian (time series). Mỗi phần tử của luồng chính là một giá trị được ghi nhận theo thời gian, tức là $A[i]$ lưu trữ giá trị quan sát được tại thời điểm i .

Mô hình máy tính tiền (cash register). Mỗi cập nhật có dạng $a_t = (j, \Delta)$, biểu diễn việc tăng giá trị tại vị trí j lên một lượng không âm Δ , tức là

$$A[j] \leftarrow A[j] + \Delta.$$

Mô hình turnstile. Mỗi cập nhật có dạng $a_t = (j, \Delta)$ với Δ có thể dương hoặc âm, cho phép cả việc tăng và giảm giá trị tại một vị trí. Trong biến thể *strict turnstile*, ta luôn yêu cầu $A[j] \geq 0$ tại mọi thời điểm.

Ba mô hình trên tạo thành một hệ phân cấp tự nhiên, trong đó mô hình turnstile là tổng quát nhất và bao hàm các mô hình còn lại như những trường hợp riêng.

Dưới góc độ thiết kế thuật toán, thuật toán luồng được thống nhất và định nghĩa như sau:

Định nghĩa 1.12 (Thuật toán luồng [50]). Thuật toán luồng là một TTX xử lý dữ liệu đầu vào dưới dạng một chuỗi các cập nhật liên tiếp, trong đó mỗi phần tử dữ liệu thường chỉ có thể được truy cập một hoặc một số rất ít lần. Do hạn chế về khả năng lưu trữ toàn bộ dữ liệu, thuật toán chỉ duy trì một trạng thái tóm tắt có kích thước nhỏ để phục vụ cho việc tính toán.

Hiệu quả của một thuật toán luồng thường được đánh giá dựa trên các tiêu chí sau:

- *Dung lượng dữ liệu cần lưu trữ (space complexity)*: Biểu thị lượng bộ nhớ cần thiết để duy trì trạng thái trung gian của thuật toán. Trong nhiều thiết kế, mục tiêu là giữ dung lượng bộ nhớ ở mức dưới tuyến tính theo kích thước dữ liệu đầu vào.
- *Số lượt quét dữ liệu (number of scans/passes)*: Thể hiện số lần thuật toán cần duyệt qua toàn bộ luồng dữ liệu. Các thuật toán hiệu quả thường chỉ yêu cầu một hoặc rất ít lần quét dữ liệu.
- *Độ phức tạp truy vấn*: Đo số lượng truy vấn tới hàm mục tiêu hoặc cấu trúc dữ liệu được duy trì trong quá trình xử lý, đặc biệt quan trọng khi các truy vấn này có chi phí cao.
- *Đảm bảo tỷ lệ xấp xỉ*: Do không thể lưu trữ toàn bộ dữ liệu, thuật toán luồng thường chỉ tìm được nghiệm gần đúng. Vì vậy, một tiêu chí quan trọng là đảm bảo nghiệm thu được đạt một tỷ lệ xấp xỉ nhất định so với nghiệm tối ưu.

1.4.3. Thuật toán song song

Trong bối cảnh thiết kế thuật toán hiện đại, các thuật toán song song (parallel algorithms) đóng vai trò then chốt, đặc biệt đối với các bài toán xử lý dữ liệu quy mô lớn trên những hệ thống đa lõi hoặc hiệu năng cao. Khi các phương pháp tuần tự dần bộc lộ những hạn chế về thời gian xử lý và tài nguyên tính toán, việc tận dụng khả năng thực thi đồng thời trên nhiều bộ xử lý trở thành một hướng tiếp cận hiệu quả nhằm giảm thời gian chạy và nâng cao khả năng mở rộng của hệ thống.

Trong lĩnh vực này, mô hình PRAM (Parallel Random Access Machine) thường được sử dụng như một khuôn khổ lý thuyết cơ bản, cho phép nhiều bộ xử lý truy cập chung vào một không gian bộ nhớ. Dựa trên cơ chế điều phối truy cập đồng thời, mô hình PRAM được phân thành các biến thể như EREW (Exclusive Read Exclusive Write), CREW (Concurrent Read Exclusive Write) và CRCW (Concurrent Read Concurrent Write).

Ngoài mô hình bộ nhớ chia sẻ, các hệ thống thực tế còn áp dụng kiến trúc bộ nhớ phân tán, trong đó dữ liệu được phân bố trên nhiều nút tính toán và việc trao đổi thông tin giữa các nút được thực hiện thông qua mạng truyền thông. Trong quá trình thiết kế thuật toán song song, một số kỹ thuật cơ bản thường được sử dụng:

- *Chia để trị song song (parallel divide-and-conquer)* [51] Bài toán được phân tách thành các bài toán con độc lập có thể xử lý đồng thời, sau đó kết hợp kết quả để thu được

lời giải cuối cùng. Phương pháp này được áp dụng rộng rãi trong sắp xếp, xử lý ma trận và nhiều bài toán số học lớn.

- *Kỹ thuật nhảy con trỏ (Pointer jumping)*. Kỹ thuật này cho phép rút ngắn chiều sâu của các cấu trúc dữ liệu dạng danh sách liên kết hoặc cây bằng cách cập nhật con trỏ song song, qua đó giảm số bước tính toán cần thiết.

- *Kỹ thuật Euler Tour [52]* Thông qua việc biến đổi cấu trúc cây thành một biểu diễn tuyến tính dựa trên chu trình Euler, nhiều phép toán trên cây như tính độ sâu, xác định tổ tiên chung gần nhất hay truy vấn đường đi có thể được thực hiện hiệu quả trong môi trường song song.

- *Tính toán tiền tố (Prefix computation) [53]* Kỹ thuật này thực hiện phép tích lũy trên mảng dữ liệu đầu vào và là một thành phần cơ bản trong nhiều thuật toán song song, đặc biệt trong xử lý dữ liệu trên GPU và CPU đa lõi.

Xét một thuật toán song song xử lý dữ liệu đầu vào kích thước n . Độ phức tạp song song của thuật toán thường được mô tả thông qua hai đại lượng cơ bản:

- *Tổng công việc (Work) $W(n)$* . Là tổng số phép toán được thực hiện bởi toàn bộ các bộ xử lý trong quá trình thực thi thuật toán. Đại lượng này tương đương với độ phức tạp của thuật toán khi thực hiện trong môi trường tuần tự.
- *Chiều sâu tính toán (Depth) $D(n)$* . Là số bước tính toán tuần tự tối thiểu cần thực hiện khi giả sử có số lượng bộ xử lý không giới hạn. Đại lượng này còn được gọi là thời gian chạy song song lý tưởng.

Khi sử dụng P bộ xử lý, thời gian thực thi song song $T_P(n)$ thường được ước lượng bởi công thức Brent:

$$T_P(n) = O\left(\frac{W(n)}{P} + D(n)\right).$$

Một thuật toán song song được xem là hiệu quả nếu tổng công việc $W(n)$ không lớn hơn đáng kể so với thuật toán tuần tự tốt nhất, đồng thời chiều sâu $D(n)$ đủ nhỏ để khai thác hiệu quả khả năng thực thi đồng thời của hệ thống tính toán.

Theo định luật Brent, nếu sử dụng P bộ xử lý thì thời gian thực thi song song có thể được ước lượng bởi

$$T_P = O\left(\frac{W}{P} + D\right),$$

cho thấy sự đánh đổi giữa tổng lượng công việc và mức độ song song hóa.

Ngoài hai đại lượng trên, các yếu tố như tốc độ tăng tốc (speedup), khả năng mở rộng (scalability), cân bằng tải (load balancing) và chi phí truyền thông giữa các bộ xử

lý cũng đóng vai trò quan trọng trong việc đánh giá hiệu quả thực tế của thuật toán song song.

Trong các bài toán CO quy mô lớn và phân tích dữ liệu, nhiều bài toán quan trọng như liệt kê tập con, tối đa hóa hàm submodular, phân cụm hay matching có thể được khai thác theo hướng song song, toàn phần hoặc từng phần. Các nghiên cứu gần đây cho thấy rằng những thuật toán tham lam và các biến thể của chúng vẫn có thể duy trì tỷ lệ xấp xỉ tương đương với phiên bản tuần tự, đồng thời giảm đáng kể độ sâu tính toán nhờ áp dụng các kỹ thuật như chọn mẫu hoặc xử lý độc lập trên từng khối dữ liệu.

Kể từ khoảng năm 2018, các mô hình tính toán song song ngày càng được ứng dụng rộng rãi trong các bài toán học máy và trí tuệ nhân tạo. Trong các hệ thống này, nhiều phép truy vấn tới hàm mục tiêu f có thể được thực hiện đồng thời và độc lập trên các luồng xử lý song song. Dựa trên mô hình tính toán này, Balkanski và các cộng sự lần đầu tiên đưa ra khái niệm *độ phức tạp song song* (adaptive complexity) [34], và từ đó khái niệm này đã trở thành một công cụ quan trọng trong việc thiết kế các thuật toán tối ưu hiện đại cho dữ liệu quy mô lớn [54, 30, 32, 31].

Định nghĩa 1.13 (Độ phức tạp song song [34]). Với một hàm mục tiêu f , độ phức tạp song song của một thuật toán là số vòng lặp tối thiểu cần thiết sao cho trong mỗi vòng lặp, thuật toán thực hiện $O(\text{poly}(n))$ truy vấn độc lập đến hàm mục tiêu, trong đó n là kích thước dữ liệu đầu vào của thuật toán.

Một đặc điểm phổ biến của nhiều thuật toán tối ưu là tính chất tuần tự, trong đó mỗi bước thực hiện thường phụ thuộc vào kết quả của các bước trước đó. Điều này làm hạn chế khả năng khai thác tính toán song song một cách tự nhiên, vì các phép tính không thể thực hiện đồng thời khi chưa có kết quả trung gian cần thiết.

Khái niệm độ phức tạp song song khắc phục hạn chế này bằng cách đưa ra một tiêu chuẩn đo lường số lượng vòng lặp bắt buộc phải thực hiện tuần tự trong thuật toán. Theo đó, những thuật toán có độ phức tạp song song càng nhỏ thì càng cho phép thực hiện nhiều phép tính độc lập trong cùng một thời điểm, từ đó khai thác hiệu quả hơn các hệ thống tính toán song song hiện đại. Tiêu chuẩn này vì vậy mang ý nghĩa thực tiễn cao trong việc thiết kế các thuật toán tối ưu cho dữ liệu quy mô lớn.

Ví dụ 1.1. Xét bài toán tối đa hóa một hàm submodular $f : 2^V \rightarrow \mathbb{R}_+$ dưới ràng buộc $|S| \leq k$, trong đó

$$V = \{e_1, e_2, \dots, e_n\}.$$

Thuật toán tham lam khởi tạo $S_0 = \emptyset$. Tại vòng thứ t , với $1 \leq t \leq k$, thuật toán cần tìm

phần tử

$$e_{\max} \in \arg \max_{e \in V \setminus S_{t-1}} \Delta(e \mid S_{t-1}),$$

trong đó

$$\Delta(e \mid S_{t-1}) = f(S_{t-1} \cup \{e\}) - f(S_{t-1})$$

là lợi ích biên của phần tử e đối với tập hiện tại S_{t-1} .

Để xác định $e_{\max}$, thuật toán thực hiện đồng thời các truy vấn

$$f(S_{t-1} \cup \{e\}), \quad e \in V \setminus S_{t-1}.$$

Do tất cả các truy vấn này chỉ phụ thuộc vào tập S_{t-1} và không phụ thuộc lẫn nhau, chúng có thể được thực hiện song song trong cùng một vòng. Sau khi nhận được kết quả của các truy vấn, thuật toán đồng thời tính các giá trị $\Delta(e \mid S_{t-1})$, xác định phần tử $e_{\max}$ có lợi ích biên lớn nhất và cập nhật

$$S_t = S_{t-1} \cup \{e_{\max}\}.$$

Như vậy, toàn bộ quá trình xác định và lựa chọn $e_{\max}$ được thực hiện trong một vòng song song. Tuy nhiên, các truy vấn tại vòng thứ t phụ thuộc vào tập S_{t-1} thu được từ vòng trước. Vì vậy, vòng thứ t chỉ có thể bắt đầu sau khi vòng thứ $t - 1$ kết thúc. Thuật toán cần thực hiện tuần tự k vòng để lựa chọn k phần tử. Do đó, độ phức tạp song song của thuật toán là $O(k)$, trong khi tổng số truy vấn đến hàm mục tiêu là $O(nk)$.

Chẳng hạn, với $n = 1000$ và $k = 5$, quá trình thực hiện của thuật toán được minh họa trong Bảng 1.2.

Bảng 1.2: Minh họa các vòng truy vấn song song với $n = 1000$ và $k = 5$.

Vòng lặp	Tập hiện tại	Số truy vấn đồng thời	Phần tử được chọn
1	$S_0 = \emptyset$	1000	$e_{\max}^{(1)}$
2	$S_1 = \{e_{\max}^{(1)}\}$	999	$e_{\max}^{(2)}$
3	$S_2 = \{e_{\max}^{(1)}, e_{\max}^{(2)}\}$	998	$e_{\max}^{(3)}$
4	$S_3 = \{e_{\max}^{(1)}, e_{\max}^{(2)}, e_{\max}^{(3)}\}$	997	$e_{\max}^{(4)}$
5	$S_4 = \{e_{\max}^{(1)}, \dots, e_{\max}^{(4)}\}$	996	$e_{\max}^{(5)}$
Tổng số truy vấn		4990	

Trong mỗi vòng, tất cả các truy vấn được thực hiện đồng thời để xác định $e_{\max}^{(t)}$. Tuy nhiên, vòng thứ t phải chờ kết quả của vòng thứ $t - 1$ để xác định tập S_{t-1} . Do đó, mặc dù thuật toán thực hiện tổng cộng 4990 truy vấn, các truy vấn này chỉ được tổ chức thành 5 vòng phụ thuộc tuần tự. Vì vậy, độ phức tạp song song của thuật toán là 5.

Trong luận án này, các thực nghiệm được xây dựng bám sát định nghĩa về độ phức tạp song song và Ví dụ 1.1. Trong đó mỗi thuật toán được tổ chức thành các vòng song song. Tại mỗi vòng, thuật toán xây dựng một tập gồm $O(\text{poly}(n))$ truy vấn độc lập và thực hiện đồng thời các truy vấn này đến hàm mục tiêu, như được mô tả cụ thể trong giả mã thuật toán. Kết quả của vòng hiện tại được sử dụng để xây dựng các truy vấn cho vòng tiếp theo. Do đó, nếu thuật toán cần r vòng phụ thuộc tuần tự thì độ phức tạp song song của thuật toán là $O(r)$.

1.5. TỔNG QUAN VỀ BÀI TOÁN TỐI ƯU SUBMODULAR

Như đã đề cập, các bài toán tối ưu submodular (SO) thường được chia thành hai nhóm chính là tối đa hóa hàm submodular (SM) và phủ submodular với chi phí tối thiểu (MCSC). Trong mục này, LA trình bày tổng quan các kết quả nghiên cứu tiêu biểu đối với hai lớp bài toán nền tảng này, tập trung vào các hướng tiếp cận chính, các kết quả lý thuyết đã đạt được, cũng như những thách thức và vấn đề nghiên cứu còn bỏ ngỏ.

1.5.1. Bài toán tối đa hàm submodular

George L. Nemhauser và các cộng sự [14] là những người đầu tiên nghiên cứu một cách hệ thống bài toán SM dưới ràng buộc lực lượng (SMC). Trong công trình kinh điển này, nhóm tác giả đã đề xuất thuật toán tham lam tuần tự, đạt tỷ lệ xấp xỉ chặt $1 - \frac{1}{e}$ thông qua việc lựa chọn liên tiếp phần tử có độ lợi biên lớn nhất. Thuật toán có độ phức tạp truy vấn $O(nk)$, trong đó n là kích thước tập nền và k là giới hạn lực lượng. Kết quả này được xem là nền tảng của lĩnh vực SO và đã thúc đẩy sự phát triển của nhiều hướng nghiên cứu nhằm cải thiện hiệu quả tính toán của các TTX cho bài toán SM.

Một trong những hạn chế của thuật toán tham lam là tính tuần tự cao, dẫn đến thời gian thực thi lớn khi kích thước dữ liệu tăng. Để khắc phục nhược điểm này, nhiều nghiên cứu đã tập trung vào việc giảm độ phức tạp truy vấn và thời gian chạy. Công trình [24] đề xuất thuật toán xác định dựa trên mô hình streaming, giảm thời gian thực thi xuống còn $O(n \log n)$. Gần đây, [55] và [18] đã đồng thời phát triển các thuật toán thời gian tuyến tính, đạt tỷ lệ xấp xỉ tiệm cận $1 - \frac{1}{e}$, mở ra hướng tiếp cận hiệu quả cho các bài toán quy mô lớn.

Song song với việc giảm độ phức tạp tính toán, một hướng nghiên cứu khác là thiết kế các thuật toán có khả năng song song hóa. Công trình tiên phong của [34] đã áp dụng kỹ thuật lấy mẫu thích nghi (adaptive sampling) cho bài toán SM, đạt tỷ lệ xấp xỉ hằng số cùng độ phức tạp thích nghi $O(\log n)$. Ý tưởng chính là loại bỏ một phần cố định các phần tử ở mỗi vòng lặp nhằm giảm số bước tuần tự. Sau đó, [56] tiếp tục cải thiện hướng tiếp cận này, đạt tỷ lệ xấp xỉ gần $1 - \frac{1}{e}$ với $O(\log n \log k)$ vòng thích nghi và $O(n \log \log k)$

truy vấn. Những kết quả này cho thấy khả năng giải quyết hiệu quả các bài toán SO trên các hệ thống tính toán song song và dữ liệu quy mô lớn.

Bên cạnh ràng buộc lực lượng, bài toán SM còn được mở rộng sang trường hợp ràng buộc chi phí (SMK), trong đó mỗi phần tử được gán một chi phí và tổng chi phí lựa chọn không vượt quá một ngân sách cho trước. Laurence A. Wolsey [9] là người đầu tiên nghiên cứu bài toán này, chứng minh tính NP-khó và đề xuất thuật toán đạt tỷ lệ xấp xỉ $1 - \frac{1}{e}$. Tiếp theo, các công trình [57, 58] phát triển các thuật toán tham lam kết hợp kỹ thuật đoán trước một số phần tử, đạt tỷ lệ xấp xỉ tối ưu nhưng phải đánh đổi bằng độ phức tạp truy vấn rất lớn.

Các nghiên cứu gần đây tập trung vào việc duy trì tỷ lệ xấp xỉ tốt đồng thời giảm chi phí tính toán. Công trình [59] giảm số phép đoán và hạ độ phức tạp xuống $O(n^3 \log n)$ trong khi vẫn giữ tỷ lệ $1 - \frac{1}{e}$. Sau đó, [60] sử dụng kỹ thuật mở rộng đa tuyến (multilinear extension) để đạt tỷ lệ xấp xỉ gần tối ưu, mặc dù chi phí tính toán vẫn còn rất cao. Trong bối cảnh dữ liệu lớn, [18] đã đề xuất thuật toán thời gian tuyến tính với tỷ lệ xấp xỉ $1/2$, đồng thời chỉ ra các giới hạn cơ bản về độ phức tạp truy vấn của bài toán.

Bên cạnh việc tập trung giảm độ phức tạp tính toán, bài toán SM còn được nghiên cứu trong xử lý dữ liệu online như thuật toán luồng. Trong mô hình này, dữ liệu liên tục với bộ nhớ hạn chế. Các công trình [61, 62] đạt tỷ lệ xấp xỉ $1/(2.5 + \epsilon)$ với một hoặc hai lượt quét dữ liệu, trong khi [63] cải thiện chất lượng nghiệm thông qua việc sử dụng nhiều lượt quét hơn. Đáng chú ý, [18] đã đề xuất thuật toán streaming với số lượng truy vấn gần tuyến tính, phù hợp cho các ứng dụng dữ liệu quy mô rất lớn.

Ngoài ràng buộc một ngân sách, nhiều nghiên cứu cũng xem xét trường hợp tồn tại đồng thời nhiều ràng buộc chi phí. Tuy nhiên, các kết quả hiện nay thường đạt tỷ lệ xấp xỉ phụ thuộc tuyến tính vào số lượng ràng buộc, dẫn đến chất lượng nghiệm suy giảm khi số ràng buộc tăng. Đặc biệt, bài toán tối đa hóa hàm submodular với các ngân sách riêng cho từng nhóm đối tượng vẫn còn là một hướng nghiên cứu mở và chưa có nhiều kết quả lý thuyết hiệu quả.

1.5.2. Bài toán phủ submodular với chi phí tối thiểu

Song song với bài toán SM, bài toán MCSC được xem là bài toán đối ngẫu tự nhiên và có nhiều ứng dụng trong lan truyền ảnh hưởng, triển khai cảm biến, thu thập dữ liệu và phân bổ tài nguyên. Thay vì tối đa hóa lợi ích dưới một ngân sách cho trước, MCSC tìm nghiệm có chi phí nhỏ nhất để đạt được một ngưỡng lợi ích mong muốn.

Bài toán MCSC đã được nghiên cứu từ rất sớm thông qua các thuật toán tham lam và các phương pháp xấp xỉ [42, 64, 16, 65, 66]. Về mặt lý thuyết, Feige [67] đã chứng

minh MCSC là NP-khó và không thể đạt tỷ lệ xấp xỉ hằng số trong thời gian đa thức. Các thuật toán tham lam của Wolsey [42] và các nghiên cứu tiếp theo [64, 16, 65] đạt được các bảo đảm lý thuyết tốt nhưng có tính tuần tự cao, gây khó khăn khi xử lý dữ liệu lớn. Gần đây, nhiều nghiên cứu tập trung vào việc giảm độ phức tạp truy vấn và tăng khả năng song song hóa. Công trình [66] đề xuất thuật toán tiền hóa với bảo đảm xấp xỉ tiêu chuẩn kép, trong khi [19] phát triển thuật toán song song đầu tiên cho MCSC với độ phức tạp thích nghi đa thức theo logarit. Tuy nhiên, các phương pháp hiện nay vẫn còn hạn chế về khả năng mở rộng hoặc chỉ áp dụng cho các hàm nhận giá trị nguyên.

Song song với MCSC, tối ưu hàm k -submodular cũng thu hút nhiều sự quan tâm. Kể từ các công trình nền tảng về tối đa hóa hàm k -submodular không ràng buộc [68, 69], nhiều kết quả đã được phát triển cho các ràng buộc lực lượng, matroid, chi phí và mô hình dữ liệu luồng [11, 70, 71, 72, 73]. Tuy nhiên, phần lớn các nghiên cứu tập trung vào bài toán tối đa hóa.

Đối với bài toán phủ k -submodular (kSC), các kết quả hiện nay còn khá hạn chế. Công trình [74] là nghiên cứu đầu tiên xem xét bài toán này, trong khi [75] mở rộng sang mô hình dữ liệu luồng với các bảo đảm xấp xỉ tiêu chuẩn kép. Mặc dù vậy, việc thiết kế các thuật toán đa thức có tỷ lệ xấp xỉ tốt, độ phức tạp thấp và khả năng mở rộng trên dữ liệu lớn vẫn là một thách thức mở.

1.5.3. Các thách thức trong nghiên cứu bài toán SO

Từ tổng quan trên, có thể nhận thấy một số hướng nghiên cứu quan trọng cần tiếp tục được quan tâm như sau:

- *Phát triển các thuật toán hiệu quả cho dữ liệu quy mô lớn.* Mặc dù đã có nhiều TTXX cho các bài toán SM, MCSC và các biến thể, phần lớn vẫn đòi hỏi số lượng truy vấn lớn hoặc có độ phức tạp thích nghi cao. Do đó, việc thiết kế các thuật toán có độ phức tạp truy vấn thấp, khả năng song song hóa tốt và phù hợp với dữ liệu quy mô rất lớn vẫn là một thách thức quan trọng.
- *Nghiên cứu các mô hình xử lý dữ liệu hiện đại.* Trong nhiều ứng dụng thực tế như OSN, hệ thống gợi ý và phân tích dữ liệu trực tuyến, dữ liệu thường xuất hiện dưới dạng luồng hoặc thay đổi liên tục theo thời gian. Điều này đặt ra nhu cầu phát triển các thuật toán SO trong các mô hình luồng, song song hóa và phân tán.
- *Mở rộng sang các biến thể mới của bài toán.* Các bài toán thực tế ngày càng đòi hỏi các mô hình tối ưu phản ánh tốt hơn các yêu cầu ứng dụng như nhiều ràng buộc chi phí, công bằng giữa các nhóm đối tượng, đa cộng đồng hoặc dữ liệu động. Việc nghiên cứu các biến thể này không chỉ có ý nghĩa thực tiễn mà còn mở ra nhiều bài toán lý

thuyết mới.

- *Thu hẹp khoảng cách giữa lý thuyết và thực nghiệm.* Một hướng nghiên cứu quan trọng là xây dựng các thuật toán vừa có bảo đảm xấp xỉ chặt chẽ, vừa đạt hiệu năng cao trên các bộ dữ liệu thực tế, từ đó nâng cao khả năng ứng dụng của các phương pháp SO trong khoa học dữ liệu và học máy.

1.6. KẾT LUẬN CHƯƠNG

Chương này đã trình bày tổng quan về CO và SO, trong đó tập trung vào hai lớp bài toán nền tảng là SM và MCSC. Bên cạnh đó, chương cũng giới thiệu các khái niệm cơ bản về thiết kế thuật toán, độ phức tạp tính toán và các thước đo đánh giá hiệu năng thuật toán.

Trên cơ sở đó, chương đã tổng hợp và phân tích các công trình nghiên cứu tiêu biểu liên quan đến SM, MCSC và các biến thể mở rộng của chúng. Từ những kết quả đã đạt được cũng như các hạn chế còn tồn tại, chương đã chỉ ra các hướng nghiên cứu tiềm năng, qua đó tạo nền tảng lý thuyết và định hướng cho các nội dung nghiên cứu được trình bày trong các chương tiếp theo của luận án.

CHƯƠNG 2

BÀI TOÁN PHỦ SUBMODULAR VỚI CHI PHÍ TỐI THIỂU

Bài toán Phủ Submodular với chi phí tối thiểu (MCSC) xuất hiện trong nhiều ứng dụng quan trọng của nghiên cứu tác nghiệp và trí tuệ nhân tạo, chẳng hạn như học chủ động (active learning) [76, 39], khai phá dữ liệu [77, 78], tối ưu phúc lợi xã hội [79], cũng như các bài toán lan truyền ảnh hưởng trong OSN [1, 80, 81]. Trong các bối cảnh này, mục tiêu thường là lựa chọn một tập phần tử có tổng chi phí nhỏ nhất nhưng vẫn đạt được mức lợi ích hoặc độ bao phủ yêu cầu, qua đó phản ánh bài toán cân bằng giữa hiệu quả và chi phí trong thực tế.

Xuất phát từ cả giá trị lý thuyết và ý nghĩa ứng dụng của bài toán, chương này tập trung nghiên cứu bài toán Phủ Submodular với chi phí tối thiểu theo hai định hướng chính: nâng cao chất lượng lời giải và tối ưu chi phí tính toán, đặc biệt trong bối cảnh dữ liệu quy mô lớn. Nội dung chương trình bày các thuật toán được đề xuất, đi kèm với phân tích lý thuyết và đánh giá thực nghiệm, qua đó làm rõ khả năng rút ngắn khoảng cách giữa các kết quả lý thuyết và hiệu quả triển khai trong thực tế.

Cụ thể, LA đề xuất một thuật toán song song hiệu quả, đạt tỷ lệ xấp xỉ tiệm cận với phương pháp tốt nhất hiện nay trong [19], đồng thời giảm đáng kể độ phức tạp song song cả về mặt lý thuyết và thực nghiệm. Các kết quả liên quan đã được công bố trên tạp chí chuyên ngành về tối ưu hóa và vận trù học **Asia-Pacific Journal of Operational Research (SCIE/Q3)** năm 2024, cũng như trong Kỷ yếu Hội thảo Quốc gia lần thứ XXVII về một số vấn đề chọn lọc của Công nghệ thông tin và Truyền thông năm 2024.

2.1. PHÁT BIỂU BÀI TOÁN

Mục này, LA trình bày và phân tích cụ thể hơn về bài toán MCSC (Dạng 1 của bài toán SO). Bài toán SO được phát biểu như sau:

Input:

- Một tập nền hữu hạn $V = \{e_1, e_2, \dots, e_n\}$, gồm n phần, mỗi phần tử $e_i \in V$ có một chi phí $c(e_i) > 0$.
- Với tập con $S \subseteq V$, hàm mục tiêu là tổng chi phí $c(S) = f(S) = \sum_{e \in S} c(e)$.
- Hàm ràng buộc $g : 2^V \mapsto \mathbb{R}_+$ là submodular, đánh giá giá trị của tập con S .
- Ngưỡng giá trị T .

Output:

- Một tập con $S \subseteq V$ sao cho $g(S) \geq T$ với $c(S)$ là nhỏ nhất, tức là tìm:

$$S = \arg \min_{S \subseteq V: g(S) \geq T} c(S). \quad (2.1)$$

Bài toán MCSC có giá trị ứng dụng rất lớn trong thực tiễn, bởi trong nhiều bối cảnh, mục tiêu đặt ra không phải là tối đa hóa giá trị tuyệt đối, mà là tìm nghiệm có chi phí nhỏ nhất sao cho đạt được một mức chất lượng chấp nhận được. Ứng dụng điển hình của MCSC là bài toán Bài toán ảnh hưởng ngưỡng (IT) trong mạng OSN như sau:

Bài toán TI (Threshold Influence-TI) là một bài toán quan trọng trong lan truyền ảnh hưởng trên OSN, trong đó mục tiêu là lan truyền thông tin sao cho mức độ ảnh hưởng đạt đến hoặc vượt qua một ngưỡng xác định cho trước (quy mô). Mô hình này phù hợp với nhiều kịch bản thực tế, chẳng hạn như chiến dịch tiếp thị cần đạt được một mức độ nhận biết tối thiểu, hoặc các ứng dụng lan truyền thông tin đòi hỏi một tỷ lệ chấp nhận đủ lớn trong mạng.

Xét một đồ thị có hướng $G = (V, E)$ biểu diễn OSN, cùng với một mô hình lan truyền thông tin $\mathcal{M}$ (chẳng hạn như mô hình Independent Cascade hoặc Linear Threshold). Với mỗi tập hạt giống $S \subseteq V$, ký hiệu $g(S)$ là kỳ vọng số lượng nút được kích hoạt khi quá trình lan truyền kết thúc, hay nói cách khác là giá trị ảnh hưởng của S . Hàm g trong trường hợp này được chứng minh là submodular và đơn điệu tăng [1].

Cho trước một ngưỡng ảnh hưởng $T > 0$, mục tiêu là tìm một tập hạt giống $S \subseteq V$ sao cho $g(S) \geq T$ với lượng ngân sách là tối thiểu, chẳng hạn như số lượng tập *hạt giống* $|S|$ hoặc tổng chi phí bỏ ra $\sum_{v \in S} c(v)$.

Bên cạnh đó, ứng dụng của MCSC xuất hiện trong nhiều lĩnh vực như lan truyền ảnh hưởng trong OSN [64, 82, 81, 83], *phân bổ ngân sách tối ưu* [65], *phủ đồ thị tổng quát* [84], *tóm tắt dữ liệu* [85, 86], *ngăn chặn thông tin sai lệch* [80, 87], v.v.

Tuy nhiên, việc giải bài toán MCSC trong quy mô lớn phải đối mặt với nhiều thách thức đáng kể. Thứ nhất, các thuật toán đòi hỏi số lượng lớn các phép truy vấn tới hàm g . Thứ hai, sự gia tăng nhanh chóng của kích thước dữ liệu đầu vào khiến không gian tìm kiếm tăng theo cấp số mũ. Cuối cùng, các ràng buộc về bộ nhớ và thời gian tính toán trở thành rào cản nghiêm trọng đối với việc triển khai trong thực tế.

2.2. CÁC NGHIÊN CỨU LIÊN QUAN

MCSC được nghiên cứu từ rất sớm với các tiếp cận tham lam để thu được các tỷ lệ xấp xỉ về mặt lý thuyết, tiêu biểu là các nghiên cứu [42, 64, 16, 65, 66]. Về mặt độ phức tạp tính toán, Feige [67] đã chứng minh rằng MCSC là NP-khó và không thể được xấp xỉ với tỷ lệ hằng số, trừ khi $NP \subset DTIME(n^{O(\log \log n)})$. Kết quả này cho thấy những giới hạn lý thuyết nghiêm ngặt đối với khả năng thiết kế các TTXH đa thức cho bài toán.

Một hướng tiếp cận cổ điển cho MCSC là chiến lược tham lam, trong đó thuật toán xây dựng nghiệm bằng cách tuần tự chọn phần tử có lợi ích biên lớn nhất tại mỗi bước. Wolsey [42] là một trong những công trình tiên phong nghiên cứu bài toán này và đã đề xuất thuật toán tham lam với tỷ lệ xấp xỉ $1 + \ln\left(\max_{e \in V} \frac{f(e)}{\beta}\right)$, trong đó β là giá trị lợi ích biên dương nhỏ nhất của một phần tử được thêm vào trong quá trình thực hiện thuật toán. Sau đó, [64] phát triển một thuật toán tham lam khác cho một trường hợp đặc biệt của MCSC trong OSN, với hàm mục tiêu là số người dùng kỳ vọng bị ảnh hưởng. Thuật toán này đảm bảo chi phí không vượt quá $1 + \log(T/\epsilon)$ và giá trị hàm mục tiêu đạt ít nhất $T - \epsilon$, với $\epsilon > 0$ là tham số đầu vào.

Các chiến lược tham lam tiếp tục được mở rộng và phân tích trong những nghiên cứu sau này. Cụ thể, [16, 65] đã đề xuất và phân tích các thuật toán tham lam cho những trường hợp tổng quát hơn của MCSC cùng với các bảo đảm lý thuyết tương ứng. Trong đó, [16] xét trường hợp hàm chi phí c là hàm con điều hòa và chứng minh rằng thuật toán tham lam đạt tỷ lệ xấp xỉ $\rho \ln T$, với $\rho = \min_{S \subseteq V} \frac{\sum_{e \in S} c(e)}{c(S)}$ là độ cong (*curvature*) của hàm chi phí. Tuy nhiên, kết quả này chỉ còn hiệu lực khi hàm phủ g nhận giá trị nguyên.

Mặc dù các thuật toán tham lam truyền thống mang lại hiệu quả tốt về chất lượng nghiệm cho MCSC, cách tiếp cận tuần tự, tức là mỗi lần chỉ bổ sung một phần tử vào nghiệm hiện tại dẫn tới khả năng thực thi trên dữ liệu thực bị giới hạn khi kích thước dữ liệu n tăng lên. Nhằm cải thiện hạn chế này, [66] đã đề xuất một thuật toán tiến hóa hiệu quả, cung cấp nghiệm xấp xỉ tiêu chuẩn $(1 + \log(1/\epsilon), 1 - \epsilon)$ với $\epsilon > 0$. Tuy nhiên, thuật toán này vẫn đòi hỏi số vòng lặp tuần tự kỳ vọng lên tới $O(n^3)$, do đó chưa thực sự hiệu quả đối với các bài toán quy mô lớn.

Một hướng tiếp cận phổ biến nhằm vượt qua các thách thức trên là song song hóa thuật toán. Trong bối cảnh này, *độ phức tạp song song* (*adaptive complexity*, hay *adaptivity*) (đã được đề cập trong Chương 1) đóng vai trò như một thước đo quan trọng để đánh giá khả năng song song hóa của một thuật toán. Thước đo này được định nghĩa là số vòng lặp tuần tự tối thiểu, trong đó tại mỗi vòng, thuật toán có thể thực hiện song song một số đa thức các truy vấn độc lập để đánh giá hàm g . Tuy vậy, nhược điểm tính toán chính của các phương pháp nêu trên nằm ở việc phải liên tục đánh giá hàm mục tiêu cho nhiều nghiệm trung gian, khiến độ phức tạp song song của chúng đạt tới $O(\text{poly}(n))$ trong trường hợp xấu nhất, và do đó khó có thể song song hiệu quả.

Vào năm 2022, tại hội nghị Quốc tế uy tín đầu ngành về Lý thuyết học máy (Conference on Learning Theory (COLT, Rank A*)) năm 2022, Ran và cộng sự [19] đã đề xuất một thuật toán song song cho bài toán MCSC với tỷ lệ xấp xỉ

$$\frac{H(\min\{\max_{e \in S} g(e), T\})}{1 - 5\epsilon},$$

và độ phức tạp song song là

$$O\left(\frac{\log(nT) \log(T) (\log T + \log \log(nT))}{\epsilon^4}\right),$$

trong đó $\epsilon > 0$ là tham số chính xác, hàm $H(\cdot) : \mathbb{N} \mapsto \mathbb{R}_+$ được định nghĩa $H(n) = 1 + \frac{1}{2} + \dots + \frac{1}{n}$. Tuy nhiên, độ phức tạp song song của thuật toán này vẫn còn khá lớn do phụ thuộc vào $\log T$ và $\frac{1}{\epsilon^4}$, khiến khả năng song song hóa trong thực tiễn bị hạn chế. Hơn nữa, thuật toán này chỉ áp dụng cho các hàm g nhận giá trị nguyên, điều này làm giảm đáng kể tính phù hợp đối với các ứng dụng thực tế, nơi hàm tiện ích thường nhận giá trị thực.

Bên cạnh bài toán MCSC tổng quát, bài toán *Tập phủ*, cũng thu hút sự quan tâm của cộng đồng học thuật. Blelloch và cộng sự [88] là những người đầu tiên song song hóa thuật toán tham lam cho bài toán Tập phủ và các bài toán liên quan, đề xuất thuật toán ngẫu nhiên với $O(\log^2 M)$ vòng lặp tuần tự, đạt tỷ lệ xấp xỉ $1 + H(n)$, trong đó M là tổng kích thước của tất cả các tập.

Bên cạnh đó, [85] nghiên cứu bài toán Tập phủ trong mô hình phân tán và đề xuất thuật toán chạy trong $O(\log(|S^*| + \log T))$ vòng, trả về nghiệm S thỏa mãn $g(S) \geq T\lambda$ với kích thước không vượt quá $2(1 + \log T)|S^*|/\lambda$, trong đó $\lambda > 0$ là tham số đầu vào. Gần đây hơn, Fahrback và cộng sự [89] đã giới thiệu thuật toán *Threshold Sampling*, cho phép giải bài toán Tập phủ với nghiệm S thỏa mãn $g(S) \geq T$ và $|S| = O(\log T) \cdot \text{OPT}$ theo kỳ vọng, với độ phức tạp song song là $O(\log(n \log T) \log T)$.

Không chỉ MCSC và bài toán Tập phủ, mô hình tính toán song song còn được khai thác cho nhiều bài toán tối ưu khác, chẳng hạn như bài toán tập phủ ngẫu nhiên [90]. *Các kết quả này nhấn mạnh tầm quan trọng ngày càng lớn của việc phát triển các thuật toán vừa đảm bảo tỷ lệ xấp xỉ tốt, vừa hoạt động hiệu quả trong môi trường tính toán song song, từ đó góp phần rút ngắn đáng kể thời gian tìm kiếm lời giải.*

Đóng góp của LA đối với bài toán MCSC và thảo luận về kết quả. Xuất phát từ các hạn chế và khoảng trống nghiên cứu đối với các nghiên cứu liên quan (xem Bảng 2.1), LA đề xuất thuật toán AdaptMCSC, một TTX tiêu chuẩn kép bài toán MCSC với hàm tiện ích g nhận giá trị thực tổng quát (có thể nhận giá trị thực dương). Cụ thể, thuật toán AdaptMCSC đạt được độ phức tạp song song là

$$O\left(\frac{\log(n) \log(1/\lambda)}{\epsilon^2}\right),$$

độ phức tạp truy vấn

$$O\left(\frac{n^2 \log^2(n/\epsilon) \log(\frac{1}{\lambda})}{\epsilon^3}\right),$$

¹ H được gọi là hàm điều hòa.

Bảng 2.1: Bảng so sánh các thuật toán hiện có cho bài toán MCSC và SC. Ký hiệu (a, b) tương ứng với TTX tiêu chuẩn kép (a, b) .

Nghiên cứu	Tỷ lệ xấp xỉ (tiêu chuẩn kép)	Truy vấn	Độ phức tạp song song
Bài toán Phủ Submodular (MCSC)			
Thuật toán trong [42]	$1 + \log(\frac{\max_{e \in S} g(e)}{\beta})$	$O(n^2)$	$O(n)$
Tham lam [64]	$(1 + \log \frac{1}{\epsilon}, 1 - \epsilon)$	$O(n^2)$	$O(n)$
MinSMC [19]	$\frac{H(\min\{\max_{e \in S} g(e), T\})}{1 - 5\epsilon}$	$O(\frac{n \log(nT) \log(T) (\log T + \log \log(nT))}{\epsilon^4})$	$O(\frac{\log(nT) \log(T) (\log T + \log \log(nT))}{\epsilon^4})$
LA	$((1 + \epsilon)(1 + \ln \frac{1}{\lambda}), 1 - \lambda)$	$O(\frac{n^2 \log^2(n/\epsilon) \log(\frac{1}{\lambda})}{\epsilon^3})$ hoặc $O(\frac{n \log(n) \log^2(n/\epsilon) \log(\frac{1}{\lambda})}{\epsilon^3})$	$O(\frac{\log(n) \log(1/\lambda)}{\epsilon^2})$ hoặc $O(\frac{\log^2(n) \log(1/\lambda)}{\epsilon^2})$
Phủ submodular với chi phí đồng nhất			
[89]	$O(\log T)$	$O(n \log(n \log T) \log T)$	$O(\log(n \log T) \log T)$
LA	$((1 + \epsilon)(1 + \ln \frac{1}{\lambda}), 1 - \lambda)$	$O(\frac{n^2 \log^2(n) \log(\frac{1}{\lambda})}{\epsilon^3})$ hoặc $O(\frac{n \log^3(n) \log(\frac{1}{\lambda})}{\epsilon^3})$	$O(\frac{\log(n) \log(1/\lambda)}{\epsilon^2})$ hoặc $O(\frac{\log^2(n) \log(1/\lambda)}{\epsilon^2})$

và trả về một nghiệm S thỏa mãn:

$$g(S) \geq T(1 - \lambda), c(S) \leq (1 + \epsilon) \left(1 + \ln \frac{1}{\lambda}\right) \text{opt}$$

theo kỳ vọng, trong đó $\epsilon, \lambda \in (0, 1)$ là các tham số đầu vào và opt là tổng chi phí của nghiệm tối ưu. So với các thuật toán hiện nay, thuật toán đề xuất đạt được những đóng góp sau

- *Tỷ lệ xấp xỉ tiêu chuẩn kép*: Thuật toán AdaptMCSC đạt tỷ lệ xấp xỉ tiêu chuẩn kép tiệm cận với thuật toán tham lam [64], thuật toán hiện đang có tỷ lệ xấp xỉ tốt nhất với bài toán này. Cụ thể, nếu thiết lập tham số $\lambda = \epsilon$, AdaptMCSC cho tỷ lệ xấp xỉ tiêu chuẩn kép $((1 + \epsilon)(1 + \ln(\frac{1}{\epsilon})), 1 - \epsilon)$: đạt được chặn tương tự với thuật toán tham lam về hàm g , chặn về chi phí lời giải chỉ cao hơn một đại lượng rất nhỏ $1 + \epsilon$ với thuật toán tham lam [64]. AdaptMCSC có độ phức tạp song song thấp hơn so với thuật toán tham lam [64] số lần là

$$\frac{n}{\log n} \cdot \frac{\epsilon^2}{\log(\frac{1}{\lambda})} = \Omega\left(\frac{n}{\log(n)}\right).$$

Đây là hệ số rất đáng về vì n tăng nhanh hơn nhiều so với $\log n$.

- *Độ phức tạp song song*: So sánh với MinSMC [19], thuật toán có độ phức tạp song song thấp tốt nhất hiện nay, AdaptMCSC **loại bỏ hoàn toàn sự phụ thuộc vào tham số $\log T$ và đồng thời làm giảm đáng kể sự phụ thuộc vào $\frac{1}{\epsilon}$** . Cụ thể, AdaptMCSC giảm độ phức tạp song song so với [19] số lần là:

$$\Omega\left(\frac{1}{\epsilon^2} \left(\log^2 T + \frac{\log^3 T}{\log n}\right)\right) \gg \frac{\log^2(T)}{\epsilon^2}.$$

Tức là thuật toán đề xuất có độ phức tạp song song tốt hơn ít nhất $\frac{\log^2(T)}{\epsilon^2}$ lần so với thuật toán của Ran và cộng sự [19] về mặt lý thuyết.

- *Tính tổng quát:* Thuật toán AdaptMCSC áp dụng được với trường hợp tổng quát khi giá trị hàm g là thực, trong khi MinSMC chỉ áp dụng được với hàm g nhận giá trị nguyên.

Đối với bài toán Phủ submodular với chi phí đồng nhất (SC), là trường hợp đặc biệt của MCSC, thuật toán đề xuất yêu cầu số vòng lặp tuần tự ít hơn so với thuật toán tốt nhất hiện nay của Fahrback và cộng sự [89] một hệ số

$$\Omega \left(\log T + \frac{\log \log(T) \log T}{\log n} \right).$$

Đáng chú ý, các thuật toán trong [19, 89] chỉ áp dụng cho trường hợp hàm g nhận giá trị nguyên; do đó, AdaptMCSC là thuật toán đầu tiên có thể xử lý bài toán MCSC và SC với hàm g nhận giá trị thực tổng quát. Ngoài ra, khi kết hợp kỹ thuật tìm kiếm nhị phân vào thuật toán, độ phức tạp truy vấn có thể được giảm xuống gần tuyến tính, đạt $O(n \log^3 n)$, trong khi độ phức tạp song song là $O(\log^2 n)$. Ngay trong trường hợp này, thuật toán đề xuất vẫn cho hiệu năng song song vượt trội so với phương pháp của Ran và cộng sự [19], do không còn phụ thuộc vào tham số T .

2.3. THUẬT TOÁN SONG SONG AdaptMCSC

Về mặt kỹ thuật, thuật toán AdaptMCSC kết hợp kỹ thuật *lấy mẫu tuần tự* [30] và chiến lược *tham lam theo ngưỡng* được giới thiệu trong [24]. Thuật toán bao gồm một vòng lặp chính; tại mỗi lần lặp, thuật toán lựa chọn một tập con có “*mật độ lợi ích*” lớn, được định nghĩa là tỉ lệ giữa lợi ích biên và chi phí. Đáng chú ý, toàn bộ quá trình chỉ yêu cầu $O(\log n)$ vòng lặp tuần tự.

Để hiện thực hóa ý tưởng này, LA đề xuất một thủ tục con ký hiệu là THRESHSEQ, có nhiệm vụ lựa chọn một tập phần tử thỏa mãn một ngưỡng mật độ lợi ích cho trước nhằm lựa chọn các tập phần tử có giá trị vào tập lời giải. Trong thuật toán chính, thuật toán điều chỉnh chiến lược tham lam theo ngưỡng trong [24] theo hướng chỉ thực hiện một số vòng lặp tuần tự hằng số. Tại mỗi vòng, thuật toán gọi THRESHSEQ như một thủ tục con để lựa chọn một tập phần tử có mật độ lợi ích không nhỏ hơn một ngưỡng thích hợp. Cuối cùng, LA sử dụng kỹ thuật phân hoạch tập nền V được đề xuất trong [19] nhằm không chế tỉ lệ $c_{\max}/c_{\min}$, từ đó suy ra các cận chặt cho độ phức tạp truy vấn và độ phức tạp song song của thuật toán.

Với mục tiêu trên, trước hết LA giới thiệu *thuật toán Ngưỡng Tuần tự (Threshold Sequence)*, ký hiệu THRESHSEQ) và phân tích các tính chất lý thuyết được sử dụng trong

thuật toán chính (AdaptMCSC). Tiếp theo, LA trình bày thuật toán tổng quát cùng với các phân tích đầy đủ về đảm bảo xấp xỉ và độ phức tạp.

2.3.1. Thuật toán ngưỡng tuần tự THRESHSEQ

Trong phần này LA giới thiệu THRESHSEQ, là một thành phần quan trọng trong thuật toán chính AdaptMCSC. Thuật toán nhận đầu vào là một tập các phần tử X (gồm có các phần tử được xem xét lựa chọn vào lời giải), một ngưỡng θ và tham số $\epsilon \in (0, 1/6)$.

2.3.1.1. Mô tả thuật toán

Dữ liệu đầu vào của thuật toán gồm: một tập các phần tử (ứng viên) X , hàm g , ngưỡng $\theta > 0$ và tham số $\epsilon \in (0, 1)$ cho trước, thuật toán THRESHSEQ chạy trong $O(\log n)$ vòng lặp tuần tự, và trả về một tập con $S \subseteq X$ có *mật độ lợi ích* (tỷ lệ giữa giá trị hàm g và tổng chi phí) không nhỏ hơn $(1 - 2\epsilon)\theta$, tức là $g(S) \geq (1 - 2\epsilon)c(S)\theta$. Đồng thời, thuật toán còn đảm bảo rằng với mọi phần tử không được chọn $e \in X \setminus S$ thì $g(e \mid S) < \theta c(e)$. Về ý tưởng, thuật toán này hoạt động tương tự như thủ tục *Threshold-Sampling* trong [89, 91, 54]. Tại mỗi vòng lặp, kỹ thuật này sinh ra một *chuỗi ngẫu nhiên duy nhất* $A = [a_1, \dots, a_{|X|}]$ từ X (Thuật toán 2.3). Sau đó, một tiền tố $A_{i^*} = [a_1, \dots, a_{i^*}]$, với $i^* \leq |X|$, được thêm vào nghiệm, trong đó i^* là chỉ số lớn nhất sao cho phần tử a_{i^*} có lợi ích biên lớn đối với $S \cup A_{i-1}$. Thuật toán đánh giá *tỷ lệ giữa lợi ích biên của a_i đối với $S \cup A_{i-1}$ và chi phí của nó*. Theo đó, tại mỗi vòng lặp, thuật toán xác định chỉ số lớn nhất i^* sao cho tỷ lệ này không nhỏ hơn ngưỡng θ .

Cụ thể, thuật toán THRESHSEQ gồm hai vòng lặp: vòng lặp ngoài (Dòng 2–9) và vòng lặp trong (Dòng 4–6). Ở mỗi vòng lặp ngoài, thuật toán lấy mẫu một chuỗi $A = [a_1, a_2, \dots, a_d]$ (Dòng 3) thông qua thủ tục con $\text{SamSeq}(X, S)$. Tiếp theo, trong vòng lặp trong, thuật toán tính các tập

$$X_i = \{a \in X : g_{S \cup A_i}(a) \geq \theta \cdot c(a)\}, \quad i = 1, \dots, d,$$

nhằm xác định chỉ số i^* tại Dòng 8, và cập nhật các tập S và X (Dòng 8–9). Sau mỗi vòng lặp ngoài, tổng chi phí của tập X giảm ít nhất một hệ số $1 - \epsilon$. Thuật toán tiếp tục quá trình này cho đến khi X rỗng.

Nếu áp dụng kỹ thuật tìm kiếm nhị phân để xác định X_{i^*} thay cho các Dòng 3–4 của Thuật toán 2.1, với

$$i^* \leftarrow \arg \max_{i \in [d]} \{c(X_i) \geq (1 - \epsilon)c(X)\},$$

(thuật toán thu được được ký hiệu là Thuật toán 2.2, hay THRESHSEQBi), ta có thể giảm độ phức tạp truy vấn nhưng phải “trả giá” bằng số vòng lặp tuần tự tăng lên thêm $O(\log n)$

lần. Việc có thể áp dụng được tìm kiếm nhị phân là do với $A_i = \{a_1, a_2, \dots, a_i\}$ thì khi $i \in [d]$ tăng thì số phần tử của tập X_i càng nhỏ do vậy $c(X_i)$ giảm. Do đó có thể áp dụng phương pháp tìm kiếm nhị phân.

Thuật toán 2.1: THRESHSEQ(g, X, θ, ϵ)

Input: Tập X , ngưỡng $\theta > 0, \epsilon \in (0, 1)$

Output: S

1. $S \leftarrow \emptyset, X \leftarrow \{e \in X : g(e) \geq \theta \cdot c(e)\}$
 2. **while** $X \neq \emptyset$ **do**
 3. $A \leftarrow [a_1, a_2, \dots, a_d] \leftarrow \text{SamSeq}(X, S)$
 4. **for** $i = 1, \dots, d$ (*in parallel*) **do**
 5. $A_i \leftarrow \{a_1, a_2, \dots, a_i\}, A_{d+1} \leftarrow A_d$
 6. $X_i \leftarrow \{a \in X : g_{S \cup A_i}(a) \geq \theta \cdot c(a)\}$
 7. **end**
 8. $i^* \leftarrow \max\{i : c(X_i) \geq (1 - \epsilon)c(X)\}$
 9. $S \leftarrow S \cup A_{i^*+1}$
 10. $X \leftarrow X_{i^*+1}$
 11. **end**
 12. **return** S
-

2.3.1.2. Phân tích lý thuyết

Các kết quả sau đây đóng vai trò then chốt trong LA, bởi chúng thiết lập một cách chặt chẽ mối liên hệ giữa *chất lượng nghiệm* và *độ phức tạp song song* của thuật toán Ngưỡng tuần tự THRESHSEQ. Cụ thể, các bổ đề không chỉ chứng minh rằng THRESHSEQ đạt được mật độ lợi ích mong muốn theo kỳ vọng, mà còn chỉ ra rằng thuật toán có thể được triển khai với số vòng song song logarit, ngay cả khi xử lý các hàm submodular nhận giá trị thực và chi phí tổng quát. Đây chính là nền tảng lý thuyết quan trọng cho việc xây dựng thuật toán chính của LA với khả năng song song hóa cao và khả năng mở rộng tốt. Độ phức tạp tính toán của thuật toán được phát biểu trong Bổ đề 2.1

Bổ đề 2.1. *Thuật toán THRESHSEQ có độ phức tạp truy vấn là*

$$O\left(\frac{n^2}{\epsilon} \log\left(\frac{nc_{\max}}{c_{\min}}\right)\right)$$

và độ phức tạp song song là

$$O\left(\frac{1}{\epsilon} \log\left(\frac{nc_{\max}}{c_{\min}}\right)\right).$$

Thuật toán 2.2: THRESHSEQBi(g, X, θ, ϵ)

Input: Hàm g , tập X , ngưỡng $\theta > 0$, tham số $\epsilon \in (0, 1)$

Output: S

1. $S \leftarrow \emptyset, X \leftarrow \{e \in X : g(e) \geq \theta \cdot c(e)\}$
 2. **while** $X \neq \emptyset$ **do**
 3. $A \leftarrow [a_1, a_2, \dots, a_d] \leftarrow \text{SamSeq}(X, S)$
 4. $l \leftarrow 0, r \leftarrow d, m \leftarrow 0$
 5. **while** $l < r$ **do**
 6. $m \leftarrow \lceil (l + r)/2 \rceil$
 7. $A_m \leftarrow \{a_1, a_2, \dots, a_m\},$
 8. $X_m \leftarrow \{a \in X : g_{S \cup A_m}(a) \geq \theta \cdot c(a)\}$
 9. **if** $c(X_m) \geq (1 - \epsilon)c(X)$ **then**
 10. $l = m$
 11. **else**
 12. $r = m - 1$
 13. **end**
 14. **end**
 15. $S \leftarrow S \cup A_{l+1}$
 16. $X \leftarrow X_{l+1}$
 17. **end**
 18. **return** S
-

Thuật toán 2.3: SamSeq(X, S): Sinh một dãy phần tử ngẫu nhiên nằm ngoài S

Input: Hai tập $X, S \subseteq V$

Output: Dãy có thứ tự A

1. $A \leftarrow [], i \leftarrow 1, X \leftarrow X \setminus S$
 2. **while** $X \neq \emptyset$ **do**
 3. Chọn a_i ngẫu nhiên theo phân bố đều từ X
 4. $A \leftarrow [a_1, a_2, \dots, a_{i-1}, a_i]$
 5. $X \leftarrow X \setminus \{a_i\}, i \leftarrow i + 1$
 6. **end**
 7. **return** A
-

Nếu áp dụng kỹ thuật tìm kiếm nhị phân để xác định X_{i^*} thay cho các Dòng 3–4 của Thuật toán 2.1, với

$$i^* \leftarrow \arg \max_{i \in [d]} \{c(X_i) \geq (1 - \epsilon)c(X)\},$$

(thuật toán thu được được ký hiệu là Thuật toán 2.2, hay THRESHSEQBi), thì độ phức tạp truy vấn giảm xuống

$$O\left(\frac{n \log n}{\epsilon} \log\left(\frac{nc_{\max}}{c_{\min}}\right)\right),$$

và độ phức tạp song song là

$$O\left(\frac{\log(n)}{\epsilon} \log\left(\frac{nc_{\max}}{c_{\min}}\right)\right).$$

Chứng minh. Thuật toán gồm hai vòng lặp chính: vòng lặp ngoài (vòng lặp while, dòng 2-11) và vòng lặp trong (vòng lặp for, dòng 4-7). Trong mỗi vòng lặp ngoài, thuật toán tính song song d tập con X_i trong vòng lặp trong (vì các tập này độc lập với nhau). Mỗi phép tính X_i yêu cầu $|X|$ truy vấn; do đó, tổng số truy vấn tới hàm g trong vòng lặp trong là $n|X| = O(n^2)$.

Sau mỗi vòng lặp ngoài, tổng chi phí của các phần tử hợp lệ trong X giảm ít nhất một hệ số $1 - \epsilon$. Gọi $c_0(X)$ là chi phí của X tại thời điểm ban đầu và l là số vòng lặp ngoài. Thuật toán kết thúc khi X rỗng, tức là $c_0(X) < c_{\min}$. Do đó, ta có

$$c_0(X)(1 - \epsilon)^l \leq c_{\min},$$

suy ra

$$l < \frac{1}{\epsilon} \log\left(\frac{c_0(X)}{c_{\min}}\right) \leq \frac{1}{\epsilon} \log\left(\frac{nc_{\max}}{c_{\min}}\right).$$

Vì các phép tính X_i trong vòng lặp trong có thể được thực hiện song song trong một vòng song song, nên độ phức tạp song song của thuật toán là

$$l = \frac{1}{\epsilon} \log\left(\frac{nc_{\max}}{c_{\min}}\right).$$

Nếu áp dụng tìm kiếm nhị phân để xác định i^* và X_{i^*} như trong Thuật toán 2.2, thì vòng lặp trong cần $O(\log d) = O(\log n)$ vòng. Tại mỗi vòng, thuật toán tính đồng thời các lợi ích biên $g_{S \cup A_m}(a)$ với mọi $a \in X$ để xây dựng X_m , do đó cần $O(|X|)$ truy vấn nhưng chỉ một vòng song song. Vì vậy, mỗi vòng lặp ngoài cần $O(|X| \log n) \leq O(n \log n)$ truy vấn và $O(\log n)$ vòng song song. Kết hợp với

$$O\left(\frac{1}{\epsilon} \log\left(\frac{nc_{\max}}{c_{\min}}\right)\right)$$

vòng lặp ngoài, ta thu được độ phức tạp truy vấn là

$$O\left(\frac{n \log n}{\epsilon} \log\left(\frac{nc_{\max}}{c_{\min}}\right)\right)$$

và độ phức tạp song song là

$$O\left(\frac{\log n}{\epsilon} \log\left(\frac{nc_{\max}}{c_{\min}}\right)\right).$$

□

Bổ đề 2.2 thiết lập bảo đảm quan trọng về *chất lượng nghiệm* của thuật toán Ngưỡng tuần tự THRESHSEQ. Kết quả này cho thấy rằng tập nghiệm thu được không chỉ đạt mật độ lợi ích cao theo kỳ vọng, mà còn đảm bảo rằng mọi phần tử không được chọn đều không thể đóng góp lợi ích trên chi phí vượt quá ngưỡng đã đặt. Thuộc tính này là then chốt để kết hợp THRESHSEQ vào thuật toán chính của LA và thiết lập các bảo đảm xấp xỉ toàn cục.

Bổ đề 2.2. Với mọi tập X và $\epsilon \in (0, 1/6)$, tại thời điểm kết thúc Thuật toán 2.1, ta có:

- 1) $\mathbb{E}[g(S)] \geq (1 - 2\epsilon)\theta \cdot \mathbb{E}[c(S)]$;
- 2) $g_S(e) < \theta \cdot c(e)$, với mọi $e \in X$.

Chứng minh. Khi Thuật toán 2.1 kết thúc, ta ký hiệu

$$S = \{s_1, s_2, \dots, s_q\}, \quad S_t = \{s_1, s_2, \dots, s_{t-1}\}, \quad 2 \leq t \leq q.$$

Xem xét quá trình lựa chọn tập lời giải của thuật toán như là một quá trình ngẫu nhiên. Gọi $\mathcal{H}_t$ là một quá trình ngẫu nhiên mô tả quá trình này cho đến thời điểm ngay trước khi phần tử s_t được thêm vào tập A . Với điều kiện $\mathcal{H}_t$, trước hết ta chứng minh rằng

$$\mathbb{E}[g_{S_t}(s_t)] \geq (1 - 2\epsilon)\theta \cdot \mathbb{E}[c(s_t)]. \quad (2.2)$$

Xét vòng lặp **while** mà tại đó phần tử s_t được thêm vào nghiệm S . Giả sử S_{old} là nghiệm S tại đầu vòng lặp đó, và $A(t)$ là tiền tố của chuỗi A tại vòng lặp tương ứng, cho đến nhưng không bao gồm s_t . Khi đó, ta có

$$S_t = S_{\text{old}} \cup A(t),$$

là tập tất cả các phần tử đã được thêm vào nghiệm trước s_t . Với điều kiện $\mathcal{H}_t$, các tập S_t , S_{old} và $A(t)$ là xác định, trong khi phần còn lại của chuỗi A là ngẫu nhiên. Ký hiệu

$$X_{(t)} = X \setminus (A(t) \cup S_{\text{old}}),$$

$$X_t = \{a \in X_{(t)} : g_{S_{\text{old}} \cup A(t)}(a) \geq \theta \cdot c(a)\}, \quad E_t = X_{(t)} \setminus X_t.$$

Theo quy tắc lựa chọn chỉ số i^* và phần tử s_t trong thuật toán, ta có

$$c(X_t) \geq (1 - \epsilon)c(X), \quad c(E_t) < \epsilon c(X).$$

Mỗi phần tử $e \in X_{(t)}$ được chọn đều với xác suất $p_e = 1/|X_{(t)}|$ thông qua thủ tục $\text{SamSeq}(X, S)$. Do đó, ta có

$$\mathbb{E}[g_{S_t}(s_t)] - (1 - 2\epsilon)\theta\mathbb{E}[c(s_t)] = \sum_{e \in X_{(t)}} p(e)g_{S_t}(e) - (1 - 2\epsilon)\theta \sum_{e \in X_{(t)}} p(e)c(e) \quad (2.3)$$

$$\begin{aligned} &\geq \sum_{e \in X_t} p(e)g_{S_t}(e) - \theta \sum_{e \in X_t} p(e)c(e) + \sum_{e \in E_t} p(e)g_{S_t}(e) \\ &\quad + 2\epsilon\theta \sum_{e \in X_t} p(e)c(e) - \theta \sum_{e \in E_t} p(e)c(e) \end{aligned} \quad (2.4)$$

$$\geq \sum_{e \in X_t} p(e) \underbrace{[g_{S_t}(e) - \theta c(e)]}_{\geq 0} + (2\epsilon(1 - \epsilon) - \epsilon)\theta \frac{c(X)}{|X_{(t)}|} \quad (2.5)$$

$$\geq (2\epsilon(1 - \epsilon) - \epsilon)\theta \frac{c(X)}{|X_{(t)}|} \quad (2.6)$$

$$\geq \epsilon(1 - 2\epsilon)\theta \frac{c(X)}{|X_{(t)}|} \geq 0, \quad (2.7)$$

trong đó phép biến đổi từ (2.4) sang (2.5) dựa trên các bất đẳng thức đơn giản sau:

$$\sum_{e \in E_t} p(e)g_{S_t}(e) \geq 0,$$

$$\sum_{e \in X_t} p(e)c(e) = \frac{c(X_t)}{|X_{(t)}|} \geq (1 - \epsilon) \frac{c(X)}{|X_{(t)}|},$$

và

$$\sum_{e \in E_t} p(e)c(e) = \frac{c(E_t)}{|X_{(t)}|} < \epsilon \frac{c(X)}{|X_{(t)}|}.$$

Để thuận tiện cho phân tích, ta giả sử rằng sau khi thuật toán kết thúc, nó tiếp tục bổ sung các phần tử giả có chi phí và giá trị bằng không cho đến khi hoàn tất n bước. Khi đó, ta phân tích dãy các quá trình ngẫu nhiên $\{\mathcal{H}_t\}_{t=1}^n$. Áp dụng bất đẳng thức (2.2), ta

suy ra:

$$\mathbb{E}[g(S)] = \mathbb{E}\left[\sum_{t=1}^n g_{S_t}(s_t)\right] = \sum_{t=1}^n \mathbb{E}[g_{S_t}(s_t)] \quad (2.8)$$

$$= \sum_{t=1}^n \mathbb{E}[\mathbb{E}[g_{S_t}(s_t) \mid \mathcal{H}_t]] \quad (2.9)$$

$$= \mathbb{E}\left[\sum_{t=1}^n \mathbb{E}[g_{S_t}(s_t) \mid \mathcal{H}_t]\right] \quad (2.10)$$

$$\geq (1 - 2\epsilon)\theta \mathbb{E}\left[\sum_{t=1}^n \mathbb{E}[c(s_t) \mid \mathcal{H}_t]\right] \quad (2.11)$$

$$= (1 - 2\epsilon)\theta \mathbb{E}\left[\sum_{t=1}^n c(s_t)\right] \quad (2.12)$$

$$= (1 - 2\epsilon)\theta \mathbb{E}[c(S)], \quad (2.13)$$

trong đó (2.8) và (2.10) xuất phát từ tính tuyến tính của kỳ vọng, (2.9) và (2.13) là hệ quả của định luật kỳ vọng toàn phần, và (2.11) theo bất đẳng thức (2.2) với điều kiện $\mathcal{H}_t$. Cuối cùng, thuật toán kết thúc khi $X = \emptyset$. Theo định nghĩa của các tập X_i , điều này suy ra $g_S(e) \leq \theta c(e)$, $\forall e \in X$. $\square$

2.3.2. Thuật toán AdaptMCSC

Phần này trình bày chi tiết thuật toán chính AdaptMCSC. Do quá trình xây dựng thuật toán gắn chặt với các bước phân tích lý thuyết, trước hết LA sẽ giới thiệu một phiên bản đơn giản của AdaptMCSC, dựa trên giả định rằng đã có một ước lượng gần đúng cho giá trị tối ưu (ký hiệu là AdaptMCSCopt). Cụ thể, giả sử tồn tại một giá trị v thỏa mãn:

$$v(1 - \epsilon) \leq \text{opt} \leq v.$$

Dưới giả thiết này, ta có thể mô tả rõ ràng ý tưởng thiết kế và các bước phân tích của thuật toán. Sau đó, LA trình bày phiên bản đầy đủ của thuật toán AdaptMCSC, trong đó giả thiết về giá trị v được loại bỏ.

2.3.2.1. Thuật toán AdaptMCSCopt (biết ước lượng tối ưu)

Trước hết, giả sử rằng ta đã biết một giá trị ước lượng v của chi phí tối ưu. Trong trường hợp này, LA trình bày AdaptMCSCopt, một phiên bản đơn giản hóa của thuật toán chính. Thuật toán AdaptMCSCopt cho phép thu được một nghiệm xấp xỉ tiêu chuẩn kép (bicriteria approximation) với độ phức tạp song song là $O(\log n)$.

Với một ngưỡng $T > 0$, ta định nghĩa hàm $h(S) = \min\{T, g(S)\}$ với mọi $S \subseteq V$. Dễ thấy rằng $h(\cdot)$ vẫn là một hàm không âm, đơn điệu và submodular. Thuật toán

AdaptMCSCopt nhận vào một thể hiện (V, g, T) của bài toán, các tham số $\epsilon, \lambda > 0$ và một giá trị ước lượng chi phí v thỏa mãn $v(1 - \epsilon) \leq \text{opt} \leq v$. Ở mức ý tưởng tổng quát, thuật toán đề xuất kế thừa và điều chỉnh phương pháp tham lam ngưỡng [24] nhằm lọc ra một tập con các phần tử có “mật độ lợi ích” cao, tức là tỉ lệ giữa lợi ích biên và chi phí, tại mỗi vòng lặp.

Cụ thể, thuật toán bao gồm một vòng lặp chính với nhiều nhất $\frac{1}{\epsilon} \log(\frac{1}{\lambda})$ lần lặp. Trong mỗi vòng lặp, thuật toán lựa chọn một tập con các phần tử có mật độ lợi ích không nhỏ hơn một ngưỡng $\theta = (1 - \epsilon)(T - h(S))/v$, tức là chọn ra tập X gồm các phần tử e với

$$\frac{h_S(e)}{c(e)} := \frac{h(S \cup \{e\}) - h(S)}{c(e)} \geq \theta. \quad (2.14)$$

Sau đó thuật toán gọi thủ tục THRESHSEQ chọn ra tập S_i "có giá trị cao" để thêm vào lời giải S (Dòng 5), thông qua việc sử dụng thủ tục THRESHSEQ. Quá trình này được lặp lại cho đến khi giá trị của nghiệm hiện tại đạt ít nhất $T(1 - \lambda)$ hoặc số vòng lặp là $\frac{1}{\epsilon} \log(\frac{1}{\lambda})$. Mô tả chi tiết của thuật toán được trình bày trong Thuật toán 2.4.

Thuật toán 2.4: AdaptMCSCopt($V, g, T, \epsilon, \lambda, v$)

Input: Thể hiện (V, g, T) , $\epsilon \in (0, 1/6)$, $\lambda \in (0, 1)$, ước lượng tối ưu v

Output: Lời giải S .

1. $S \leftarrow \emptyset, h(\cdot) = \min\{T, g(\cdot)\}$
 2. **for** $i = 1$ **to** $\lceil \ln(1/\lambda)/\epsilon \rceil$ **do**
 3. $\theta \leftarrow (1 - \epsilon)(T - h(S))/v$
 4. $X \leftarrow \{e \in V : h_S(e)/c(e) \geq \theta\}$
 5. $S_i \leftarrow \text{THRESHSEQ}(h_S(\cdot), X, \theta, \epsilon/2)$
 6. $S \leftarrow S \cup S_i$
 7. **If** $h(S) \geq (1 - \lambda)T$ **then break;**
 8. **end**
 9. **return** S
-

Để hoàn thiện phân tích thuật toán AdaptMCSCopt, tiếp theo LA lần lượt chứng minh các chặn về độ phức tạp thuật toán và tốc độ hội tụ so với lời giải tối ưu của lời giải trong mỗi vòng lặp.

Trước hết, độ phức tạp của thuật toán được phát biểu trong Bổ đề 2.3.

Bổ đề 2.3. Với $\lambda > 0$ và $\epsilon \in (0, 1)$ là các tham số đầu vào Thuật toán 2.4 có độ phức tạp truy vấn là $O(\frac{n^2}{\epsilon^2} \log(\frac{1}{\lambda}) \log(\frac{nc_{\max}}{c_{\min}}))$ và độ phức tạp song song là $O(\frac{1}{\epsilon^2} \log(\frac{1}{\lambda}) \log(\frac{nc_{\max}}{c_{\min}}))$. Đối với bài toán SC, thuật toán có độ phức tạp truy vấn là $O(\frac{n^2}{\epsilon^2} \log(\frac{1}{\lambda}) \log n)$ và độ phức tạp song song là $O(\frac{1}{\epsilon^2} \log(\frac{1}{\lambda}) \log n)$.

Nếu trong Thuật toán 2.4 ta sử dụng THRESHSEQBi thay cho THRESHSEQ, thì thuật toán có độ phức tạp truy vấn là $O(\frac{n}{\epsilon^2} \log(n) \log(\frac{1}{\lambda}) \log(\frac{nc_{\max}}{c_{\min}}))$ và độ phức tạp song song là $O(\frac{1}{\epsilon^2} \log(\frac{1}{\lambda}) \log(n) \log(\frac{nc_{\max}}{c_{\min}}))$.

Chứng minh. Thuật toán 2.4 thực hiện $\log(1/\lambda)/\epsilon$ vòng lặp tuần tự trong vòng lặp ngoài (các dòng 2–8). Trong mỗi vòng lặp, thuật toán gọi thủ tục THRESHSEQ như một chương trình con.

Kết hợp với Bổ đề 2.1, thuật toán THRESHSEQ có độ phức tạp song song $O(\frac{1}{\epsilon} \log(\frac{nc_{\max}}{c_{\min}}))$ và độ phức tạp truy vấn là $O(\frac{1}{\epsilon} \log(\frac{nc_{\max}}{c_{\min}}))$ ta suy ra rằng Thuật toán 2.4 có độ phức tạp song song là

$$\frac{1}{\epsilon} \log(\frac{1}{\lambda}) \cdot O(\frac{1}{\epsilon} \log(\frac{nc_{\max}}{c_{\min}})) = O(\frac{1}{\epsilon^2} \log(\frac{1}{\lambda}) \log(\frac{nc_{\max}}{c_{\min}})),$$

và độ phức tạp truy vấn là

$$\frac{1}{\epsilon} \log(\frac{1}{\lambda}) \cdot O\left(\frac{n^2}{\epsilon} \log\left(\frac{nc_{\max}}{c_{\min}}\right)\right) = O\left(\frac{n^2}{\epsilon^2} \log(\frac{1}{\lambda}) \log(\frac{nc_{\max}}{c_{\min}})\right).$$

Trong trường hợp bài toán SC, do $\frac{c_{\max}}{c_{\min}} = 1$, các chặn độ phức tạp song song và truy vấn rút gọn lần lượt là

$$O(\frac{1}{\epsilon^2} \log(\frac{1}{\lambda}) \log(n)), O(\frac{n^2}{\epsilon^2} \log(\frac{1}{\lambda}) \log(n)).$$

Nếu trong dòng 5 của Thuật toán 2.4 ta sử dụng THRESHSEQBi với độ phức tạp song song là $O(\frac{1}{\epsilon^2} \log(\frac{1}{\lambda}) \log(\frac{nc_{\max}}{c_{\min}}))$ thay cho THRESHSEQ, thì dựa trên Bổ đề 2.1 và lập luận hoàn toàn tương tự như trên, ta thu được các chặn độ phức tạp đã nêu. $\square$

Bổ đề 2.4 đưa ra tính chất của lời giải sau mỗi vòng lặp.

Bổ đề 2.4. Ký hiệu S^i là nghiệm S thu được tại thời điểm kết thúc vòng lặp thứ i của Thuật toán 2.4. Khi kết thúc vòng lặp thứ $(i + 1)$, ta có:

$$T - h(S^{i+1}) \leq \text{opt} \frac{(1 - \epsilon)}{v} (T - h(S^i)). \quad (2.15)$$

Chứng minh. Tại thời điểm kết thúc vòng lặp thứ $(i + 1)$, tập X trở thành rỗng, theo cơ chế hoạt động của Thuật toán 2.1. Theo Bổ đề 2.2, với mọi phần tử $e \in V$ bị loại khỏi V tại vòng lặp $(i + 1)$, ta có $h_{S^{i+1}}(e)/c(e) < \theta$.

Nhờ tính submodular của hàm h , ta suy ra

$$\begin{aligned}
T - h(S^{i+1}) &\leq h(S^* \cup S^{i+1}) - h(S^{i+1}) \\
&\leq \sum_{o \in S^* \setminus S^{i+1}} h_{S^{i+1}}(o) \quad (\text{do tính submodular của } h) \\
&\leq \sum_{o \in S^* \setminus S^{i+1}} c(o)\theta \leq \text{opt}\theta \\
&= \frac{\text{opt}(1 - \epsilon)}{v}(T - h(S^i)).
\end{aligned}$$

□

Bổ đề 2.4 cho thấy phần chênh lệch giữa $T - h(S)$ giảm theo cấp số nhân sau mỗi vòng lặp của thuật toán. Cụ thể, giá trị $T - h(S^{i+1})$ được chặn trên bởi một tỉ lệ không đổi của $T - h(S^i)$, phụ thuộc vào các tham số ϵ , v và opt . Kết quả này đóng vai trò quan trọng trong việc chứng minh rằng thuật toán AdaptMCSCopt đạt được tốc độ hội tụ nhanh và chỉ cần một số vòng lặp hữu hạn để tiến gần đến ngưỡng mục tiêu T .

Định lý 2.1. Với $\epsilon \in (0, 1/6)$, $\lambda \in (0, 1)$ và v thỏa mãn $v(1 - \epsilon) \leq \text{opt} \leq v$, thuật toán 2.4 trả về một nghiệm S sao cho:

$$g(S) \geq T(1 - \lambda), \mathbb{E}[c(S)] \leq (1 + \frac{19\epsilon}{4})(1 + \ln(\frac{1}{\lambda}))\text{opt}.$$

Chứng minh. Giả sử rằng vòng lặp ngoài của Thuật toán 2.4 được thực hiện trong Δ vòng lặp. Ta chứng minh định lý bằng cách xét các trường hợp khác nhau của Δ như sau.

Trường hợp 1. Giả sử $\Delta = \log(1/\lambda)/\epsilon$. Khi đó, ta có $h(S^{\Delta-1}) < (1 - \lambda)T$. Áp dụng Bổ đề 2.4, cùng với giả thiết $v(1 - \epsilon) \leq \text{opt} \leq v$, ta thu được

$$\begin{aligned}
T - h(S^\Delta) &\leq (1 - \epsilon)\frac{\text{opt}}{v}(T - h(S^{\Delta-1})) \\
&\leq (1 - \epsilon)(T - h(S^{\Delta-1})) \\
&\leq (1 - \epsilon)^\Delta T \\
&\leq \lambda T,
\end{aligned}$$

từ đó suy ra $g(S^\Delta) \geq h(S^\Delta) \geq (1 - \lambda)T$. Mặt khác, từ Bổ đề 2.2, với mọi $1 \leq i \leq \Delta$, ta có

$$\mathbb{E}[h(S^i) - h(S^{i-1})] \geq (1 - \epsilon)\theta \mathbb{E}[c(S^i \setminus S^{i-1})] \quad (2.16)$$

$$\geq (1 - \epsilon)^2 \frac{(T - h(S^{i-1}))}{v} \mathbb{E}[c(S^i \setminus S^{i-1})]. \quad (2.17)$$

Điều này kéo theo

$$T - \mathbb{E}[h(S^i)] \leq \left(1 - \frac{(1-\epsilon)^2}{v} \mathbb{E}[c(S^i \setminus S^{i-1})]\right) (T - \mathbb{E}[h(S^{i-1})]). \quad (2.18)$$

Áp dụng bất đẳng thức (2.18) nhiều lần, cùng với điều kiện $(1-\epsilon)v \leq \text{opt} \leq v$, ta có

$$\lambda T < T - \mathbb{E}[h(S^{\Delta-1})] \quad (2.19)$$

$$\leq \left(1 - \frac{(1-\epsilon)^2 \mathbb{E}[c(S^{\Delta-1} \setminus S^{\Delta-2})]}{v}\right) (T - h(S^{\Delta-2})) \quad (2.20)$$

$$\leq e^{-\frac{(1-\epsilon)^2}{v} \sum_{i=1}^{\Delta-1} \mathbb{E}[c(S^i \setminus S^{i-1})]} T \quad (2.21)$$

$$\leq e^{-\frac{(1-\epsilon)^2}{v} \mathbb{E}[c(S^{\Delta-1})]} T \quad (2.22)$$

$$\leq e^{-\frac{(1-\epsilon)^3}{\text{opt}} \mathbb{E}[c(S^{\Delta-1})]} T. \quad (2.23)$$

Từ đó suy ra

$$\mathbb{E}[c(S^{\Delta-1})] \leq \frac{1}{(1-\epsilon)^3} \ln\left(\frac{1}{\lambda}\right) \cdot \text{opt}.$$

Mặt khác, từ (2.17) và tính đơn điệu của h , ta có

$$\mathbb{E}[h(S^\Delta) - h(S^{\Delta-1})] \geq \frac{(1-\epsilon)^3 (T - h(S^{\Delta-1}))}{\text{opt}} \mathbb{E}[c(S^\Delta \setminus S^{\Delta-1})], \quad (2.24)$$

suy ra

$$\mathbb{E}[c(S^\Delta \setminus S^{\Delta-1})] \leq \frac{\text{opt} \mathbb{E}[h_{S^{\Delta-1}}(S^\Delta)]}{(1-\epsilon)^3 (T - h(S^{\Delta-1}))} \quad (2.25)$$

$$\leq \frac{\text{opt}}{(1-\epsilon)^3}. \quad (2.26)$$

Do đó,

$$\begin{aligned} \mathbb{E}[c(S^\Delta)] &= \mathbb{E}[c(S^{\Delta-1})] + \mathbb{E}[c(S^\Delta \setminus S^{\Delta-1})] \\ &\leq \frac{1}{(1-\epsilon)^3} (1 + \ln\left(\frac{1}{\lambda}\right)) \text{opt} \\ &\leq (1 + \frac{19}{4}\epsilon) (1 + \ln\left(\frac{1}{\lambda}\right)) \text{opt}, \end{aligned}$$

trong đó bất đẳng thức cuối cùng đúng do $\epsilon \in (0, 1/6)$, suy ra $1/(1-\epsilon)^3 \leq 1 + \frac{19}{4}\epsilon$.

Trường hợp 2. Giả sử $\Delta < \log(1/\lambda)/\epsilon$. Khi đó, ta vẫn có $h(S^\Delta) \geq (1-\lambda)T$ và $h(S^{\Delta-1}) < (1-\lambda)T$. Lập luận tương tự như các biến đổi từ (2.19) đến (2.23) cho thấy

$$\mathbb{E}[c(S^{\Delta-1})] \leq \frac{1}{(1-\epsilon)^3} \ln\left(\frac{1}{\lambda}\right) \cdot \text{opt}.$$

Đồng thời, tương tự như biến đổi trong (2.26), ta có $\mathbb{E}[c(S^\Delta \setminus S^{\Delta-1})] \leq \frac{\text{opt}}{(1-\epsilon)^3}$. Do đó, ta thu được $g(S^\Delta) \geq h(S^\Delta) \geq (1-\lambda)T$ và

$$\begin{aligned} \mathbb{E}[c(S^\Delta)] &= \mathbb{E}[c(S^{\Delta-1})] + \mathbb{E}[c(S^\Delta \setminus S^{\Delta-1})] \\ &\leq (1 + \frac{19}{4}\epsilon) (1 + \ln\left(\frac{1}{\lambda}\right)) \text{opt}. \end{aligned}$$

Kết hợp hai trường hợp trên, ta hoàn tất chứng minh định lý. $\square$

2.3.2.2. Thuật toán chính AdaptMCSC

Mặc dù có thể thiết lập các đảm bảo lý thuyết cho phiên bản đơn giản AdaptMCSCopt, việc triển khai thuật toán này trong thực tế vẫn gặp hai thách thức quan trọng. Thứ nhất, thuật toán yêu cầu một ước lượng v của giá trị tối ưu theo điều kiện đầu vào, trong khi v thường không sẵn có. Thứ hai, các chặn độ phức tạp của AdaptMCSCopt còn phụ thuộc vào tỉ lệ chi phí $\rho = c_{\max}/c_{\min}$, mà trong trường hợp tổng quát có thể rất lớn so với kích thước đầu vào, làm suy giảm ý nghĩa thực tiễn của các đảm bảo.

Để khắc phục đồng thời hai hạn chế này, trong thuật toán chính AdaptMCSC LA đề xuất một cơ chế kết hợp gồm hai ý tưởng then chốt. (i) Trước hết, ta loại bỏ sự phụ thuộc vào ρ trong bài toán con trọng tâm bằng cách phân hoạch tập V thành hai phần phù hợp (theo cấu trúc chi phí), sao cho trên mỗi phần ta có thể kiểm soát tốt dải chi phí. (ii) Tiếp theo, ta áp dụng AdaptMCSCopt một cách thích hợp để tìm nghiệm chất lượng trên từng phần, rồi kết hợp các nghiệm thu được thành nghiệm cuối cùng. Quá trình kết hợp này đi kèm với một phân tích lý thuyết chặt chẽ nhằm đảm bảo chất lượng lời giải và độ phức tạp tổng thể của thuật toán AdaptMCSC.

a. Mô tả thuật toán

Dựa trên ý tưởng trên, thuật toán AdaptMCSC chia làm 2 pha như sau

Pha 1: Tìm lời giải hợp lệ với chặn chi phí tối ưu

Trong pha đầu tiên, mục tiêu của thuật toán là đưa ra chặn $c_{\max}/c_{\min}$ theo đa thức với n bằng cách điều chỉnh phương pháp được đề xuất bởi Ran và cộng sự [19]. Cụ thể, ta sắp xếp các phần tử trong tập nền

$$V = \{u_1, u_2, \dots, u_{|V|}\}$$

theo thứ tự tăng của chi phí, tức là $c(u_1) \leq c(u_2) \leq \dots \leq c(u_n)$. Sau đó, thuật toán xác định chỉ số

$$j = \min\{i : g(\{u_1, u_2, \dots, u_i\}) \geq T\}.$$

Dựa trên giá trị u_j , ta phân hoạch tập nền V thành ba tập con rời nhau:

$$V_0 = \{u \in V : c(u) < c'_{\min}\}, \quad V_1 = \{u \in V : c(u) > c'_{\max}\},$$

và

$$V' = V \setminus (V_0 \cup V_1),$$

trong đó các ngưỡng chi phí được xác định bởi

$$c'_{\min} \leftarrow \frac{\epsilon c(u_j)}{n}, \quad c'_{\max} \leftarrow j c(u_j).$$

Cách phân hoạch này đảm bảo rằng:

- 1) Mọi nghiệm tối ưu đều không chứa phần tử thuộc V_1 : Thật vậy, gọi S^* là nghiệm tối ưu và $\text{OPT} = c(S^*)$. Do $g(\{u_1, \dots, u_j\}) \geq T$, tập $\{u_1, \dots, u_j\}$ là một nghiệm khả thi. Vì các phần tử được sắp xếp theo chi phí tăng dần nên

$$\text{OPT} \leq \sum_{i=1}^j c(u_i) \leq j c(u_j) = c'_{\max}.$$

Do đó, mọi phần tử $u \in V_1$ thỏa mãn $c(u) > c'_{\max} \geq \text{OPT}$ nên không thể thuộc nghiệm tối ưu. Suy ra $S^* \cap V_1 = \emptyset$.

- 2) Các phần tử trong nghiệm chủ yếu nằm trong V' : Do $S^* \cap V_1 = \emptyset$, ta có $S^* \subseteq V_0 \cup V'$. Mặt khác,

$$c(V_0) \leq n c'_{\min} = \epsilon c(u_j) \leq \epsilon \text{OPT}.$$

Do đó,

$$c(S^* \cap V') = \text{OPT} - c(S^* \cap V_0) \geq (1 - \epsilon) \text{OPT}.$$

Như vậy, ít nhất tỷ lệ $1 - \epsilon$ tổng chi phí của nghiệm tối ưu nằm trong V' , còn phần thuộc V_0 chỉ chiếm nhiều nhất tỷ lệ ϵ .

- 3) V_0 là tập có chi phí rất nhỏ nên có thể được đưa trực tiếp vào nghiệm cuối cùng.

Do vậy, ta có thể xác định được độ phức tạp truy vấn ở pha này dựa vào việc tính toán số phần tử tối đa trong V' .

Pha 2: Giải các bài toán con

Trong pha thứ hai, thuật toán giải bài toán con trên thể hiện

$$(g_{V_0}, V', T - g(V_0)),$$

trong đó $g_{V_0}(S) = g(V_0 \cup S) - g(V_0)$ bằng cách gọi thuật toán AdaptMCSCopt như một thủ tục con nhiều lần, tương ứng với các giá trị ước lượng khác nhau của chi phí tối ưu. Gọi opt' là chi phí của nghiệm tối ưu cho bài toán con này. Do ta có

$$c'_{\min} \leq \text{opt}' \leq n c'_{\max},$$

nên một dãy các ước lượng hợp lý cho opt' có thể được xác định dưới dạng $v_i = \frac{c'_{\min}}{(1-\epsilon)^i}$. Với mỗi giá trị v_i , thuật toán AdaptMCSCopt được gọi để tìm ra một nghiệm ứng viên. Trong số các nghiệm ứng viên thu được, thuật toán lựa chọn nghiệm có chi phí nhỏ nhất S thỏa mãn

$$g(S) \geq (1 - \lambda)(T - g(V_0)).$$

Cuối cùng, thuật toán AdaptMCSC trả về nghiệm $S \cup V_0$. làm nghiệm đầu ra cho bài toán MCSC ban đầu. Toàn bộ các bước của thuật toán chính AdaptMCSC được mô tả chi tiết trong Thuật toán 2.5.

Thuật toán 2.5: Thuật toán AdaptMCSC

Input: Thể hiện (g, V, T) , $\epsilon \in (0, 1/6)$, $\lambda \in (0, 1)$.

Output: Tập lời S

1. $S \leftarrow \emptyset$, $\epsilon' \leftarrow 4\epsilon/23$
 2. Sắp xếp $V = \{u_1, u_2, \dots, u_{|V|}\}$ theo thứ tự tăng chi phí chi phí, tức là

$$c(u_1) \leq c(u_2) \leq \dots \leq c(u_n)$$
 3. $j \leftarrow \min\{i : g(\{u_1, u_2, \dots, u_i\}) \geq T\}$
 4. $c'_{\min} \leftarrow \epsilon' c(u_j)/n$, $c'_{\max} \leftarrow j c(u_j)$
 5. $V_0 \leftarrow \{u \in V : c(u) < c'_{\min}\}$
 6. $V_1 \leftarrow \{u \in V : c(u) > c'_{\max}\}$
 7. $V' \leftarrow V \setminus (V_0 \cup V_1)$, $T' \leftarrow T - g(V_0)$, $g'(\cdot) \leftarrow g_{V_0}(\cdot)$
 // Thực hiện vòng lặp sau một cách song song
 8. **for** $i = 0$ **to** $\lceil \log_{1/(1-\epsilon')} (n^3/\epsilon') \rceil$ **do**
 9. $v_i = c'_{\min}/(1 - \epsilon')^i$
 10. $S(i) \leftarrow \text{AdaptMCSCopt}(V', g', T', \epsilon', \lambda, v_i)$
 11. **end**
 12. $S \leftarrow \arg \min\{c(S(i)) : g(S(i)) \geq T'(1 - \lambda), i = 0, \dots, \lceil \log_{1/(1-\epsilon')} (n^3/\epsilon') \rceil\}$
 13. $S_{\text{final}} \leftarrow S \cup V_0$
 14. **return** S_{final}
-

Ví dụ 2.1 (Ví dụ thực hiện thuật toán AdaptMCSC). Xét một mạng xã hội có tập người dùng $V = \{u_1, u_2, \dots, u_{12}\}$. Mỗi người dùng đều có thể được chọn làm hạt giống. Chi phí của các đỉnh được cho như sau:

u	u_1	u_2	u_3	u_4	u_5	u_6	u_7	u_8	u_9	u_{10}	u_{11}	u_{12}
$c(u)$	20	15	10	8	4	1	100	100	100	100	100	100

Đối với mỗi người dùng $u_i \in V$, ký hiệu $I(u_i)$ là tập người dùng có thể bị ảnh hưởng bởi u_i . Hàm ảnh hưởng của tập người dùng $S \subseteq V$ được định nghĩa bởi

Bảng 2.2: Tập người dùng bị ảnh hưởng bởi mỗi người dùng.

Người dùng	Tập ảnh hưởng
u_1	$\{u_1, u_2, u_3, u_4, u_5\}$
u_2	$\{u_2, u_3, u_6, u_7\}$
u_3	$\{u_1, u_3, u_7, u_8, u_9\}$
u_4	$\{u_4, u_5, u_{10}\}$
u_5	$\{u_5, u_8, u_{11}\}$
u_6	$\{u_6, u_9, u_{12}\}$
u_7	$\{u_7\}$
u_8	$\{u_8\}$
u_9	$\{u_9\}$
u_{10}	$\{u_{10}\}$
u_{11}	$\{u_{11}\}$
u_{12}	$\{u_{12}\}$

$$g(S) = \left| \bigcup_{u \in S} I(u) \right|.$$

Chẳng hạn,

$$\begin{aligned}
 g(\{u_1, u_2\}) &= |I(u_1) \cup I(u_2)| \\
 &= |\{u_1, u_2, u_3, u_4, u_5\} \cup \{u_2, u_3, u_6, u_7\}| \\
 &= |\{u_1, u_2, u_3, u_4, u_5, u_6, u_7\}| = 7.
 \end{aligned}$$

Cho $T = 12$, $\epsilon = \lambda = 0.1$. Thuật toán hoạt động như sau.

- Dòng 1. Ta có

$$\epsilon' = \frac{4\epsilon}{23} = \frac{0.4}{23} \approx 0.01739.$$

- Dòng 2. Sắp xếp V theo thứ tự tăng dần chi phí. Một thứ tự phù hợp là

$$V = \{u_6, u_5, u_4, u_3, u_2, u_1, u_7, u_8, u_9, u_{10}, u_{11}, u_{12}\}.$$

Các đỉnh $u_7, \dots, u_{12}$ có cùng chi phí 100; thứ tự giữa chúng không ảnh hưởng đến các phép tính tiếp theo.

- Dòng 3. Tính giá trị của j . Với các tiền tố theo thứ tự trên, ta có

$$g(\{u_6, u_5, u_4, u_3\}) = 10,$$

trong khi

$$g(\{u_6, u_5, u_4, u_3, u_2\}) = 12.$$

Do đó, tiền tố đầu tiên đạt ngưỡng $T = 12$ tại $j = 5$.

- Dòng 4. Phần tử thứ 5 là u_2 , nên $c(u_j) = c(u_2) = 15$. Suy ra

$$c'_{\min} = \frac{\epsilon' c(u_j)}{n} = \frac{0.01739 \times 15}{12} \approx 0.0217,$$

$$c'_{\max} = j c(u_j) = 5 \times 15 = 75.$$

- Dòng 5–7. Ta có

$$V_0 = \{u \in V : c(u) < c'_{\min}\} = \emptyset,$$

$$V_1 = \{u \in V : c(u) > c'_{\max}\} = \{u_7, u_8, u_9, u_{10}, u_{11}, u_{12}\},$$

$$V' = V \setminus (V_0 \cup V_1) = \{u_1, u_2, u_3, u_4, u_5, u_6\},$$

$$T' = T - g(V_0) = 12 - g(\emptyset) = 12.$$

Vì $V_0 = \emptyset$, ta có

$$g'(\cdot) = g_{V_0}(\cdot) = g_{\emptyset}(\cdot) = g(\cdot).$$

- Dòng 8–11. Thuật toán tạo dãy các giá trị dự đoán

$$v_i = \frac{c'_{\min}}{(1 - \epsilon')^i}, \quad i = 0, 1, \dots, \left\lceil \log_{1/(1-\epsilon')} \left(\frac{n^3}{\epsilon'} \right) \right\rceil.$$

Một số giá trị của v_i là

i	0	1	2	100	200	300	400	500	600	665
v_i	0.0217	0.0221	0.0225	0.126	0.738	4.25	24.4	140.2	804.7	2155

Với mỗi giá trị v_i , thuật toán gọi

$$S(i) \leftarrow \text{AdaptMCSCopt}(V', g', T', \epsilon', \lambda, v_i).$$

Một số kết quả được liệt kê dưới đây:

i	v_i	$S(i)$	$g(S(i))$	$c(S(i))$
0	0.0217	$\{u_6\}$	3	1
1	0.0221	$\{u_5, u_6\}$	6	5
2	0.0225	$\{u_4, u_5, u_6\}$	8	13
5	0.0239	$\{u_3, u_5, u_6\}$	9	15
10	0.0261	$\{u_3, u_4, u_5, u_6\}$	10	23
20	0.0313	$\{u_2, u_5, u_6\}$	10	20
50	0.0520	$\{u_2, u_3, u_5, u_6\}$	11	30
100	0.126	$\{u_1, u_5, u_6\}$	10	25
200	0.738	$\{u_1, u_2, u_6\}$	10	36
300	4.25	$\{u_1, u_2, u_5, u_6\}$	11	40
350	10.26	$\{u_1, u_2, u_3, u_5, u_6\}$	12	50

- Dòng 12. Với $\lambda = 0.1$, điều kiện chọn nghiệm là

$$g(S(i)) \geq T'(1 - \lambda) = 12(1 - 0.1) = 10.8.$$

Vì $g(S(i))$ là số nguyên, các nghiệm thỏa mãn điều kiện phải có $g(S(i)) \geq 11$. Trong các kết quả được liệt kê, đó là

Nghiệm hợp lệ	$S(i)$	$c(S(i))$
$S(50)$	$\{u_2, u_3, u_5, u_6\}$	30
$S(350)$	$\{u_1, u_2, u_3, u_5, u_6\}$	50

Trong các nghiệm hợp lệ được liệt kê, nghiệm có chi phí nhỏ nhất là $S(50)$. Do đó, $S = S(50) = \{u_2, u_3, u_5, u_6\}$. Cuối cùng, vì $V_0 = \emptyset$, do vậy

$$S_{\text{final}} = S \cup V_0 = \{u_2, u_3, u_5, u_6\}.$$

Các chặn lý thuyết và độ phức tạp của thuật toán AdaptMCSC được phát biểu trong Định lý sau.

Định lý 2.2. Với $\lambda \in (0, 1)$ và $\epsilon \in (0, 1/2)$, Thuật toán 2.5 trả về một nghiệm S thỏa mãn các tính chất sau:

- đạt được tỷ lệ xấp xỉ tiêu chuẩn kép theo kỳ vọng là

$$\left((1 + \epsilon) \left(1 + \ln \left(\frac{1}{\lambda} \right) \right), (1 - \lambda) \right)$$

$$\text{Cụ thể } g(S) \geq (1 - \lambda)T \text{ và } \mathbb{E}[c(S)] \leq (1 + \epsilon) \left(1 + \ln \left(\frac{1}{\lambda} \right) \right) \text{opt.}$$

- độ phức tạp truy vấn:

$$O\left(\frac{n^2}{\epsilon^3} \log\left(\frac{1}{\lambda}\right) \log^2\left(\frac{n}{\epsilon}\right)\right).$$

- độ phức tạp song song (adaptive complexity):

$$O\left(\frac{1}{\epsilon^2} \log\left(\frac{1}{\lambda}\right) \log\left(\frac{n}{\epsilon}\right)\right).$$

Ngoài ra, nếu tại dòng 10 của Thuật toán 2.5 ta sử dụng THRESHSEQBi thay cho THRESHSEQ trong thủ tục AdaptMCSCopt, thì ta thu được các chặn độ phức tạp sau:

- độ phức tạp truy vấn:

$$O\left(\log\left(\frac{1}{\lambda}\right) \frac{n}{\epsilon^3} \log(n) \log^2\left(\frac{n}{\epsilon}\right)\right).$$

- độ phức tạp song song (adaptive complexity):

$$O\left(\frac{1}{\epsilon^2} \log\left(\frac{1}{\lambda}\right) \log(n) \log\left(\frac{n}{\epsilon}\right)\right).$$

Chứng minh. Từ chứng minh của Định lý 13 trong [19], ta có $c(u_j) \leq \text{opt} \leq jc(u_j)$ và $c(V_0) < \epsilon' \text{opt}$ và $f(V_0) < T$, tức là V_0 không phải là một nghiệm hợp lệ của bài toán. Trước hết, gọi j' là chỉ số của phần tử có chi phí lớn nhất trong lời giải tối ưu O , tức là

$$c(v_{j'}) = \max\{c(v_i) : v_i \in O\}.$$

Ta chứng minh $c(v_{j'}) \geq c(v_j)$. Giả sử ngược lại rằng $c(v_{j'}) < c(v_j)$. Do các phần tử đã được sắp xếp theo chi phí, ta có $j' < j$, đồng thời $O \subseteq \{v_1, \dots, v_{j'}\}$. Tính đơn điệu của g suy ra $g(\{v_1, \dots, v_{j'}\}) \geq g(O) \geq T$, mâu thuẫn với tính tối tiểu của j . Do đó,

$$c(v_j) \leq c(v_{j'}) \leq c(O) = \text{opt}.$$

Mặt khác, theo định nghĩa của j , tập $\{v_1, \dots, v_j\}$ là một nghiệm khả thi. Tính tối ưu của O và thứ tự sắp xếp các chi phí cho ta $\text{opt} = c(O) \leq \sum_{i=1}^j c(v_i) \leq jc(v_j)$. Vậy

$$c(v_j) \leq \text{opt} \leq jc(v_j).$$

Ký hiệu S'^* là nghiệm tối ưu của bài toán MCSC trên thể hiện (V', g', T') và $\text{opt}' = c(S'^*)$. Theo cách chọn của thuật toán, ta có

$$\frac{\epsilon' c(u_j)}{n} \leq c'_{\min} \leq \text{opt}' \leq c'_{\max} \leq nc(u_j).$$

Do vậy

$$c(V_0) \leq |V_0|c'_{\min} \leq \epsilon' c(u_j) \leq \epsilon' \text{opt}.$$

Đặt $v_i = \frac{c'_{\min}}{(1-\epsilon')^i} = \frac{\epsilon' c(u_j)}{n(1-\epsilon')^i}$. Khi i chạy từ 0 đến $\lceil \log 1/(1-\epsilon')(n^3/\epsilon') \rceil$, ta có

$$\frac{\epsilon' c(u_j)}{n} \leq c'_{\min} \leq v_i = \frac{c'_{\min}}{(1-\epsilon')^i} = \frac{\epsilon' c(u_j)}{n(1-\epsilon')^i} \leq nc(u_j). \quad (2.27)$$

Do đó, tồn tại một vòng lặp i của vòng lặp chính sao cho $v_i(1-\epsilon') \leq \text{opt}' < v_i$. Theo Định lý 2.1, ta suy ra

$$\mathbb{E}[c(S(i))] \leq (1 + \frac{19}{4}\epsilon')\text{opt}'.$$

Nhờ tính đơn điệu tăng và submodular của hàm g , ta có

$$\begin{aligned} g'(S^*) &= g_{V_0}(S^*) \quad (\text{Do định nghĩa } g_{V_0}) \\ &= g(S^* \cup V_0) - g(V_0) \\ &\geq T - g(V_0) \geq T' \quad (\text{Do tính đơn điệu của } g). \end{aligned}$$

Suy ra S^* là một nghiệm hợp lệ của thể hiện (V', g', T') . Do đó, $\text{opt}' \leq \text{opt}$. Mặt khác, theo Định lý 2.1 và cách lựa chọn nghiệm ứng viên S , ta có

$$\begin{aligned} \mathbb{E}[c(S \cup V_0)] &= \mathbb{E}[c(V_0) + c(S)] = \mathbb{E}[c(V_0)] + \mathbb{E}[c(S)] \\ &\leq \epsilon' \text{opt} + (1 + \frac{19}{4}\epsilon')(1 + \log(\frac{1}{\lambda}))\text{opt}' \\ &< (1 + \frac{23}{4}\epsilon')(1 + \log(\frac{1}{\lambda}))\text{opt}. \end{aligned}$$

Đồng thời, từ $g_{V_0}(S) \geq T'(1-\lambda)$ suy ra

$$\begin{aligned} g(S \cup V_0) &= g_{V_0}(S) + g(V_0) \\ &\geq T'(1-\lambda) + g(V_0) \\ &\geq (T - g(V_0))(1-\lambda) + g(V_0) \\ &\geq T(1-\lambda). \end{aligned}$$

Thuật toán sử dụng n truy vấn và một vòng song song để tìm chỉ số j tại dòng 3 của Thuật toán 2.5. Sau đó, vòng lặp chính (các dòng 8–11) được thực hiện trong $\log_{(1-\epsilon')^{-1}}(n^3/\epsilon') = O(\log(n/\epsilon')/\epsilon')$ vòng lặp, trong đó mỗi lần lặp gọi Thuật toán 2.4 như một thủ tục con. Kết hợp với Bổ đề 2.4, ta thu được độ phức tạp truy vấn của Thuật toán 2.5:

$$\begin{aligned} n + \log_{(1-\epsilon')^{-1}}(\frac{n^3}{\epsilon'}) \frac{n^2}{\epsilon'^2} \log(\frac{1}{\lambda}) \log(\frac{nc'_{\max}}{c'_{\min}}) &\leq n + \frac{n^2}{\epsilon'^3} \log(\frac{n^3}{\epsilon'}) \log(\frac{1}{\lambda}) \log(\frac{n^2 j}{\epsilon'}) \\ &\leq n + \frac{n^2}{\epsilon'^3} \log(\frac{n^3}{\epsilon'}) \log(\frac{1}{\lambda}) \log(\frac{n^3}{\epsilon'}) \\ &= O(\log(\frac{1}{\lambda}) \frac{n^2}{\epsilon'^3} \log^2(\frac{n}{\epsilon'})). \end{aligned}$$

Độ phức tạp song song (adaptive complexity) của thuật toán là

$$\begin{aligned} 1 + \frac{1}{\epsilon'^2} \log\left(\frac{1}{\lambda}\right) \log\left(\frac{nc_{\max}}{c_{\min}}\right) &= 1 + \frac{1}{\epsilon'^2} \log\left(\frac{1}{\lambda}\right) \log\left(\frac{n^2 j}{\epsilon'}\right) \\ &= O\left(\frac{1}{\epsilon'^2} \log\left(\frac{1}{\lambda}\right) \log\left(\frac{n}{\epsilon}\right)\right). \end{aligned}$$

Nếu sử dụng THRESHSEQBi thay cho THRESHSEQ trong AdaptMCSCopt, thì độ phức tạp truy vấn của Thuật toán 2.5 là

$$\begin{aligned} n + \log_{(1-\epsilon)^{-1}}\left(\frac{n^3}{\epsilon'}\right) \frac{n \log(n)}{\epsilon'^2} \log\left(\frac{1}{\lambda}\right) \log\left(\frac{nc'_{\max}}{c'_{\min}}\right) \\ = O\left(\log\left(\frac{1}{\lambda}\right) \frac{n}{\epsilon^3} \log(n) \log^2\left(\frac{n}{\epsilon}\right)\right). \end{aligned}$$

và độ phức tạp song song (adaptive complexity) là

$$1 + \frac{1}{\epsilon'^2} \log\left(\frac{1}{\lambda}\right) \log(n) \log\left(\frac{nc_{\max}}{c_{\min}}\right) = O\left(\frac{1}{\epsilon'^2} \log\left(\frac{1}{\lambda}\right) \log(n) \log\left(\frac{n}{\epsilon}\right)\right).$$

□

Định lý 2.2 có thể suy ra một số nhận xét quan trọng sau:

Thứ nhất, định lý nêu kết quả về mặt lý thuyết mà thuật toán AdaptMCSC đạt được như sau: giá trị hàm mục tiêu đạt ít nhất $T(1 - \lambda)$ trong khi chi phí kỳ vọng chỉ lớn hơn nghiệm tối ưu một hệ số logarithm theo $1/\lambda$. Điều này cho thấy thuật toán vẫn duy trì được chất lượng nghiệm gần tối ưu trong khi chỉ cần đạt ngưỡng mục tiêu gần đúng.

Thứ hai, các chặn về độ phức tạp truy vấn và độ phức tạp song song cho thấy thuật toán có thể xử lý các bài toán với kích thước dữ liệu lớn một cách hiệu quả. Đặc biệt, độ phức tạp song song chỉ phụ thuộc đa thức theo $1/\epsilon$ và logarit theo kích thước đầu vào, điều này cho phép thuật toán có khả năng thực thi tốt trong các môi trường tính toán song song hoặc phân tán.

Thứ ba, biến thể sử dụng thủ tục THRESHSEQBi cho thấy có thể cải thiện đáng kể độ phức tạp truy vấn từ bậc n^2 xuống bậc gần tuyến tính theo n mà vẫn giữ nguyên các đảm bảo về chất lượng nghiệm. Điều này mở ra khả năng áp dụng thuật toán cho các bài toán quy mô lớn trong thực tế.

2.4. THỰC NGHIỆM VÀ KẾT QUẢ

Trong phần này, LA trình bày các kết quả thực nghiệm nhằm đánh giá hiệu quả của thuật toán AdaptMCSC so với các phương pháp hiện có cho bài toán *IT* [83, 81], một ứng dụng tiêu biểu của bài toán MCSC (được mô tả tại mục 2.1.), trong đó hàm ảnh hưởng g là hàm submodular.

Các thí nghiệm được xây dựng để đánh giá đồng thời cả chất lượng nghiệm và hiệu năng tính toán của thuật toán đề xuất trong nhiều bối cảnh dữ liệu khác nhau. Cụ thể, các thuật toán được so sánh theo các tiêu chí sau:

- **Chất lượng nghiệm**, đo bằng tổng chi phí của tập lời giải cần thiết để đạt được ngưỡng ảnh hưởng yêu cầu.
- **Độ phức tạp song song** (adaptive complexity or number of sequential iterations), thể hiện số vòng truy cập dữ liệu theo mô hình tính toán song song;
- **Thời gian chạy thực tế** của thuật toán trong các thí nghiệm (running time).

Các phương pháp được lựa chọn để so sánh bao gồm những thuật toán tiên tiến đạt kết quả tốt về cả tỷ lệ xấp xỉ lẫn độ phức tạp song song, được liệt kê trong Bảng 2.3. Nhìn chung, chúng đại diện cho hai hướng tiếp cận chủ đạo trong việc giải bài toán MCSC: các thuật toán tham lam truyền thống và các thuật toán song song hiện đại. Mô tả cụ thể như sau:

- **Thuật toán tham lam (Greedy algorithm) [64]**: Đây là phương pháp đạt tỷ lệ xấp xỉ hai tiêu chí tốt nhất hiện nay cho bài toán MCSC trong trường hợp tổng quát, khi hàm g nhận giá trị thực và chi phí các phần tử là tùy ý. Thuật toán thuộc nhóm tiếp cận tham lam cổ điển, trong đó ở mỗi bước chọn phần tử có tỷ lệ lợi ích trên chi phí lớn nhất. Tuy nhiên, do quá trình lựa chọn mang tính tuần tự, độ phức tạp song song của thuật toán là $\Omega(|S|)$ và có thể lên đến $O(n)$ trong trường hợp xấu nhất.
- **Thuật toán MinSMC của Ran và cộng sự [19]**: Đây là một trong những phương pháp song song hiệu quả cho bài toán MCSC, với số vòng lặp tuần tự được giảm đáng kể so với các thuật toán tham lam truyền thống. Dẫu vậy, thuật toán này yêu cầu hàm g nhận giá trị nguyên, điều này có thể hạn chế khả năng áp dụng trong các bài toán thực tế khi hàm lợi ích có giá trị thực.

Việc so sánh với các phương pháp trên cho phép đánh giá toàn diện hiệu quả của thuật toán AdaptMCSC, không chỉ về chất lượng nghiệm mà còn về khả năng giảm số vòng lặp tuần tự và chi phí tính toán trong thực tế.

2.4.1. Thiết lập thực nghiệm

2.4.1.1. Dữ liệu

LA sử dụng các OSN từ cơ sở dữ liệu về các OSN nổi tiếng của nhóm nghiên cứu ở đại học Stanford SNAP (Stanford Large Network Dataset Collection) [92] bao gồm

các mạng EmailEuCore ($n = 1,005$), EgoFacebook ($n = 4,039$), NetHEPT ($n = 15,233$) trong đó n là số đỉnh trong mạng.

Bảng 2.3: Bảng thống kê các dữ liệu được sử dụng

Tên mạng	Nguồn	Số đỉnh (n)	Số cạnh (m)
EmailEuCore	SNAP	1,005	25,571
EgoFacebook	SNAP	4,039	88,234
NetHEPT	SNAP	15,233	62,774

2.4.1.2. Môi trường thực nghiệm và tham số

LA triển khai song song các thuật toán bằng thư viện OpenMP, với toàn bộ mã nguồn được cài đặt bằng ngôn ngữ C++. Các thí nghiệm được thực hiện trên hệ thống tính toán hiệu năng cao (HPC) với cấu hình: partition = large, số luồng CPU là 64, sử dụng 2 node, và bộ nhớ tối đa đạt 3,073 GB. Cấu hình máy tính CPU Intel Xeon Gold 6226R. Một core có xung cơ bản 2.90 GHz.

Các tham số được thiết lập thống nhất như sau: $\lambda = \epsilon = 0.1$ cho tất cả các thuật toán. Đối với mô hình lan truyền thông tin IC, xác suất trên mỗi cạnh $e = (u, v)$ của đồ thị được khởi tạo theo công thức $p_e = \frac{1}{d_v}$, trong đó d_v là bậc vào của đỉnh v , phù hợp với cách thiết lập tiêu chuẩn trong các nghiên cứu trước [1, 81]. Chi phí của mỗi phần tử được

Bảng 2.4: Thiết lập các tham số thực nghiệm.

Ký hiệu	Ý nghĩa	Giá trị
ϵ, λ	Tham số sai số của các thuật toán	0.1 [1, 64]
p_e	Xác suất lan truyền trên cạnh $e = (u, v)$ trong mô hình IC	$p_e = \frac{1}{d_v}$, trong đó d_v là bậc của đỉnh v [1, 81]
$c(u)$	Chi phí của phần tử u	$U(0, 1)$ [93]
T	Ngưỡng phủ cần đạt	$\{0.1g(V), 0.2g(V), 0.3g(V), 0.4g(V), 0.5g(V)\}$
MC	Số lần mô phỏng Monte-Carlo để ước lượng hàm mục tiêu	1000 [1, 43, 94]

khởi tạo ngẫu nhiên trong $(0, 1)$, ngưỡng T được chọn trong tập

$$T \in \{0.1g(V), 0.2g(V), 0.3g(V), 0.4g(V), 0.5g(V)\}$$

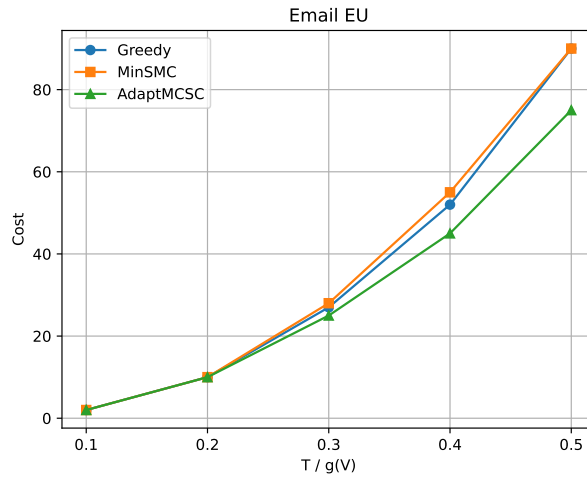
để đánh giá toàn diện ảnh hưởng của các thuật toán theo các nghiên cứu trước [93, 81]. Với bài toán IT, luận án sử dụng phương pháp Monte-Carlo với **1.000** lần mô phỏng để ước lượng giá trị của hàm mục tiêu. Đây là một thiết lập tiêu chuẩn và được sử dụng rộng rãi trong các nghiên cứu về lan truyền ảnh hưởng trên OSN [1, 43]. Số lượng mô phỏng này đủ lớn để giảm đáng kể sai số ngẫu nhiên trong quá trình ước lượng, đồng thời bảo đảm tính ổn định, độ tin cậy và khả năng tái lập của các kết quả thực nghiệm. Các thuật toán ngẫu nhiên được thực hiện 10 lần và thống kê kết quả trung bình.

2.4.2. Kết quả và thảo luận

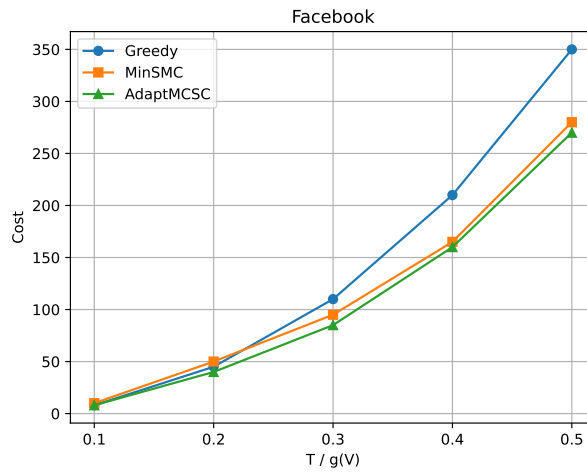
2.4.2.1. Chất lượng lời giải

Hình 2.1 thể hiện tổng chi phí của các lời giải do các thuật toán tạo ra trên ba bộ dữ liệu Email EU, Facebook và NetHEPT tương ứng với các mức ngưỡng T . Nhìn chung, hai thuật toán MinSMC và AdaptMCSC cho chất lượng nghiệm khá tương đồng và tốt hơn so với thuật toán tham ở đa số các trường hợp, vốn thường được sử dụng làm chuẩn so sánh trong các nghiên cứu trước.

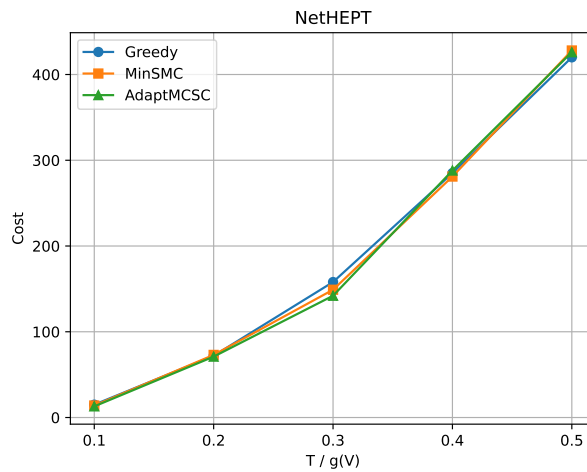
Cụ thể, trên phần lớn các bộ dữ liệu và các giá trị T , sự khác biệt về chi phí giữa các thuật toán là không đáng kể. Chẳng hạn, với bộ dữ liệu Facebook tại $T = 0.5g(V)$, chi phí thu được từ GREEDY, MinSMC và AdaptMCSC lần lượt là 350, 280 và 270. Tương tự, trên bộ dữ liệu NetHEPT, các kết quả cũng rất sát nhau; ví dụ tại $T = 0.3g(V)$, chi phí tương ứng là 158, 149 và 142. Điều này cho thấy các phương pháp đề xuất vẫn duy trì được chất lượng nghiệm tốt hơn MinSMC và AdaptMCSC trong hầu hết các trường hợp. Bảng 2.5 thống kê sự cải thiện chất lượng lời giải (%) giữa AdaptMCSC với MinSMC và tham lam. Những khác biệt về chi phí giữa các thuật toán là tương đối nhỏ, phù hợp với các kết quả phân tích lý thuyết đã trình bày trước đó. Cụ thể, do các thuật toán đều đạt được các tỷ lệ xấp xỉ gần tương đương, chất lượng nghiệm thu được trong thực nghiệm cũng khá sát nhau. Mặc dù vậy, AdaptMCSC thường cho chi phí thấp hơn hoặc cạnh tranh với các phương pháp còn lại trên hầu hết các bộ dữ liệu. Kết quả này cho thấy AdaptMCSC và MinSMC có khả năng duy trì chất lượng nghiệm tương đương với phương pháp tham lam, đồng thời mang lại lợi thế đáng kể về hiệu quả tính toán, như sẽ được thể hiện rõ hơn thông qua các kết quả về số vòng lặp và thời gian thực thi ở các phần tiếp theo.



(a) Email EU



(b) Facebook



(c) NetHEPT

Hình 2.1: So sánh tổng chi phí của các thuật toán tham lam, MinSMC và AdaptMCSC theo các giá trị ngưỡng T .

Bảng 2.5: So sánh tỷ lệ cải thiện của AdaptMCSC so với các thuật toán cơ sở, trong đó
 $\% \text{ cải thiện} = \frac{\text{Cost}(\text{Baseline}) - \text{Cost}(\text{Adapt})}{\text{Cost}(\text{Baseline})} \times 100$.

Dataset	AdaptMCSC vs. Greedy	AdaptMCSC vs. MinSMC
Email EU	0% ~ 17%	0% ~ 16%
Facebook	8% ~ 23%	0% ~ 6%
NetHEPT	0% ~ 3%	-1% ~ 2%
Overall	0% ~ 23%	-1% ~ 16%

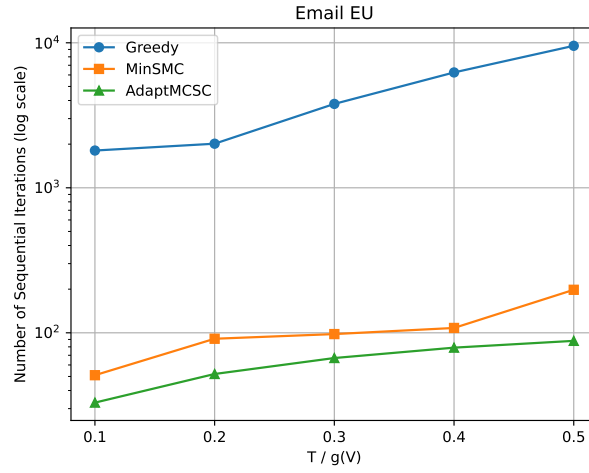
2.4.2.2. Số vòng lặp tuần tự

Hình 2.2 trình bày số vòng lặp tuần tự của các thuật toán GREEDY, MinSMC và AdaptMCSC trên ba bộ dữ liệu Email EU, Facebook và NetHEPT với các giá trị ngưỡng T . Nhìn chung, các thuật toán MinSMC và AdaptMCSC yêu cầu số vòng lặp tuần tự ít hơn đáng kể so với thuật toán tham lam truyền thống.

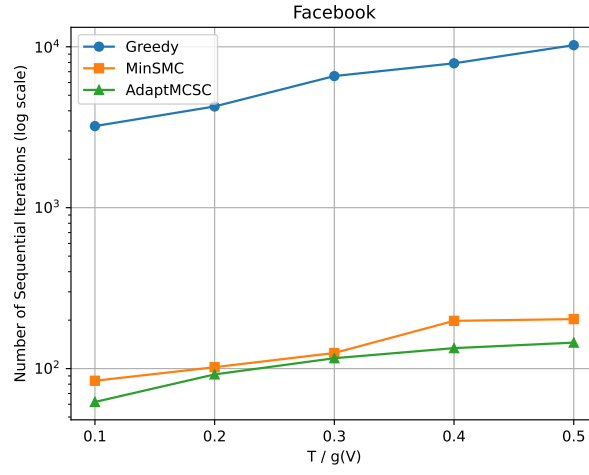
Cụ thể, trên bộ dữ liệu Email EU, thuật toán GREEDY cần từ 1,807 đến 9,532 vòng lặp khi T tăng từ $0.1g(V)$ đến $0.5g(V)$, trong khi MinSMC và AdaptMCSC chỉ cần khoảng 51–198 và 33–88 vòng lặp tương ứng. Xu hướng tương tự cũng được quan sát trên các bộ dữ liệu lớn hơn như Facebook và NetHEPT. Chẳng hạn, trên NetHEPT với $T = 0.5g(V)$, GREEDY cần tới 56,290 vòng lặp, trong khi MinSMC và AdaptMCSC chỉ cần 307 và 187 vòng lặp tương ứng.

Một điểm đáng chú ý là các thuật toán MinSMC và AdaptMCSC được xây dựng nhằm tận dụng hiệu quả năng lực tính toán song song. Trong mỗi vòng lặp, nhiều phép đánh giá hàm lợi ích có thể được thực hiện đồng thời trên các luồng xử lý khác nhau. Vì vậy, không chỉ số vòng lặp tuần tự được giảm đáng kể so với GREEDY, mà chi phí thực tế của từng vòng lặp cũng được cải thiện rõ rệt khi triển khai trên các hệ thống đa lõi hoặc môi trường song song.

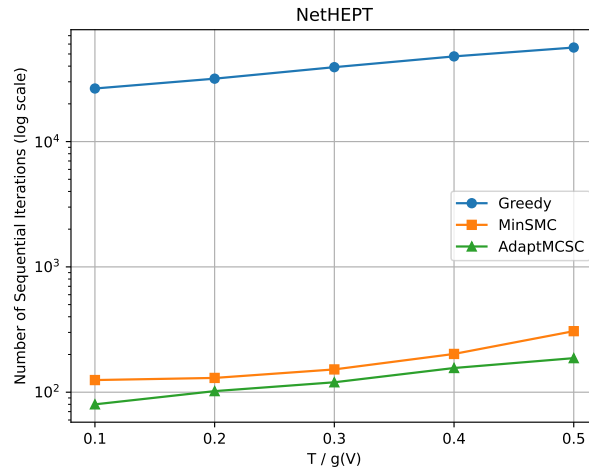
Từ các kết quả thực nghiệm, có thể thấy rằng MinSMC và AdaptMCSC vừa duy trì chất lượng nghiệm tương đương với phương pháp tham lam, vừa giảm đáng kể số vòng lặp tuần tự. Đáng chú ý, thuật toán AdaptMCSC thường đạt số vòng lặp nhỏ nhất trong phần lớn các thiết lập thực nghiệm. Quan sát này phù hợp với các phân tích lý thuyết đã trình bày trước đó, khi AdaptMCSC được thiết kế nhằm giảm độ phức tạp song song (adaptive complexity) trong quá trình lựa chọn phần tử. Nhờ cơ chế chọn theo ngưỡng kết hợp với việc đánh giá đồng thời nhiều phần tử trong mỗi vòng lặp, thuật toán có thể cắt giảm đáng kể số bước tuần tự so với các phương pháp tham lam truyền thống.



(a) Email EU



(b) Facebook



(c) NetHEPT

Hình 2.2: Số vòng lặp tuần tự của các thuật toán GREEDY, MinSMC và AdaptMCSC trên ba bộ dữ liệu Email EU, Facebook và NetHEPT với các giá trị ngưỡng $T \in \{0.1g(V), 0.2g(V), \dots, 0.5g(V)\}$. Trục tung được biểu diễn theo thang logarit.

2.4.2.3. Thời gian chạy

Hình 2.3 trình bày thời gian chạy của các thuật toán trên ba bộ dữ liệu Email EU, Facebook và NetHEPT với các giá trị ngưỡng $T \in \{0.1g(V), 0.2g(V), \dots, 0.5g(V)\}$. Nhìn chung, thời gian chạy của các thuật toán có xu hướng tỷ lệ với số vòng lặp tuần tự được trình bày trong Hình 2.2. Điều này là hợp lý vì trong mỗi vòng lặp, các phép đánh giá hàm lợi ích của các phần tử thường được thực hiện đồng thời trên nhiều luồng xử lý, do đó chi phí tính toán chủ yếu phụ thuộc vào số vòng lặp tuần tự của thuật toán.

Tuy nhiên, trong thực tế, thời gian chạy không hoàn toàn tỷ lệ tuyến tính với số vòng lặp. Nguyên nhân là do số lượng luồng xử lý của hệ thống tính toán là hữu hạn, dẫn đến việc một số phép tính vẫn phải được thực hiện tuần tự hoặc chia sẻ tài nguyên tính toán. Ngoài ra, các chi phí phụ trợ như quản lý bộ nhớ và đồng bộ hóa giữa các luồng cũng có thể ảnh hưởng đến thời gian thực thi tổng thể của thuật toán.

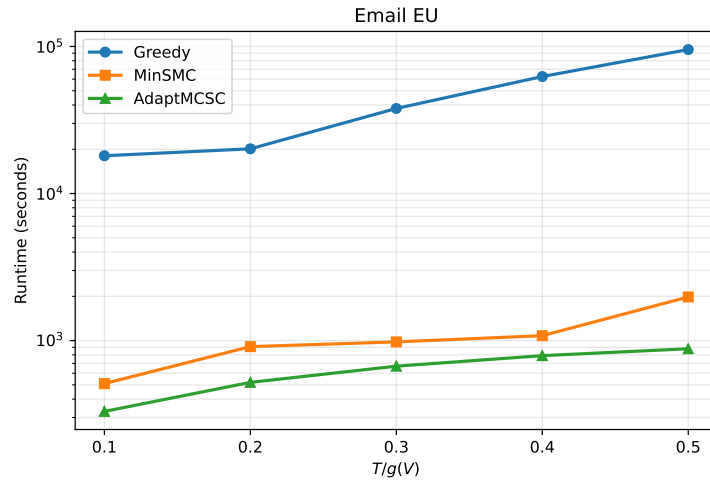
Quan sát cụ thể các kết quả thực nghiệm cho thấy thuật toán Tham lam có thời gian chạy lớn nhất trên cả ba bộ dữ liệu, do số vòng lặp tuần tự rất lớn. Chẳng hạn, trên bộ dữ liệu NetHEPT với $T = 0.5g(V)$, thời gian chạy của thuật toán tham lam lên tới 562,900 giây, trong khi MinSMC và AdaptMCSC chỉ cần lần lượt 3,070 và 1,870 giây. Điều này cho thấy AdaptMCSC nhanh hơn khoảng tới 301 lần so với tham lam và tới 1.64 lần so với MinSMC. Tương tự, trên các bộ dữ liệu Email EU và Facebook, hai thuật toán MinSMC và AdaptMCSC đều cho thấy sự cải thiện đáng kể về thời gian chạy so với phương pháp tham lam.

Thuật toán AdaptMCSC do tác giả đề xuất thường đạt thời gian chạy nhỏ nhất trong hầu hết các thiết lập thực nghiệm. Điều này phản ánh trực tiếp ưu thế của thuật toán trong việc giảm độ phức tạp song song của quá trình lựa chọn phần tử, từ đó làm giảm số vòng lặp tuần tự cần thiết.

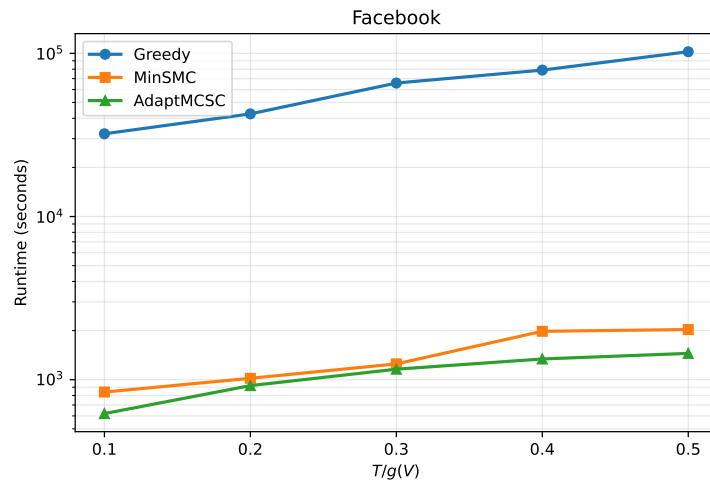
AdaptMCSC thể hiện ưu thế rõ về thời gian chạy trên NetHEPT, đặc biệt khi ngưỡng yêu cầu cao, đồng thời duy trì chất lượng lời giải cạnh tranh. Tuy nhiên, mức cải thiện về chi phí so với MinSMC còn tương đối nhỏ, khoảng 3,6% trên Facebook và 4,7% trên NetHEPT tại các thiết lập đã nêu. Hiệu quả tăng tốc phụ thuộc vào tài nguyên tính toán song song và có thể giảm trên dữ liệu nhỏ hoặc khi chi phí đánh giá hàm mục tiêu thấp. Với phạm vi thực nghiệm trên ba bộ dữ liệu, chưa đủ cơ sở để xác định bộ dữ liệu mà thuật toán hoạt động yếu hoặc khái quát hiệu quả cho các cấu trúc mạng khác.

2.5. KẾT LUẬN CHƯƠNG

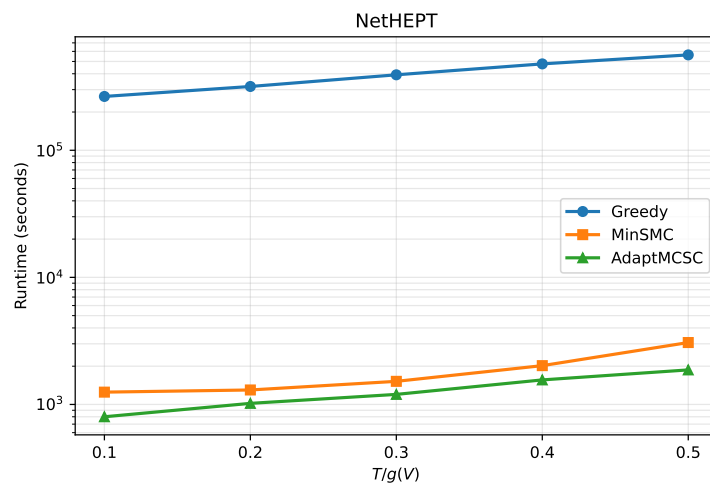
Trong chương này, luận án tập trung nghiên cứu bài toán MCSC, một lớp bài toán tối ưu có nhiều ứng dụng trong trí tuệ nhân tạo, nghiên cứu tác nghiệp và phân tích dữ



(a) Email EU



(b) Facebook



(c) NetHEPT

Hình 2.3: Thời gian chạy của các thuật toán GREEDY, MinSMC và AdaptMCSC.

liệu. Mục tiêu đặt ra là xây dựng các thuật toán vừa có kết quả lý thuyết tốt về chất lượng nghiệm, vừa tối ưu chi phí tính toán, đặc biệt là độ phức tạp song song, nhằm đáp ứng yêu cầu của các bài toán quy mô lớn.

Về phương diện lý thuyết, luận án đề xuất thuật toán AdaptMCSC cho bài toán MCSC với hàm g là submodular, đơn điệu tăng và nhận giá trị thực. Thuật toán này đạt tỷ lệ xấp xỉ tiêu chuẩn kép tiệm cận các kết quả tốt nhất hiện nay, đồng thời cải thiện đáng kể độ phức tạp truy vấn cũng như độ phức tạp song song, nhất là khi loại bỏ được sự phụ thuộc vào tham số T trong quá trình phân tích. Bên cạnh đó, biến thể sử dụng thủ tục THRESHSEQBi còn giúp đưa độ phức tạp truy vấn xuống gần tuyến tính theo kích thước đầu vào.

Các kết quả thực nghiệm trên những bộ dữ liệu thực cũng củng cố các phân tích lý thuyết đã nêu. Cụ thể, thuật toán AdaptMCSC vẫn giữ được chất lượng nghiệm tương đương với các phương pháp hiện có, trong khi giảm đáng kể số vòng lặp tuần tự và thời gian thực thi, đặc biệt trên các tập dữ liệu lớn.

Mặc dù đạt được những kết quả khả quan, các thuật toán đề xuất trong chương vẫn được xây dựng dưới giả thiết hàm mục tiêu là submodular đơn điệu tăng và dữ liệu được truy cập theo mô hình tĩnh. Do đó, hiệu quả lý thuyết của các thuật toán chưa được bảo đảm đối với các lớp hàm tổng quát hơn hoặc trong các môi trường mà dữ liệu thay đổi liên tục theo thời gian. Bên cạnh đó, mặc dù độ phức tạp song song đã được cải thiện đáng kể về mặt lý thuyết, việc triển khai trên các hệ thống phân tán hoặc các nền tảng xử lý dữ liệu cực lớn vẫn cần được nghiên cứu sâu hơn để đánh giá đầy đủ khả năng mở rộng trong thực tế.

Trong thời gian tới, một hướng nghiên cứu đáng quan tâm là tiếp tục cải thiện các chặn lý thuyết của thuật toán, đặc biệt theo hướng giảm hơn nữa sự phụ thuộc vào tham số ϵ . Ngoài ra, việc mở rộng các kỹ thuật đề xuất sang các mô hình dữ liệu động, mô hình luồng dữ liệu, cũng như các kiến trúc tính toán song song và phân tán quy mô lớn là những hướng nghiên cứu tiềm năng nhằm nâng cao khả năng ứng dụng của các thuật toán trong các hệ thống dữ liệu hiện đại.

CHƯƠNG 3

BÀI TOÁN PHỦ k -SUBMODULAR VỚI CHI PHÍ TỐI THIỂU

Như đã trình bày ở chương trước, bài toán *Phủ submodular với chi phí tối thiểu* (MCSC) là một bài toán CO quan trọng, đã được nghiên cứu sâu rộng trong các lĩnh vực như nghiên cứu tác nghiệp, trí tuệ nhân tạo và phân tích dữ liệu với nhiều ứng dụng quan trọng như học chủ động, tóm tắt dữ liệu, tối ưu hóa phúc lợi xã hội và lan truyền ảnh hưởng trên OSN [76, 1, 79, 95].

Tuy nhiên, trong nhiều ứng dụng thực tế, khi một phần tử được lựa chọn vào lời giải, nó không chỉ đơn thuần được chọn hay không chọn mà còn có thể được gán vào một trong nhiều loại quyết định khác nhau. Điều này dẫn tới mô hình ra quyết định đa lựa chọn, trong đó mỗi phần tử được gán duy nhất vào một trong k phương án loại trừ lẫn nhau. Mô hình này xuất hiện tự nhiên trong nhiều bài toán ứng dụng quan trọng như tối đa hóa ảnh hưởng theo nhiều chủ đề trong OSN, bố trí cảm biến đa loại, lựa chọn đặc trưng đa lớp trong học máy và tối đa hóa độ bao phủ thông tin [11, 6, 96, 73, 97, 98, 99, 95]. Để mô hình hóa các bài toán này, khái niệm *hàm k -submodular* đã được giới thiệu như một sự mở rộng tự nhiên của hàm submodular cổ điển [99]. Hàm k -submodular cho phép biểu diễn các quyết định đa lựa chọn trong khi vẫn bảo toàn nhiều tính chất cấu trúc quan trọng của SO. Từ đó, bài toán *Phủ k -submodular* được hình thành như một mở rộng tự nhiên của bài toán Phủ submodular, trong đó hàm mục tiêu thỏa mãn tính chất k -submodular thay vì submodular thông thường. Bài toán này đã thu hút nhiều sự quan tâm nghiên cứu gần đây do vai trò quan trọng trong các bài toán tối ưu hóa trên dữ liệu có cấu trúc đa nhân hoặc đa loại [96, 97, 95].

Trong chương này, luận án nghiên cứu bài toán Phủ k -submodular trong CO, xét hai thiết lập chính là chi phí đồng nhất và chi phí tổng quát. Với mỗi trường hợp, các TXXX hiệu quả được đề xuất kèm theo phân tích về bảo đảm lý thuyết và độ phức tạp tính toán. Các thuật toán cũng được đánh giá thực nghiệm trên dữ liệu thực tế để kiểm chứng hiệu năng. Một số kết quả chính của chương đã được công bố trên tạp chí **Asia-Pacific Journal of Operational Research (SCIE/Q3)** và tại hội nghị **International Symposium on Information and Communication Technology (SoICT)** năm 2025.

3.1. ĐẶT VẤN ĐỀ VÀ PHÁT BIỂU BÀI TOÁN

Bài toán tối ưu hàm k -submodular là một bài toán quan trọng trong nghiên cứu tác nghiệp và trí tuệ nhân tạo, do có khả năng mô hình hóa các quá trình ra quyết định đa lựa chọn, trong đó mỗi phần tử phải được gán vào một trong k phương án loại trừ lẫn nhau. Cấu trúc này đóng vai trò nền tảng trong nhiều ứng dụng thực tiễn, chẳng hạn như tối đa hóa ảnh hưởng với k chủ đề trong mạng xã hội [11, 6, 75, 76, 77, 78], bố trí cảm biến đa loại [11, 6, 75, 77], lựa chọn đặc trưng đa lớp [79], và tối đa hóa độ bao phủ thông tin [75, 74]. Không chỉ giới hạn ở các ví dụ trên, các hàm k -submodular còn cung cấp một sự mở rộng giàu tính khái quát của tính submodular cổ điển, cho phép xây dựng các mô hình có cơ sở lý thuyết chặt chẽ cho những bài toán liên quan đến cạnh tranh, đa dạng hóa, phân cụm và dự đoán có cấu trúc. Trong bối cảnh các hệ thống trí tuệ nhân tạo hiện đại ngày càng đòi hỏi việc ra quyết định đồng thời trên nhiều nhóm, nhiều nhãn và nhiều phương thức dữ liệu khác nhau, việc nghiên cứu các thuật toán hiệu quả cho bài toán tối ưu k -submodular trở nên thiết yếu nhằm đảm bảo khả năng mở rộng và duy trì các bảo đảm lý thuyết mạnh mẽ.

3.1.1. Hàm k -submodular

Cho tập nền $V = \{e_1, e_2, \dots, e_n\}$ và một số nguyên k . Ký hiệu $[k] = \{1, 2, \dots, k\}$ và đặt

$$(k+1)^V = \{(V_1, V_2, \dots, V_k) \mid V_i \subseteq V, \forall i \in [k], V_i \cap V_j = \emptyset, \forall i \neq j\}$$

là họ các bộ gồm k tập con đôi một rời nhau của V , được gọi là một **k -tập** (k -set).

Với $\mathbf{x} = (X_1, X_2, \dots, X_k) \in (k+1)^V$, ta định nghĩa $\text{supp}_i(\mathbf{x}) = X_i$, $\text{supp}(\mathbf{x}) = \bigcup_{i \in [k]} X_i$. Tập X_i được gọi là **tập thứ i của $\mathbf{x}$** . Ngoài ra, ký hiệu $\mathbf{0} = (\emptyset, \dots, \emptyset)$ là k -tập rỗng.

Nếu $e \in X_i$ thì ta viết $\mathbf{x}(e) = i$ và nói rằng i là **vị trí** của phần tử e trong $\mathbf{x}$; ngược lại, nếu $e \notin \text{supp}(\mathbf{x})$ thì $\mathbf{x}(e) = 0$. Việc thêm một phần tử $e \notin \text{supp}(\mathbf{x})$ vào tập X_i được ký hiệu bởi $\mathbf{x} \sqcup (e, i)$. Ta cũng có thể biểu diễn

$$\mathbf{x} = \{(e_1, i_1), (e_2, i_2), \dots, (e_t, i_t)\},$$

trong đó $e_j \in \text{supp}(\mathbf{x})$ và $i_j = \mathbf{x}(e_j)$ với mọi $1 \leq j \leq t$. Trong trường hợp $X_i = \{e\}$ và $X_j = \emptyset$ với mọi $j \neq i$, khi đó $\mathbf{x}$ được ký hiệu gọn là (e, i) .

Với hai k -tập $\mathbf{x} = (X_1, X_2, \dots, X_k)$ và $\mathbf{y} = (Y_1, Y_2, \dots, Y_k)$ trong $(k+1)^V$, ta ký hiệu $\mathbf{x} \sqsubseteq \mathbf{y}$ nếu và chỉ nếu $X_i \subseteq Y_i$ với mọi $i \in [k]$.

Để đơn giản, ta giả sử rằng hàm g là không âm, tức là $g(\mathbf{x}) \geq 0$ với mọi $\mathbf{x} \in (k+1)^V$, và được chuẩn hoá, nghĩa là $g(\mathbf{0}) = 0$. Một hàm $g : (k+1)^V \mapsto \mathbb{R}_+$ được gọi là k -

submodular nếu với mọi $\mathbf{x} = (X_1, X_2, \dots, X_k)$ và $\mathbf{y} = (Y_1, Y_2, \dots, Y_k)$ trong $(k+1)^V$, ta có

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

trong đó

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

và

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

Các tính chất đặc trưng của hàm k -submodular được thể hiện thông qua hai điều kiện quan trọng sau đây: *trực giao submodular (orthant submodularity)* và (*đơn điệu cặp*) *pairwise monotonicity*. Hai tính chất này có thể xem như sự mở rộng tự nhiên của tính chất *lợi ích biên giảm dần* (diminishing returns) trong trường hợp hàm submodular cổ điển sang không gian $(k+1)^V$. Cụ thể, ta có các định nghĩa sau.

Tính trực giao submodular (Orthant submodularity) [68]. Hàm g được gọi là có tính trực giao submodular nếu

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

với mọi $\mathbf{x}, \mathbf{y} \in (k+1)^V$ thỏa mãn $\mathbf{x} \sqsubseteq \mathbf{y}$, $e \notin \text{supp}(\mathbf{y})$ và $i \in [k]$.

Tính đơn điệu cặp (Pairwise monotonicity) [68]. Hàm g được gọi là đơn điệu từng cặp nếu với mọi $i, j \in [k]$, $i \neq j$, ta có

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

Theo [68], nếu g là hàm k -submodular thì g thỏa mãn cả hai tính chất *trực giao* và *đơn điệu cặp*.

Chú ý. Hàm k -submodular là một mở rộng tự nhiên của hàm submodular cổ điển và kế thừa nhiều tính chất cấu trúc quan trọng trong CO. Một số trường hợp đặc biệt đáng chú ý gồm:

- Khi $k = 1$, không gian $(k+1)^V$ suy biến thành họ các tập con của V , và khi đó hàm k -submodular trở thành hàm submodular cổ điển.
- Khi $k = 2$, hàm k -submodular tương ứng với lớp hàm *bisubmodular*, vốn đã được nghiên cứu rộng rãi trong tối ưu rời rạc.

- Hàm k -submodular cho phép mô hình hóa các bài toán ra quyết định đa lựa chọn, trong đó mỗi phần tử của tập nền V có thể được gán vào tối đa một trong k loại khác nhau hoặc không được chọn. Điều này làm cho mô hình k -submodular đặc biệt phù hợp để biểu diễn các bài toán phân loại đa nhãn, tối ưu hóa ảnh hưởng theo nhiều chủ đề, cũng như các bài toán tối ưu hóa trên dữ liệu có cấu trúc đa loại.
- Tương tự như trường hợp submodular, nhiều bài toán tối ưu trên hàm k -submodular có thể được giải xấp xỉ hiệu quả bằng các thuật toán tham lam hoặc các kỹ thuật CO chuyên biệt, điều này khiến lớp hàm này trở thành một công cụ quan trọng trong các bài toán tối ưu hóa quy mô lớn.

Mặc dù hàm k -submodular kế thừa nhiều tính chất cấu trúc quan trọng của hàm submodular, việc tối ưu trên lớp hàm này thường thách thức hơn đáng kể. Một số nguyên nhân chính bao gồm:

- **Không gian lời giải lớn hơn đáng kể.** Trong khi bài toán submodular cổ điển chỉ xét các tập con của V (kích thước $2^{|V|}$), không gian $(k+1)^V$ của các k -tập có kích thước $(k+1)^{|V|}$. Điều này làm tăng đáng kể độ phức tạp của việc tìm kiếm lời giải tối ưu.
- **Cấu trúc tương tác phức tạp giữa các nhãn.** Trong mô hình k -submodular, mỗi phần tử không chỉ được chọn hay không chọn mà còn phải quyết định thuộc về loại nào trong k loại. Sự tương tác giữa các loại này làm cho việc phân tích lợi ích biên trở nên phức tạp hơn so với trường hợp submodular cổ điển.
- **Khó thiết kế thuật toán tham lam trực tiếp.** Nhiều kỹ thuật tham lam hiệu quả cho hàm submodular không thể áp dụng trực tiếp cho hàm k -submodular do sự phụ thuộc giữa các thành phần của k -tập, dẫn đến việc cần phát triển các kỹ thuật phân tích và thiết kế thuật toán mới.
- **Độ phức tạp lý thuyết cao hơn.** Trong nhiều bài toán tối ưu hóa liên quan đến k -submodular, việc đạt được các bảo đảm xấp xỉ tốt thường khó hơn và đòi hỏi các kỹ thuật phân tích tinh vi hơn so với các bài toán submodular cổ điển.

Những thách thức này khiến việc thiết kế các TXXX hiệu quả cho các bài toán tối ưu trên hàm k -submodular trở thành một hướng nghiên cứu quan trọng và đang thu hút nhiều sự quan tâm trong những năm gần đây.

3.1.2. Bài toán Phủ k -submodular

Giả sử mỗi phần tử $e \in V$ được gán một chi phí dương $c(e)$, khi đó tổng chi phí của một nghiệm $\mathbf{x}$ được xác định bởi

$$c(\mathbf{x}) = \sum_{i=1}^k \sum_{e \in X_i} c(e).$$

Trong chương này, LA nghiên cứu bài toán Bao phủ k -Submodular (kSC), được định nghĩa như sau.

Định nghĩa 3.1 (Phủ k -Submodular (kSC)). Cho một ngưỡng $T > 0$, bài toán kSC yêu cầu tìm một nghiệm $\mathbf{x} = (X_1, X_2, \dots, X_k) \in (k+1)^V$ có tổng chi phí nhỏ nhất $c(\mathbf{x})$ sao cho $g(\mathbf{x}) \geq T$.

LA xem xét trường hợp bài toán luôn tồn tại lời giải, tức là

$$T \leq \max_{\mathbf{x} \in (k+1)^V} g(\mathbf{x}).$$

Một thể hiện của bài toán kSC được ký hiệu bởi bộ (V, k, g, T) . Gọi $\mathbf{o}$ là lời giải tối ưu của bài toán và ký hiệu $\text{opt} = c(\mathbf{o})$ là giá trị tối ưu tương ứng. Định nghĩa $c_{\min} = \min_{e \in V} c(e)$, $c_{\max} = \max_{e \in V} c(e)$.

Với trường hợp $c(e) = 1, \forall e \in V$, bài toán được gọi là **Phủ k -Submodular đồng nhất** (uniform-cost k -submodular cover). Khi đó bài toán đi tìm lời giải $\mathbf{x}$ với số lượng phần tử $|supp(\mathbf{x})|$ nhỏ nhất sao cho $g(\mathbf{x}) \geq T$.

Trong trường hợp này, ta không còn xét đến sự khác biệt về chi phí giữa các phần tử mà chỉ quan tâm đến số lượng phần tử được lựa chọn. Trường hợp này có nhiều ý nghĩa thực tiễn và thường xuất hiện trong các bài toán lựa chọn tập con kích thước nhỏ nhất nhằm đạt được một mức bao phủ mong muốn, do đó được nghiên cứu chuyên sâu trong luận án.

Chú ý. Dưới đây là một số đặc điểm phân biệt giữa bài toán kSC và MCSC:

- Thứ nhất, do hàm k -submodular là một mở rộng tổng quát của hàm submodular, trong đó mỗi phần tử có thể được gán vào một trong k nhãn khác nhau, nên cấu trúc không gian nghiệm trở nên phức tạp hơn đáng kể so với trường hợp submodular cổ điển. Vì vậy, bài toán Phủ k -Submodular (kSC) nhìn chung khó hơn bài toán Phủ Submodular truyền thống. Do sự khác biệt này, các kỹ thuật và kết quả xấp xỉ đã biết cho bài toán Phủ Submodular không thể áp dụng trực tiếp để suy ra các tỷ lệ xấp xỉ cho bài toán kSC.

- Thứ hai, bài toán còn gặp những khó khăn đáng kể về mặt tính toán. Cụ thể, không gian nghiệm của bài toán là $(k + 1)^V$, có kích thước tăng theo cấp số mũ theo $|V|$, khiến cho việc đánh giá giá trị hàm và tìm kiếm lời giải trở nên tốn kém hơn so với trường hợp submodular thông thường. Điều này đặt ra thách thức lớn trong việc thiết kế các thuật toán hiệu quả cho bài toán.

Đối với trường hợp tổng quát $k \geq 2$, bài toán kSC có thể được xem là bài toán đối ngẫu tự nhiên của bài toán tối đa hóa k -submodular dưới ràng buộc knapsack, vốn đã được nghiên cứu sâu rộng [98, 95, 71]. Khuôn khổ này cho phép mô hình hóa thống nhất các tình huống trong đó cần lựa chọn một k -tập có chi phí tối thiểu sao cho giá trị tiện ích, được đo bởi một hàm k -submodular, vượt qua một ngưỡng yêu cầu. Cách mô hình này phản ánh tự nhiên nhiều bài toán thực tiễn, bao gồm lan truyền thông tin và quảng cáo trực tuyến trong OSN, bố trí cảm biến đa loại, lựa chọn đặc trưng đa lớp, cùng nhiều ứng dụng khác được trình bày dưới đây.

Ứng dụng 1: Ảnh hưởng ngưỡng với k chủ đề (ITk). Trong các kịch bản marketing lan truyền và gợi ý sản phẩm, một doanh nghiệp có thể mong muốn triển khai một chiến dịch quảng cáo gồm k nhóm sản phẩm khác nhau và tác động đến ít nhất T người dùng trong OSN. Để giảm thiểu tổng ngân sách, doanh nghiệp cần lựa chọn một tập các người dùng có ảnh hưởng lớn, đủ khả năng tạo ra mức lan tỏa mong muốn. Mô hình này là sự mở rộng tự nhiên của các mô hình lan truyền thông tin cổ điển cho trường hợp một chủ đề ($k = 1$) [1, 100], đồng thời mở rộng bài toán ngưỡng ảnh hưởng được nghiên cứu trong [81].

Xét một đồ thị có hướng $G = (V, E)$, trong đó V là tập người dùng và E là tập các cạnh biểu diễn mối quan hệ ảnh hưởng giữa các người dùng. Mỗi cạnh $(u, v) \in E$ được gán một trọng số $w_i(u, v)$, biểu diễn mức độ ảnh hưởng từ người dùng u tới người dùng v đối với chủ đề i . Mỗi người dùng v có một ngưỡng ảnh hưởng $\theta_i(v) \in [0, 1]$ cho mỗi chủ đề i , biểu thị mức ảnh hưởng tối thiểu cần thiết để người dùng đó được kích hoạt trong chủ đề tương ứng. Ngoài ra, mỗi người dùng u có một chi phí lựa chọn $c(u) > 0$, phản ánh chi phí để chọn người dùng đó làm hạt giống khởi phát quá trình lan truyền ảnh hưởng dưới một mô hình khuếch tán $\mathcal{M}$ cho trước.

Với một ngưỡng ảnh hưởng mục tiêu T , mục tiêu của bài toán là lựa chọn các tập hạt giống S_i cho mỗi chủ đề $i \in [k]$ sao cho tổng chi phí lựa chọn là nhỏ nhất, đồng thời tổng số người dùng được kích hoạt đạt ít nhất T . Nói cách khác, bài toán yêu cầu tìm nghiệm $\mathbf{s} = (S_1, \dots, S_k) \in (k + 1)^V$ sao cho

$$\min_{\mathbf{s} \in (k+1)^V} c(\mathbf{s})$$

với điều kiện

$$g(s) = \mathbb{E} \left[\left| \bigcup_{i \in [k]} \sigma_i(S_i) \right| \right] \geq T,$$

trong đó $\sigma_i(S_i)$ là tập các đỉnh được kích hoạt bởi tập hạt giống S_i trong chủ đề i .

Hàm ảnh hưởng $\sigma(\cdot)$ được biết là một *hàm đơn điệu và k -submodular* [6]. Do đó, với một ngưỡng ảnh hưởng T cho trước, bài toán trên có thể được mô hình hóa như một trường hợp của bài toán kSC.

Ứng dụng 2: Ngưỡng doanh thu với chi phí tối thiểu (Revenue Threshold-RT)

Cho đồ thị $G = (V, E)$, trong đó V là tập các đỉnh đại diện cho khách hàng và mỗi cạnh $(u, v) \in E$ có trọng số $w_{uv} \geq 0$ biểu diễn mức độ liên hệ giữa hai khách hàng u và v . Mục tiêu là phân bổ k loại sản phẩm cho khách hàng bằng cách xác định một k -tập

$$\mathbf{s} = (S_1, S_2, \dots, S_k),$$

trong đó $S_i \subseteq V$ là tập khách hàng được phân bổ sản phẩm loại i , và các tập $S_1, \dots, S_k$ đôi một rời nhau (tức là mỗi khách hàng được gán nhiều nhất một loại sản phẩm).

Hàm doanh thu của một phương án phân bổ $\mathbf{s}$ được định nghĩa bởi

$$g(\mathbf{s}) = \sum_{u \in V} \sum_{i=1}^k \left(\sum_{v \in S_i} w_{uv} \right)^{\alpha_{u,i}},$$

với $\alpha_{u,i} \in (0, 1)$. Hàm $f(\mathbf{s})$ là một hàm k -submodular và đơn điệu [95].

Giả sử mỗi khách hàng $v \in V$ khi được chọn để phân bổ sản phẩm sẽ phát sinh một chi phí $c(v) > 0$. Khi đó, chi phí của một phương án $\mathbf{s} = (S_1, \dots, S_k)$ được xác định là

$$c(\mathbf{s}) = \sum_{i=1}^k \sum_{v \in S_i} c(v).$$

Với một ngưỡng doanh thu cho trước $T > 0$, bài toán Ngưỡng Doanh thu với Chi phí Tối thiểu theo k -loại được phát biểu như sau:

$$\min_{\mathbf{s} \in (k+1)^V} c(\mathbf{s}) \quad \text{với điều kiện} \quad g(\mathbf{s}) \geq T.$$

Nói cách khác, mục tiêu của bài toán là tìm một phương án phân bổ $\mathbf{s} = (S_1, \dots, S_k)$ có tổng chi phí nhỏ nhất sao cho doanh thu thu được đạt ít nhất ngưỡng yêu cầu T .

Ứng dụng 3: Lựa chọn đặc trưng k -lớp với ngưỡng thông tin. Lựa chọn đặc trưng là một bài toán nền tảng trong học máy, giúp xử lý hiệu quả dữ liệu có số chiều lớn bằng cách loại bỏ các đặc trưng dư thừa hoặc không liên quan [101, 99, 102]. Trong bối cảnh đa lớp, mục tiêu là lựa chọn một tập gồm k đặc trưng giàu thông tin, đồng thời gán mỗi

đặc trưng cho một biến dự đoán phù hợp. Cấu trúc này tự nhiên dẫn đến một mô hình hàm k -submodular. Trong thực tế, mỗi đặc trưng thường đi kèm một chi phí, phản ánh độ phức tạp tính toán, chi phí thu thập hoặc chi phí xử lý. Việc tích hợp yếu tố chi phí là thiết yếu, bởi việc lựa chọn quá nhiều đặc trưng đắt đỏ có thể làm gia tăng đáng kể chi phí tính toán. Do đó, bài toán lựa chọn đặc trưng k -lớp với ngưỡng hướng tới việc lựa chọn các đặc trưng giàu thông tin nhất, gán chúng cho các biến dự đoán thích hợp, sao cho mỗi đặc trưng được chọn đều đóng góp lượng thông tin vượt một ngưỡng xác định trước, đồng thời kiểm soát tổng chi phí.

3.2. CÁC NGHIÊN CỨU LIÊN QUAN VÀ ĐÓNG GÓP CỦA LUẬN ÁN

Nghiên cứu về tối ưu các hàm k -submodular khởi nguồn từ trường hợp đặc biệt của bài toán bisubmodular (tương ứng với $k = 2$) trong công trình [99]. Tiếp đó, Ward và cộng sự [68] đã đặt nền móng cho bài toán tối đa hóa hàm k -submodular không ràng buộc, với một thuật toán tham lam xác định đạt tỷ lệ xấp xỉ $1/3$. Trong công trình sau đó [69], tỷ lệ này được nâng lên $1/2$, đồng thời một biến thể tham lam ngẫu nhiên được đề xuất với tỷ lệ xấp xỉ kỳ vọng $k/(2k - 1)$ thông qua cơ chế chọn phần tử dựa trên phân phối xác suất. Nhằm tránh yếu tố ngẫu nhiên, [103] đã đưa ra một thuật toán xác định đạt cùng tỷ lệ xấp xỉ $k/(2k - 1)$, nhưng phải đánh đổi bằng chi phí truy vấn rất cao, lên tới $O(n^2 k^2)$.

Sau đó, bài toán tiếp tục được mở rộng sang các bối cảnh có ràng buộc, bao gồm ràng buộc lực lượng (cardinality) và các dạng ràng buộc tổng quát hơn. Cụ thể, [11] xem xét trường hợp hàm đơn điệu với hai kiểu ràng buộc lực lượng: (i) ràng buộc tổng trên toàn bộ nghiệm và (ii) ràng buộc riêng lẻ trên từng thành phần. Trong hai trường hợp này, các thuật toán tham lam đạt tỷ lệ xấp xỉ lần lượt là $1/2$ và $1/3$. Những nghiên cứu tiếp theo đã cải thiện hiệu năng thông qua nhiều hướng tiếp cận khác nhau, như tối ưu hóa đa mục tiêu [96], phân tích dựa trên regret [104], và các mô hình xử lý dữ liệu luồng [73, 105]. Ngoài ra, [70] cũng đề xuất một thuật toán tham lam đạt tỷ lệ $1/2$ cho trường hợp có ràng buộc matroid.

Đối với ràng buộc knapsack, một mở rộng tự nhiên của ràng buộc lực lượng, [71] đã đạt được kết quả xấp xỉ đầu tiên với tỷ lệ $(1/2 - 1/(2e))$ bằng phương pháp tham lam. Tuy nhiên, cách tiếp cận này đòi hỏi số lượng truy vấn rất lớn, lên tới $O(n^4 k^3)$. Kết quả sau đó được cải thiện lên $1/2$ trong [72] nhờ sử dụng kỹ thuật mở rộng đa tuyến (multilinear extension) với độ phức tạp đa thức. Gần đây, [95] đã đề xuất một thuật toán thực tiễn với tỷ lệ xấp xỉ $1/3$ và chi phí truy vấn gần tuyến tính $O(nk)$.

Bài toán kSC được giới thiệu lần đầu trong [74], nơi các tác giả đưa ra những kết

quả ban đầu cùng một TTXX tiêu chuẩn kép. Tuy nhiên, hệ số xấp xỉ đạt được còn khá hạn chế trong bối cảnh này (xem Bảng 3.1). Do đó, việc thiết kế các TTXX đa thức với đảm bảo lý thuyết chặt chẽ hơn, đồng thời vẫn đảm bảo hiệu quả thực nghiệm, vẫn là một thách thức đáng kể.

Gần đây, [75] đã nghiên cứu bài toán kSC trong mô hình xử lý luồng và đề xuất các TTXX tiêu chuẩn kép với bảo đảm $\left(\frac{3-\epsilon}{2\epsilon(1-\epsilon)}, \frac{1-\epsilon}{2}\right)$. Tuy nhiên, các phương pháp này chưa đảm bảo thời gian chạy đa thức trong trường hợp tổng quát, đồng thời chi phí tính toán có thể tăng cao do phụ thuộc vào số lần quét dữ liệu và các ràng buộc về bộ nhớ.

Tổng hợp các kết quả hiện có cho thấy rằng việc phát triển các TTXX hiệu quả với thời gian chạy đa thức cho bài toán kSC vẫn là một hướng nghiên cứu mở và cần được tiếp tục khai thác.

Đóng góp của Luận án. Luận án tập trung nghiên cứu bài toán kSC trong hai trường hợp chính: (1) chi phí đồng nhất (*uniform-cost*) và (2) chi phí tổng quát (*general-cost*).

a. Trường hợp bài toán Phủ k-Submodular đồng nhất.

Đối với trường hợp chi phí đồng nhất, luận án đề xuất thuật toán FBiU và phân tích các đảm bảo lý thuyết của thuật toán *xấp xỉ tiêu chuẩn kép*. Cụ thể, thuật toán trả về một nghiệm s sao cho kích thước nghiệm được khống chế so với nghiệm tối ưu và đồng thời giá trị hàm mục tiêu đạt được một tỷ lệ nhất định so với ngưỡng yêu cầu. Bảng 3.1 trình bày so sánh giữa thuật toán FBiU được đề xuất trong luận án và thuật toán của Pham và cộng sự trong [74]. Kết quả cho thấy thuật toán FBiU đạt được đảm bảo xấp xỉ tiêu chuẩn kép mạnh hơn đáng kể. Cụ thể hơn, khi thiết lập $\delta = \epsilon$, thuật toán đảm bảo nghiệm thu được có kích thước

$$|supp(\mathbf{s})| \leq (1 + \epsilon) \left(1 + \ln\left(\frac{1}{\epsilon}\right)\right) \text{opt} << \frac{(1 - \epsilon)(1 + \epsilon)}{2\epsilon} \text{opt}.$$

với $\epsilon = 0.1$ trong thực nghiệm, đồng thời đạt giá trị hàm mục tiêu

$$g(\mathbf{s}) \geq \frac{1 - \epsilon}{2} T$$

trong đó opt là kích thước nghiệm tối ưu.

b. Trường hợp bài toán Phủ k-Submodular tổng quát

Đối với trường hợp chi phí tổng quát, luận án đề xuất hai thuật toán xấp xỉ tiêu chuẩn kép xác định cho bài toán kSC trong thời gian đa thức, bao gồm thuật toán *Tham lam mở rộng* (*Extended Greedy*) và thuật toán FBi. Các thuật toán này được thiết kế nhằm đồng thời cải thiện các đảm bảo lý thuyết về chất lượng nghiệm và giảm độ phức tạp truy vấn so với các phương pháp hiện có trong tài liệu. Cụ thể, thuật toán *Extended*

Bảng 3.1: Các thuật toán cho bài toán kSC. Cặp (x, y) biểu diễn tỷ lệ xấp xỉ tiêu chuẩn kép (x, y) của một thuật toán, tức là thuật toán trả về nghiệm $\mathbf{s}$ sao cho $|supp(\mathbf{s})| \leq x \cdot \text{opt}$ và $g(\mathbf{s}) \geq y \cdot T$, với $x, y > 0$.

Tài liệu	Tỷ lệ xấp xỉ	Độ phức tạp truy vấn
FBiU (Thuật toán 3.2, LA)	$\left(\frac{1+\epsilon}{2} \log\left(\frac{1}{\delta}\right), \frac{1-\delta}{2}\right)$	$O\left(\frac{nk \log^2(n)}{\epsilon} \log\left(\frac{n}{\delta}\right)\right)$
Thuật toán luồng [74]	$\left(\frac{(1-\epsilon)(1+\epsilon)}{2\epsilon}, \frac{1-\epsilon}{2}\right)$	$O\left(nk \frac{\log(n)}{\epsilon}\right)$

Greedy đạt độ phức tạp truy vấn $O(n^2 k)$ và đảm bảo tỷ lệ xấp xỉ tiêu chuẩn kép

$$\left(\frac{1}{2} \left(\ln\left(\frac{1}{\epsilon}\right) + \frac{1+\epsilon}{\epsilon}\right), \frac{1-\epsilon}{2}\right)$$

đối với trường hợp hàm lợi ích g là đơn điệu, và

$$\left(\frac{1}{3} \left(\ln\left(\frac{1}{\epsilon}\right) + \frac{2+\epsilon}{\epsilon}\right), \frac{1-\epsilon}{3}\right)$$

đối với trường hợp g không đơn điệu.

Bên cạnh đó, LA đề xuất thuật toán FBi cho trường hợp hàm lợi ích đơn điệu, với độ phức tạp truy vấn $O(\frac{nk}{\epsilon^2} \log(\frac{n}{\epsilon}) \log(\frac{1}{\delta}))$, đồng thời đạt tỷ lệ xấp xỉ tiêu chuẩn kép gần tương đương với tham lam mở rộng. FBi hiệu quả khi kích thước dữ liệu n lớn. Ví dụ khi $n \geq 2.000, k = 100, \epsilon = 0.1$ thì tham lam mở rộng cần tới 400 triệu truy vấn còn FBi cần khoảng 198 triệu truy vấn.

Bảng 3.2 trình bày so sánh giữa các thuật toán được đề xuất trong luận án với các thuật toán tiên tiến hiện nay cho bài toán kSC. Có thể thấy rằng các thuật toán của luận án đạt được đảm bảo tốt hơn về chi phí nghiệm so với các thuật toán luồng trong [75], đồng thời giảm đáng kể độ phức tạp truy vấn từ dạng không đa thức xuống còn đa thức.

Thuật toán	Tỷ lệ xấp xỉ	Độ phức tạp truy vấn	Loại thuật toán
Hàm đơn điệu			
Tham lam mở rộng (LA)	$\left(\frac{1}{2} \left(\log\left(\frac{1}{\epsilon}\right) + \frac{1+\epsilon}{\epsilon}\right), \frac{1-\epsilon}{2}\right)$	$O(n^2 k)$	-
FBi (LA)	$\left(\frac{1}{2} (1+\epsilon) \left(\ln\left(\frac{1}{\delta}\right) + \frac{\delta+1}{\delta}\right), \frac{1-\delta}{2}\right)$	$O\left(\frac{nk}{\epsilon^2} \log\left(\frac{n}{\epsilon}\right) \log\left(\frac{1}{\delta}\right)\right)$	-
Thuật toán 2 [75]	$\left(\frac{3-\epsilon}{2\epsilon(1-\epsilon)}, \frac{1-\epsilon}{2}\right)$	$O\left(nk \frac{\log(\frac{c_{min}}{nc_{max}})}{\log(1-\epsilon)}\right)$	Luồng
Thuật toán 3 [75]	$\left(\frac{3-\epsilon}{2\epsilon(1-\epsilon)}, \frac{1-\epsilon}{2}\right)$	$O\left(nk \frac{\log(\epsilon T (B \max_{e \in V, i \in [k]} g((e, i)))^{-1})}{\log(1-\epsilon)}\right)$	Luồng
Hàm không đơn điệu			
Tham lam mở rộng (LA)	$\left(\frac{1}{3} \left(\log\left(\frac{1}{\epsilon}\right) + \frac{2+\epsilon}{\epsilon}\right), \frac{1-\epsilon}{3}\right)$	$O(n^2 k)$	-
Thuật toán 2 [75]	$\left(\frac{4-\epsilon}{3\epsilon(1-\epsilon)}, \frac{1-\epsilon}{3}\right)$	$O\left(nk \frac{\log(\frac{c_{min}}{nc_{max}})}{\log(1-\epsilon)}\right)$	Luồng
Thuật toán 3 [75]	$\left(\frac{4-\epsilon}{3\epsilon(1-\epsilon)}, \frac{1-\epsilon}{3}\right)$	$O\left(nk \frac{\log(\epsilon T (B \max_{e \in V, i \in [k]} g((e, i)))^{-1})}{\log(1-\epsilon)}\right)$	Luồng

Bảng 3.2: Các thuật toán cho bài toán kSC. Bộ (x, y) biểu diễn tỷ lệ xấp xỉ tiêu chuẩn kép (x, y) .

So sánh với các thuật toán [75], ta có so sánh cụ thể hơn như sau:

- Thuật toán tham lam mở rộng đạt được cùng mức bảo đảm về giá trị hàm mục tiêu như các thuật toán trong [75], nhưng cho tỷ số xấp xỉ tốt hơn đáng kể, tức là cận trên của chi phí nghiệm so với chi phí tối ưu được giảm xuống. Cụ thể, mức cải thiện đạt trên 1.5 lần đối với trường hợp hàm g đơn điệu và trên 2 lần đối với trường hợp hàm g không đơn điệu. Điều này dựa trên cơ sở các đánh giá sau:

$$\frac{3 - \epsilon}{2\epsilon(1 - \epsilon)} \geq 1.5 \cdot \frac{1}{2} \left(\ln \left(\frac{1}{\epsilon} \right) + \frac{1 + \epsilon}{\epsilon} \right) + 1$$

và

$$\frac{4 - \epsilon}{3\epsilon(1 - \epsilon)} \geq 2 \cdot \frac{1}{3} \left(\ln \left(\frac{1}{\epsilon} \right) + \frac{2 + \epsilon}{\epsilon} \right) + 1.$$

- Thuật toán FBI đạt được tỷ lệ xấp xỉ tiêu chuẩn kép tiệm cận với Tham lam mở rộng, sai khác một đại lượng nhỏ là $1 + \epsilon$ tuy nhiên giảm độ phức tạp truy vấn xuống $\Omega(\frac{n}{\log(n)})$ lần.

3.3. THUẬT TOÁN CHO TRƯỜNG HỢP CHI PHÍ ĐỒNG NHẤT

Phần này trình bày thuật toán FBIU đề xuất cho bài toán kSC trong trường hợp hàm tiện ích g là đơn điệu và các chi phí của mỗi phần tử $c(e) = 1, \forall e \in V$.

Trước hết, LA mô tả thuật toán FBIUOpt, một biến thể tham lam ngẫu nhiên giả định đã biết (ước đoán) kích thước tối ưu opt (hay tương đương là kích thước nghiệm tối ưu). Thuật toán FBIUOpt đóng vai trò như phiên bản đơn giản hóa của FBIU nhằm thuận tiện cho việc phân tích. Sau đó, LA trình bày thuật toán đầy đủ FBIU, trong đó loại bỏ giả định phải biết trước opt.

3.3.1. Thuật toán FBIUOpt (biết ước lượng tối ưu)

Thuật toán FBIUOpt nhận đầu vào là một thể hiện (V, g, k, T) , một giá trị ước đoán v của kích thước nghiệm tối ưu, cùng hai tham số chính xác $\epsilon \in (0, 1/2)$ và $\delta > 0$. Ta định nghĩa lại hàm g như sau: $g : (k + 1)^V \rightarrow \mathbb{R}_+$ bởi

$$g(\cdot) \leftarrow \min\{g(\cdot), T/2\},$$

lưu ý rằng hàm thu được vẫn là k -submodular và đơn điệu.

Ý tưởng chính của FBIUOpt là sử dụng lấy mẫu ngẫu nhiên trên tập $V \setminus \text{supp}(\mathbf{s})$ nhằm giảm số lượng truy vấn, dưới giả định đã biết ước lượng v . Theo đó, thuật toán thực hiện tối đa $\lceil \frac{v}{2} \log \left(\frac{1}{\delta} \right) \rceil$ vòng lặp. Tại mỗi vòng lặp j , thuật toán lấy mẫu đều một tập con R^j có kích thước Υ từ $V \setminus \text{supp}(\mathbf{s})$, sau đó chọn cặp (e, i) với $e \in R^j$ và $i \in [k]$ sao

Thuật toán 3.1: Thuật toán FBIUOpt

Input: Một thể hiện (V, g, k, T) , $\delta \in (0, 1)$, $v \in (1, n)$, Υ

Output: Một nghiệm $\mathbf{s}$

```

1:  $\mathbf{s} \leftarrow \mathbf{0}$ , Định nghĩa lại hàm  $g : (k+1)^V \mapsto \mathbb{R}_+$  như sau  $g(\cdot) \leftarrow \min\{g(\cdot), T/2\}$ 
2:  $t_0 \leftarrow \lfloor \frac{v}{2} \ln \frac{1}{\delta} \rfloor + 1$ ,  $T \leftarrow \min\{t_0, v\}$ 
3: for  $j = 1$  to  $T$  do
4:    $R^j \leftarrow$  chọn ngẫu nhiên  $\Upsilon$  phần tử từ tập  $V \setminus \text{supp}(\mathbf{s})$ 
5:    $(e, i) \leftarrow \arg \max_{e \in R, i \in [k]} \Delta_{(e,i)} g(\mathbf{s})$ 
6:    $\mathbf{s} \leftarrow \mathbf{s} \sqcup (e, i)$ 
7: end
8: return  $\mathbf{s}$ 

```

cho lợi ích biên $\Delta_{(e,i)} g(\mathbf{s})$ là lớn nhất, và cập nhật nghiệm theo $\mathbf{s} \leftarrow \mathbf{s} \sqcup (e, i)$. Thuật toán kết thúc sau số vòng lặp đã định và trả về nghiệm cuối cùng $\mathbf{s}$. Giả mã chi tiết được trình bày trong Thuật toán 3.1.

Để phân tích Thuật toán 3.1, Luận án sử dụng các ký hiệu sau:

- (e_j, i_j) là cặp phần tử được thêm vào tại vòng lặp thứ j .
- $\mathbf{s} = \{(e_1, i_1), \dots, (e_t, i_t)\}$ là nghiệm cuối cùng, với $t = |\text{supp}(\mathbf{s})|$.
- $\mathbf{s}^j = \{(e_1, i_1), \dots, (e_j, i_j)\}$ là nghiệm sau j vòng lặp, trong đó $\mathbf{s}^0 = \mathbf{x}$ và $\mathbf{s}^t = \mathbf{s}$.
- Với mọi k -set $\mathbf{x}$ thỏa $|\text{supp}(\mathbf{x})| = v$, đặt

$$\mathbf{x}^j = (\mathbf{x} \sqcup \mathbf{s}^j) \sqcup \mathbf{s}^j, \quad \mathbf{x}^{j-\frac{1}{2}} = (\mathbf{x} \sqcup \mathbf{s}^j) \sqcup \mathbf{s}^{j-1}.$$

- $X^j = \text{supp}(\mathbf{x}^{j-1}) \setminus \text{supp}(\mathbf{s}^{j-1})$.
- $\mathbf{u}^t = \{(u_1, i_1), \dots, (u_r, i_r)\}$ là tập các phần tử thuộc $\mathbf{x}^t$ nhưng không thuộc $\mathbf{s}^t$, với $r = |\text{supp}(\mathbf{u}^t)|$.
- $\mathbf{u}_\ell^t = \mathbf{s}^t \sqcup \{(u_1, i_1), \dots, (u_\ell, i_\ell)\}$ với $1 \leq \ell \leq r$, và $\mathbf{u}_0^t = \mathbf{s}^t$.

Việc phân tích nhằm thiết lập các bảo đảm lý thuyết và hệ số xấp xỉ của thuật toán được khởi đầu từ bổ đề cơ bản nhưng có vai trò then chốt sau đây.

Bổ đề 3.1. *Tại mỗi vòng lặp j của Thuật toán 3.1, nếu chọn*

$$\Upsilon = \min \left\{ V \setminus \text{supp}(\mathbf{s}), \frac{n-j+1}{v-j+1} \log \left(\frac{n}{\delta} \right) \right\},$$

thì ta có

$$\Pr[R^j \cap X^j = \emptyset] \leq \frac{\delta}{n}.$$

Chứng minh. Tại vòng lặp thứ j , các phần tử đã được chọn ở các vòng trước thuộc $\text{supp}(\mathbf{s}^{j-1})$ nên không được xét lại. Vì vậy, R^j được lấy mẫu đều và không hoàn lại từ tập $V \setminus \text{supp}(\mathbf{s}^{j-1})$. Đặt

$$N_j = |V \setminus \text{supp}(\mathbf{s}^{j-1})| = n - j + 1, \quad K_j = |X^j| \geq v - j + 1, \quad r_j = |R^j|.$$

Do lấy mẫu không hoàn lại, xác suất chính xác để R^j không chứa phần tử nào thuộc X^j là

$$\Pr[R^j \cap X^j = \emptyset] = \frac{\binom{N_j - K_j}{r_j}}{\binom{N_j}{r_j}} = \prod_{t=0}^{r_j-1} \left(1 - \frac{K_j}{N_j - t}\right).$$

Do $N_j - t \leq N_j$, ta có

$$\Pr[R^j \cap X^j = \emptyset] \leq \left(1 - \frac{K_j}{N_j}\right)^{r_j} \quad (3.1)$$

$$\leq \exp\left(-\frac{K_j r_j}{N_j}\right). \quad (3.2)$$

Nếu $r_j = N_j$ thì toàn bộ các phần tử còn lại được chọn và xác suất trên bằng 0. Ngược lại, theo cách chọn kích thước mẫu,

$$r_j \geq \frac{N_j}{v - j + 1} \log\left(\frac{n}{\delta}\right).$$

Kết hợp với $K_j \geq v - j + 1$, ta được $\frac{K_j r_j}{N_j} \geq \log\left(\frac{n}{\delta}\right)$. Do đó,

$$\Pr[R^j \cap X^j = \emptyset] \leq \exp\left(-\log\left(\frac{n}{\delta}\right)\right) = \frac{\delta}{n}.$$

□

Bổ đề sau đây thiết lập một bất đẳng thức then chốt, liên hệ trực tiếp giữa phần giá trị mục tiêu còn thiếu của nghiệm tối ưu và giá trị đạt được bởi nghiệm trung gian của thuật toán tại vòng lặp hiện tại. Cụ thể, khi tập mẫu ngẫu nhiên R^j chứa ít nhất một phần tử “tốt” trong X^j , ta chứng minh rằng mức cải thiện còn lại của nghiệm tối ưu so với nghiệm hiện tại $\mathbf{x}^j$ có thể được khống chế bởi giá trị tích lũy của nghiệm xây dựng $\mathbf{s}^j$. Kết quả này đóng vai trò nền tảng cho việc suy ra bảo đảm xấp xỉ toàn cục của thuật toán.

Bổ đề 3.2. Sau vòng lặp j của Thuật toán 3.1, giả sử $R^j \cap X^j \neq \emptyset$, ta có

$$g(\mathbf{x}) - g(\mathbf{x}^j) \leq g(\mathbf{s}^j).$$

Chứng minh. Đặt $\mathbf{x}^{j-\frac{1}{2}} = (\mathbf{x} \sqcup \mathbf{s}^j) \sqcup \mathbf{s}^{j-1}$, và xác định nghiệm trung gian $\mathbf{s}^{j-\frac{1}{2}}$ như sau:

- Nếu $e_j \in X^j$ thì $\mathbf{s}^{j-\frac{1}{2}} = \mathbf{s}^{j-1} \sqcup (e_j, \mathbf{x}(e_j))$.
- Nếu $e_j \notin X^j$ thì $\mathbf{s}^{j-\frac{1}{2}} = \mathbf{s}^{j-1}$.

Khi đó, ta có chuỗi bất đẳng thức

$$g(\mathbf{x}^{l-1}) - g(\mathbf{x}^l) \leq g(\mathbf{x}^{l-1}) - g(\mathbf{x}^{l-\frac{1}{2}}), \quad (3.3)$$

$$\leq g(\mathbf{s}^{l-\frac{1}{2}}) - g(\mathbf{s}^{l-1}), \quad (3.4)$$

$$\leq g(\mathbf{s}^l) - g(\mathbf{s}^{l-1}), \quad (3.5)$$

trong đó:

- Bất đẳng thức (3.3) suy ra từ tính đơn điệu của hàm f ;
- Bất đẳng thức (3.4) là hệ quả trực tiếp của tính k -submodular của f ;
- Bất đẳng thức (3.5) được đảm bảo bởi quy tắc lựa chọn phần tử của thuật toán.

Cộng hai vế theo $l = 1, \dots, j$, ta thu được

$$g(\mathbf{x}) - g(\mathbf{x}^j) = \sum_{l=1}^j (g(\mathbf{x}^{l-1}) - g(\mathbf{x}^l)) \quad (3.6)$$

$$\leq \sum_{l=1}^j (g(\mathbf{s}^l) - g(\mathbf{s}^{l-1})) \quad (3.7)$$

$$\leq g(\mathbf{s}^j). \quad (3.8)$$

□

Định lý 3.1. Với $\delta \in (0, 1)$ và $\frac{v}{2} \log(\frac{1}{\delta}) \leq n$, thuật toán 3.1 sử dụng

$$O\left(kn \log(v \log(\frac{1}{\delta})) \log\left(\frac{n}{\delta}\right)\right)$$

truy vấn và, với xác suất ít nhất $1 - \delta$, trả về nghiệm $\mathbf{s}$ thỏa

$$|supp(\mathbf{s})| = \frac{v}{2} \log(\frac{1}{\delta}), \quad g(\mathbf{s}) \geq \left(1 - e^{-\frac{v}{\text{opt}} \log(\frac{1}{\delta})}\right) \frac{T}{2}.$$

Chứng minh. Thuật toán thực hiện tối đa $t = \lceil \frac{v}{2} \log(\frac{1}{\delta}) \rceil$ vòng lặp for. Trong mỗi vòng lặp, thuật toán sử dụng $k\Upsilon$ truy vấn. Do vậy độ phức tạp truy vấn của thuật toán được xác định bởi

$$\begin{aligned} \sum_{i=1}^{\lceil \frac{v}{2} k \log(\frac{1}{\delta}) \rceil} k\Upsilon &= k \sum_{i=1}^t \frac{n-j+1}{v-j+1} \log\left(\frac{n}{\delta}\right) = k \sum_{j \in [t]} \frac{n-t+j}{j} \log\left(\frac{n}{\delta}\right) \\ &= O(kn \log(t) \log(\frac{n}{\delta})). \end{aligned}$$

Áp dụng Bổ đề 3.1, với việc xét $\mathbf{x} = \mathbf{o}$ (lời giải tối ưu) với mỗi vòng lặp j thì $\Pr[X^j \cap R^j = \emptyset] \leq \delta$. Theo đẳng thức hợp (union bound) của xác suất, ta có

$$\Pr[\exists j \in [j], X^j \cap R^j = \emptyset] \leq \frac{\delta}{n} t \leq \delta. \quad (3.9)$$

Trong đó t là số vòng lặp for của thuật toán. Suy ra

$$\Pr[\forall j \in [j], X^j \cap R^j \neq \emptyset] \geq 1 - \delta. \quad (3.10)$$

Dựa trên điều kiện $X^j \cap R^j \neq \emptyset, \forall j \in [j]$, áp dụng Bổ đề 3.2, ta thu được

$$T - g(\mathbf{s}^j) \leq g(\mathbf{o}) - g(\mathbf{s}^j) \quad (3.11)$$

$$= g(\mathbf{o}) - g(\mathbf{o}^j) + g(\mathbf{o}^j) - g(\mathbf{s}^j) \quad (3.12)$$

$$\leq g(\mathbf{s}^j) + \sum_{e \in X^j} \Delta_{(e, \mathbf{o}(e))} f(\mathbf{s}^j) \quad (3.13)$$

$$\leq g(\mathbf{s}^j) + \text{opt} \max_{e \in X^j} (f(\mathbf{s}^j \sqcup (e, i)) - f(\mathbf{s}^j)) \quad (\text{Do } X^j \subseteq \text{supp}(\mathbf{o}), \text{ nên } |X^j| \leq \text{opt}) \quad (3.14)$$

$$\leq g(\mathbf{s}^j) + \text{opt}(g(\mathbf{s}^{j+1}) - g(\mathbf{s}^j)) \quad (3.15)$$

Áp dụng bất đẳng thức trên qua t vòng lặp (với giả thiết rằng sự kiện $X^j \cap R^j \neq \emptyset$ xảy ra với mọi j), ta có

$$T - 2g(\mathbf{s}^t) \leq \left(1 - \frac{2}{\text{opt}}\right) (T - 2g(\mathbf{s}^{t-1})) \quad (3.16)$$

$$\leq e^{-\frac{2}{\text{opt}}} (T - 2g(\mathbf{s}^{t-1})) \quad (3.17)$$

$$\leq e^{-\frac{2}{\text{opt}} t} T \quad (\text{lặp lại bất đẳng thức (3.17) trong } t \text{ vòng}) \quad (3.18)$$

$$\leq e^{-\frac{2}{\text{opt}} \frac{v}{2} \log(\frac{1}{\delta})} T \quad (3.19)$$

$$= e^{-\frac{v}{\text{opt}} \log(\frac{1}{\delta})} T. \quad (3.20)$$

Do đó, ta suy ra

$$g(\mathbf{s}) \geq \frac{1}{2} \left(1 - e^{-\frac{v}{\text{opt}} \log(\frac{1}{\delta})}\right) T,$$

hoàn tất chứng minh. □

3.3.2. Thuật toán FBiU

Thuật toán chính FBiU (Thuật toán 3.2), trong đó loại bỏ giả định rằng đã biết trước một ước lượng v của kích thước nghiệm tối ưu. Thuật toán FBiU trước hết xây dựng tập ứng viên

$$O = \{(1 + \epsilon)^i : 1 \leq (1 + \epsilon)^i \leq n, i \in \mathbb{N}\}$$

Thuật toán 3.2: Thuật toán FBiU

Input: Một thể hiện (V, g, k, T) , g đơn điệu, $\epsilon \in (0, 1/2)$, $\delta > 0$

Output: Một nghiệm $\mathbf{s}$

```

1:  $O = \{(1 + \epsilon)^i : i \in \mathbb{N}, 1 \leq (1 + \epsilon)^i \leq n\}$ 
2: foreach  $v \in O$  do
3:    $\mathbf{s}_v \leftarrow \text{FBiUOpt}(V, g, k, T, v, \delta)$ 
4: end
5:  $\mathbf{s} \leftarrow \arg \max \{g(\mathbf{s}_v) : |\text{supp}(\mathbf{s}_v)| \leq v \log(\frac{1}{\delta}), v \in O\}$ 
6: return  $\mathbf{s}$ 

```

các giá trị v . Với mỗi $v \in O$, thuật toán gọi thủ tục con $\text{FBiOpt}(V, f, k, v, \delta)$ để tính một nghiệm ứng viên $\mathbf{s}_v$. Sau khi sinh ra toàn bộ các nghiệm ứng viên, FBiU lựa chọn nghiệm có giá trị hàm mục tiêu lớn nhất trong số các nghiệm thỏa mãn ràng buộc kích thước $|\text{supp}(\mathbf{s}_v)| \leq v \log(\frac{1}{\delta})$ (Dòng 5). Cuối cùng, thuật toán xuất ra nghiệm được chọn $\mathbf{s}$.

Định lý 3.2. Với $\epsilon \in (0, 1/2)$, $\delta \in (0, 1)$, với xác suất ít nhất $1 - \delta$, Thuật toán 3.2 trả về một nghiệm xấp xỉ tiêu chuẩn kép $\left(\frac{1+\epsilon}{2} \log(\frac{1}{\delta}), \frac{1-\delta}{2}\right)$ và có độ phức tạp truy vấn $O\left(\frac{nk \log^2(n)}{\epsilon} \log\left(\frac{n}{\delta}\right)\right)$.

Chứng minh. Thuật toán 3.2 gọi thủ tục FBiUOpt đúng $|O|$ lần. Do đó, tổng số truy vấn được chặn trên bởi

$$\begin{aligned} \sum_{v \in O} O\left(kn \log\left(v \log\left(\frac{1}{\delta}\right)\right) \log\left(\frac{n}{\delta}\right)\right) &\leq O\left(kn \log_{1+\epsilon}(n) \log\left(n \log\left(\frac{1}{\delta}\right)\right) \log\left(\frac{n}{\delta}\right)\right) \\ &= O\left(\frac{kn \log^2(n)}{\epsilon} \log\left(\frac{n}{\delta}\right)\right). \end{aligned}$$

Vì $1 \leq \text{opt} \leq n$, tồn tại một chỉ số $j \in \mathbb{N}$ sao cho

$$\text{opt} \leq v = (1 + \epsilon)^j \leq (1 + \epsilon) \text{opt}.$$

Theo các bảo đảm lý thuyết của thuật toán FBiUOpt (Định lý 3.1), ta có

$$|\text{supp}(\mathbf{s}_v)| = \frac{v}{2} \log\left(\frac{1}{\delta}\right) \leq \frac{1+\epsilon}{2} \log\left(\frac{1}{\delta}\right) \text{opt}. \quad (3.21)$$

Hơn nữa, với xác suất ít nhất $1 - \delta$,

$$g(\mathbf{s}) \geq \left(1 - e^{-\frac{v}{\text{opt}} \log(\frac{1}{\delta})}\right) \frac{T}{2} \quad (3.22)$$

$$\geq \left(1 - e^{-\log(\frac{1}{\delta})}\right) \frac{T}{2} \quad (\text{do } v \geq \text{opt}) \quad (3.23)$$

$$\geq \frac{(1 - \delta)T}{2}. \quad (3.24)$$

Do đó, theo quy tắc tìm kiếm của thuật toán, FBiU luôn lựa chọn được một nghiệm thỏa mãn các bảo đảm lý thuyết đã nêu. $\square$

3.3.3. Thực nghiệm

Để đánh giá hiệu quả của thuật toán FBiU, LA tiến hành so sánh thuật toán đề xuất với các phương pháp tiên tiến nhất hiện có trong tài liệu.

3.3.3.1. Thiết lập thực nghiệm

LA tiến hành đánh giá thực nghiệm thuật toán FBiU thông qua việc so sánh với hai phương pháp cơ sở tiêu biểu hiện nay là **GREEDY** [11] và **STREAMING** [74]. Việc so sánh được thực hiện dựa trên ba tiêu chí chính: (i) chất lượng nghiệm (đo bằng số lượng phần tử trong tập lời giải), (ii) số lần truy vấn hàm, và (iii) thời gian thực thi của thuật toán.

Các thí nghiệm được triển khai trên bài toán ứng dụng ***k*-type Product Revenue Cover**, sử dụng hai bộ dữ liệu đại diện: (i) một đồ thị Erdos–Rényi (ER) sinh ngẫu nhiên với tham số $(n = 2000, k = 3)$ và xác suất cạnh $p = 0.01$; (ii) mạng Email¹ với $(n = 1005, k = 5)$.

Cho tập nền V gồm n người dùng trong OSN và một nghiệm k -set $\mathbf{s} = (S_1, S_2, \dots, S_k) \in (k+1)^V$, LA xét hàm tiện ích dưới dạng giá trị doanh thu mở rộng, tương tự như trong [23], được định nghĩa bởi

$$g(\mathbf{s}) = \sum_{u \in V} \sum_{i=1}^k \left(\sum_{v \in S_i} w_{uv} \right)^{\alpha_{u,i}},$$

trong đó $\alpha_{u,i} \in (0, 1)$ mô hình hóa mức độ nhạy cảm của người dùng u đối với sản phẩm i . Hàm f là đơn điệu và k -submodular trên V .

Trong toàn bộ các thí nghiệm, các tham số được cố định ở mức $\epsilon = 0.1$ và $\delta = 0.1$.

3.3.3.2. Kết quả thực nghiệm

Các kết quả thực nghiệm được minh họa trong Hình 3.1 và 3.2. Mỗi hình trình bày sự đối chiếu giữa thuật toán đề xuất FBiU và hai phương pháp cơ sở là GREEDY và luồng, dựa trên ba tiêu chí chính: (i) kích thước nghiệm thu được, (ii) số lần truy vấn hàm, và (iii) thời gian thực thi của các thuật toán. Những kết quả này cho phép đánh giá đồng thời chất lượng lời giải cũng như chi phí tính toán của từng phương pháp trên các bộ dữ liệu khác nhau.

a. Chất lượng lời giải (kích thước)

¹<https://snap.stanford.edu/data/email-Eu-core.html>

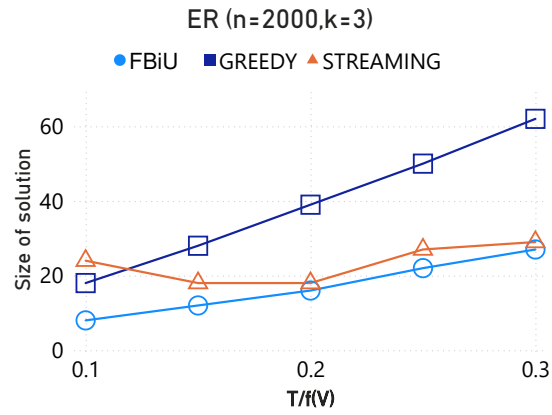
Thuật toán FBiU duy trì các nghiệm có kích thước nhỏ gọn, trong đó kích thước nghiệm tăng tuyến tính theo ngưỡng T . Trên đồ thị ER, FBiU tạo ra các nghiệm có kích thước từ 8 đến 27 phần tử, nhỏ hơn đáng kể so với GREEDY (18–62) và tương đương với luồng (18–29). Đối với tập dữ liệu Email, FBiU chỉ cần từ 1 đến 4 phần tử để đạt được tính khả thi, trong khi GREEDY cần tới 10 phần tử và luồng có thể cần tới 30 phần tử. Kết quả này cho thấy FBiU có khả năng tạo ra các nghiệm gọn nhẹ hơn trong khi vẫn đảm bảo các ràng buộc bài toán.

b. Số lượng truy vấn

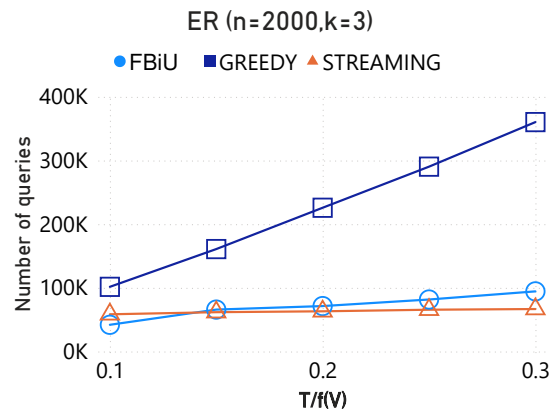
FBiU thể hiện ưu thế rõ rệt trong việc giảm số lượng truy vấn hàm. Trên dữ liệu ER, số truy vấn của FBiU nhỏ hơn khoảng 3–4 lần so với GREEDY và tương đương hoặc thấp hơn luồng. Trên dữ liệu Email, sự khác biệt còn rõ ràng hơn: FBiU chỉ cần khoảng 10^3 – 10^4 truy vấn, trong khi GREEDY và luồng yêu cầu từ 10^4 đến 10^5 truy vấn. Điều này cho thấy FBiU vượt trội về hiệu quả tính toán so với GREEDY. Tại bộ dữ liệu ER, STREAMING cho kết quả truy vấn khá tốt, tuy nhiên kết quả này không ổn định trên bộ dữ liệu Email (có kích thước lớn hơn).

c. Thời gian

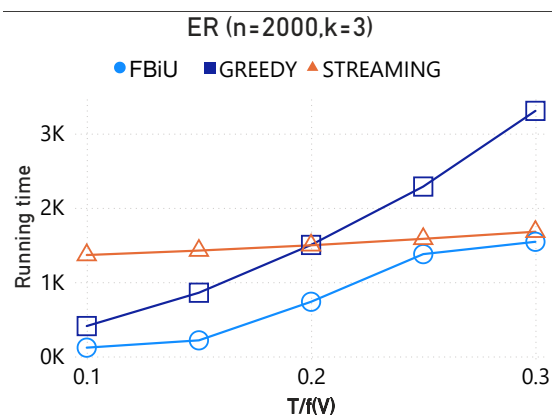
Xét về thời gian thực thi, FBiU vượt trội hơn hẳn so với cả GREEDY và luồng. Trên đồ thị ER, thời gian chạy của FBiU dao động từ 118 đến 1543 giây, thấp hơn đáng kể so với GREEDY (409–3305 giây) và luồng (1365–1679 giây). Trên dữ liệu Email, FBiU chỉ cần từ 1 đến 15 giây, trong khi GREEDY mất 8–87 giây và luồng lên tới 370–600 giây. Nhìn chung, FBiU đạt mức tăng tốc khoảng 3–12 lần so với GREEDY và lên tới 40 lần so với luồng, cho thấy tính hiệu quả cao và đặc biệt phù hợp với các ứng dụng yêu cầu thời gian xử lý nhanh.



(a)

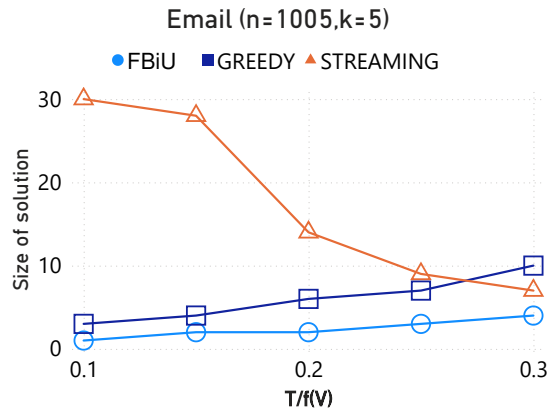


(b)

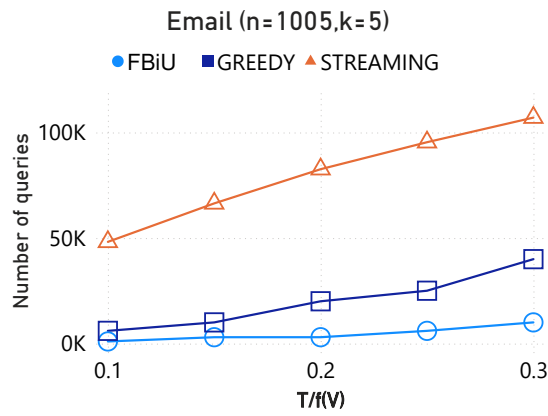


(c)

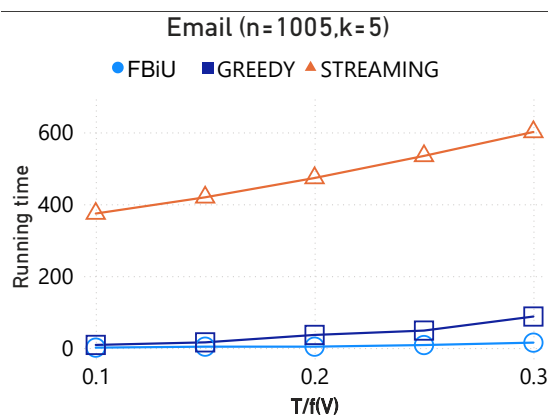
Hình 3.1: So sánh hiệu năng (chi phí, số lượng truy vấn và kích thước nghiệm) giữa FBIU và các phương pháp GREEDY và luồng trên bộ dữ liệu ER.



(a)



(b)



(c)

Hình 3.2: So sánh hiệu năng (giá trị hàm mục tiêu, số lượng truy vấn và kích thước nghiệm) giữa FBIU và các phương pháp GREEDY và luồng trên bộ dữ liệu Email.

3.4. THUẬT TOÁN CHO BÀI TOÁN PHỦ k -SUBMODULAR TỔNG QUÁT

Trong phần này, luận án tập trung xây dựng các thuật toán giải bài toán kSC trong bối cảnh tổng quát, bao gồm: (i) Thuật toán tham lam mở rộng (Extended Greedy), và (ii) thuật toán FBI, được đề xuất nhằm rút ngắn thời gian thực thi của biến thể tham lam mở rộng khi hàm mục tiêu là đơn điệu.

Các phương pháp đề xuất cho thấy sự cải thiện đáng kể về hiệu năng, thể hiện không chỉ qua phân tích lý thuyết mà còn được xác nhận thông qua các kết quả thực nghiệm.

3.4.1. Thuật toán tham lam mở rộng

Ý tưởng của thuật toán được kế thừa từ thuật toán tham lam cổ điển cho bài toán *Submodular Cover* [64] (một trường hợp đặc biệt của kSC), và được tổng quát hóa để xử lý hàm mục tiêu k -submodular. Cụ thể, thay vì chọn một phần tử theo kiểu nhị phân (chọn/không chọn), trong bối cảnh k -submodular mỗi phần tử khi được chọn còn phải được *gán* vào một trong k nhãn loại trừ lẫn nhau. Do đó, tại mỗi bước, thuật toán lựa chọn một cặp (e, i) (phần tử–nhãn) có *mật độ lợi ích* lớn nhất, tức tỉ số giữa độ tăng biên và chi phí, và lặp lại cho đến khi giá trị hàm đạt một ngưỡng xấp xỉ mong muốn.

3.4.1.1. Mô tả thuật toán

Xét một thể hiện (V, g, k, T) , trong đó $g : (k+1)^V \rightarrow \mathbb{R}_+$ là hàm k -submodular và T là ngưỡng cần đạt. Thuật toán nhận thêm tham số chính xác $\epsilon \in (0, 1)$. Thuật toán khởi tạo nghiệm $\mathbf{s} \leftarrow \mathbf{0}$ và xây dựng hàm phụ

$$h(\cdot) \leftarrow \min\{g(\cdot), T\}, \quad \alpha = \begin{cases} \frac{1}{2}, & \text{nếu } h \text{ đơn điệu,} \\ \frac{1}{3}, & \text{nếu } h \text{ không đơn điệu.} \end{cases}$$

Hàm $h(\cdot)$ vẫn là k -submodular khi hàm g đơn điệu và khi g không đơn điệu với điều kiện $g(\cdot) < T$. Sau đó, thuật toán thực hiện tối đa n vòng lặp; ở mỗi vòng lặp, chọn cặp (e, i) (với e chưa nằm trong $\text{supp}(\mathbf{s})$) sao cho mật độ lợi ích biên $\Delta_{(e,i)}h(\mathbf{s})/c(e)$ là lớn nhất, rồi cập nhật $\mathbf{s} \leftarrow \mathbf{s} \sqcup (e, i)$. Thuật toán dừng khi $h(\mathbf{s}) \geq (1 - \epsilon)\alpha T$.

3.4.1.2. Phân tích lý thuyết

Để phân tích bảo đảm nghiệm, luận án sử dụng các ký hiệu sau:

- (e_j, i_j) : cặp phần tử–nhãn được thêm ở lần chọn thứ j trong vòng lặp chính của Thuật toán 3.3.
- $\mathbf{s} = \{(e_1, i_1), \dots, (e_t, i_t)\}$: nghiệm thu được khi kết thúc thuật toán, với $t = |\text{supp}(\mathbf{s})|$.

Thuật toán 3.3: Thuật toán tham lam mở rộng (Extended Greedy) cho bài toán kSC

Input: Thể hiện (V, g, k, T) của bài toán kSC, tham số $\epsilon \in (0, 1)$

Output: Nghiệm $\mathbf{s}$

```

1:  $\mathbf{s} \leftarrow \mathbf{0}$ ;  $\alpha \leftarrow \begin{cases} \frac{1}{2}, & \text{nếu } g \text{ đơn điệu,} \\ \frac{1}{3}, & \text{nếu } g \text{ không đơn điệu} \end{cases}$ 
2: Định nghĩa  $h : (k+1)^V \mapsto \mathbb{R}_+$  bởi  $h(\cdot) \leftarrow \min\{g(\cdot), T\}$ 
3: while  $h(\mathbf{s}) < \alpha(1 - \epsilon)T$  do
4:    $(e, i) \leftarrow \arg \max_{e \in V \setminus \text{supp}(\mathbf{s}), i \in [k]} \frac{\Delta_{(e,i)}h(\mathbf{s})}{c(e)}$ 
5:    $\mathbf{s} \leftarrow \mathbf{s} \sqcup (e, i)$ 
6: end
7: return  $\mathbf{s}$ 

```

- $\mathbf{s}^j = \{(e_1, i_1), \dots, (e_j, i_j)\}$: nghiệm sau khi đã chọn j cặp, $1 \leq j \leq t$; đặt $\mathbf{s}^0 = \mathbf{0}$ và $\mathbf{s}^t = \mathbf{s}$.
- $\mathbf{o}$: nghiệm tối ưu của bài toán.
- $\mathbf{o}^j = (\mathbf{o} \sqcup \mathbf{s}^j) \sqcup \mathbf{s}^j$ và $\mathbf{o}^{j-1/2} = (\mathbf{o} \sqcup \mathbf{s}^j) \sqcup \mathbf{s}^{j-1}$: các nghiệm trung gian phục vụ so sánh giữa $\mathbf{o}$ và $\mathbf{s}^j$.
- $\mathbf{s}^{j-1/2}$: nếu $e_j \in \text{supp}(\mathbf{o})$ thì $\mathbf{s}^{j-1/2} = \mathbf{s}^{j-1} \sqcup (e_j, \mathbf{o}(e_j))$; nếu $e_j \notin \text{supp}(\mathbf{o})$ thì $\mathbf{s}^{j-1/2} = \mathbf{s}^{j-1}$.
- $\mathbf{u}^t = \{(u_1, i_1), \dots, (u_r, i_r)\}$: tập các cặp thuộc $\mathbf{o}^t$ nhưng không thuộc $\mathbf{s}^t$, với $r = |\text{supp}(\mathbf{u}^t)|$.
- $\mathbf{u}_l^t = \mathbf{s}^t \sqcup \{(u_1, i_1), \dots, (u_l, i_l)\}$, $1 \leq l \leq r$ và $\mathbf{u}_0^t = \mathbf{s}^t$.

Bổ đề sau thiết lập mối quan hệ then chốt giữa nghiệm tối ưu và nghiệm tham lam sau j bước.

Bổ đề 3.3. Sau vòng lặp thứ $j \leq t$ của Thuật toán 3.3, ta có:

- Nếu h đơn điệu thì $h(\mathbf{o}) - h(\mathbf{o}^j) \leq h(\mathbf{s}^j)$.
- Nếu h không đơn điệu thì $h(\mathbf{o}) - h(\mathbf{o}^j) \leq 2h(\mathbf{s}^j)$.

Chứng minh. Chứng minh a). Với mọi $0 \leq l \leq t$, ta có

$$h(\mathbf{o}^{l-1}) - h(\mathbf{o}^l) \leq h(\mathbf{o}^{l-1}) - h(\mathbf{o}^{l-1/2}) \quad (3.25)$$

$$\leq h(\mathbf{s}^{l-1/2}) - h(\mathbf{s}^{l-1}) \quad (3.26)$$

$$\leq h(\mathbf{s}^l) - h(\mathbf{s}^{l-1}), \quad (3.27)$$

trong đó (3.25) do tính đơn điệu của h , (3.26) do tính k -submodular của h , và (3.27) do quy tắc chọn tham lam của thuật toán. Cộng về theo $l = 1, \dots, j$ suy ra

$$h(\mathbf{o}) - h(\mathbf{o}^j) = \sum_{l=1}^j (h(\mathbf{o}^{l-1}) - h(\mathbf{o}^l)) \leq \sum_{l=1}^j (h(\mathbf{s}^l) - h(\mathbf{s}^{l-1})) \leq h(\mathbf{s}^j).$$

Chứng minh b). Khi h không đơn điệu, xét hai trường hợp.

Trường hợp 1: $e_l \notin \text{supp}(\mathbf{o})$. Khi đó $\text{supp}(\mathbf{o}^{l-1}) \subset \text{supp}(\mathbf{o}^l)$ và $e_l \in \text{supp}(\mathbf{o}^l) \setminus \text{supp}(\mathbf{o}^{l-1})$. Lấy $p \in [k]$, $p \neq i_l$. Do tính đơn điệu theo cặp (pairwise monotonicity), ta có

$$h(\mathbf{o}^l) + h(\mathbf{o}^{l-1} \sqcup (e_l, p)) - 2h(\mathbf{o}^{l-1}) = \Delta_{(e_l, i_l)} h(\mathbf{o}^{l-1}) + \Delta_{(e_l, p)} h(\mathbf{o}^{l-1}) \geq 0.$$

Suy ra

$$\begin{aligned} h(\mathbf{o}^{l-1}) - h(\mathbf{o}^l) &\leq h(\mathbf{o}^{l-1} \sqcup (e_l, p)) - h(\mathbf{o}^{l-1}) \\ &\leq h(\mathbf{s}^{l-1} \sqcup (e_l, p)) - h(\mathbf{s}^{l-1}) \end{aligned} \quad (3.28)$$

$$\leq h(\mathbf{s}^l) - h(\mathbf{s}^{l-1}), \quad (3.29)$$

trong đó (3.28) do tính k -submodular và (3.29) do quy tắc chọn tham lam.

Trường hợp 2: $e_l \in \text{supp}(\mathbf{o})$. Khi đó $\text{supp}(\mathbf{o}^{l-1}) = \text{supp}(\mathbf{o}^l)$. Nếu $\mathbf{o}^{l-1}(e_l) = \mathbf{o}^l(e_l)$ thì $h(\mathbf{o}^{l-1}) - h(\mathbf{o}^l) = 0 \leq h(\mathbf{s}^l) - h(\mathbf{s}^{l-1})$. Nếu $\mathbf{o}^{l-1}(e_l) \neq \mathbf{o}^l(e_l)$, nhờ tính đơn điệu theo cặp ta có

$$h(\mathbf{o}^l) + h(\mathbf{o}^{l-1}) - 2h(\mathbf{o}^{l-1/2}) = \Delta_{(e_l, \mathbf{o}^{l-1}(e_l))} h(\mathbf{o}^{l-1/2}) + \Delta_{(e_l, \mathbf{o}^l(e_l))} h(\mathbf{o}^{l-1/2}) \geq 0.$$

Suy ra

$$\begin{aligned} h(\mathbf{o}^{l-1}) - h(\mathbf{o}^l) &\leq 2(h(\mathbf{o}^{l-1}) - h(\mathbf{o}^{l-1/2})) \\ &\leq 2(h(\mathbf{s}^l) - h(\mathbf{s}^{l-1})). \end{aligned}$$

Kết hợp hai trường hợp cho mọi l ,

$$h(\mathbf{o}^{l-1}) - h(\mathbf{o}^l) \leq 2(h(\mathbf{s}^l) - h(\mathbf{s}^{l-1})).$$

Cộng theo $l = 1, \dots, j$ suy ra

$$h(\mathbf{o}) - h(\mathbf{o}^j) \leq \sum_{l=1}^j 2(h(\mathbf{s}^l) - h(\mathbf{s}^{l-1})) \leq 2h(\mathbf{s}^j),$$

□

Bằng cách sử dụng cẩn thận Bổ đề 3.3 cùng với cách xây dựng hàm h , ta thu được kết quả về hiệu năng của thuật toán *Tham lam mở rộng* như được phát biểu trong Định lý 3.3 sau đây.

Định lý 3.3. Với mọi $\epsilon \in (0, 1)$, thuật toán 3.3 có độ phức tạp truy vấn $O(n^2k)$ và đạt được các tỷ lệ xấp xỉ tiêu chuẩn kép sau:

- Nếu g là hàm đơn điệu, thuật toán đạt tỷ lệ xấp xỉ

$$\left(\frac{1}{2} \left(\ln \left(\frac{1}{\epsilon} \right) + \frac{1+\epsilon}{\epsilon} \right), \frac{1-\epsilon}{2} \right).$$

- Nếu g là hàm không đơn điệu, thuật toán đạt tỷ lệ xấp xỉ

$$\left(\frac{1}{3} \left(\ln \left(\frac{1}{\epsilon} \right) + \frac{2+\epsilon}{\epsilon} \right), \frac{1-\epsilon}{3} \right).$$

Ngoài ra, trong trường hợp chi phí đồng nhất (tức là $c(e) = 1$ với mọi $e \in V$), thuật toán đạt tỷ lệ xấp xỉ tiêu chuẩn kép

$$\left(\alpha \left(1 + \ln \left(\frac{1}{\epsilon} \right) \right), \alpha(1 - \epsilon) \right),$$

trong đó

$$\alpha = \begin{cases} \frac{1}{2}, & \text{nếu } g \text{ đơn điệu,} \\ \frac{1}{3}, & \text{nếu } g \text{ không đơn điệu.} \end{cases}$$

Chứng minh. Trước hết, ta phân tích độ phức tạp truy vấn của thuật toán. Thuật toán có nhiều nhất n vòng lặp chính. Trong mỗi vòng lặp, để tìm một cặp (e, i) ở Dòng 5, thuật toán cần nhiều nhất nk truy vấn. Do đó, tổng độ phức tạp truy vấn của thuật toán là $O(n^2k)$.

Tiếp theo, ta chứng minh chặn lý thuyết đối với nghiệm thu được.

Trường hợp g là hàm đơn điệu. Khi đó, theo cách xây dựng, hàm h cũng là hàm đơn điệu. Áp dụng Bổ đề 3.3, ta có

$$\begin{aligned} T - h(\mathbf{s}^{t-1}) &\leq h(\mathbf{o}) - h(\mathbf{s}^{t-1}) \\ &= h(\mathbf{o}) - h(\mathbf{o}^{t-1}) + h(\mathbf{o}^{t-1}) - h(\mathbf{s}^{t-1}) \\ &\leq h(\mathbf{s}^{t-1}) + \sum_{e \in \text{supp}(\mathbf{o}^{t-1}) \setminus \text{supp}(\mathbf{s}^t)} \Delta_{(e, \mathbf{o}(e))} h(\mathbf{s}^{t-1}) \\ &\leq h(\mathbf{s}^{t-1}) + \text{opt} \frac{\sum_{e \in \text{supp}(\mathbf{o}^{t-1}) \setminus \text{supp}(\mathbf{s}^{t-1})} \Delta_{(e, \mathbf{o}(e))} h(\mathbf{s}^{t-1})}{\sum_{e \in \text{supp}(\mathbf{o}^{t-1}) \setminus \text{supp}(\mathbf{s}^{t-1})} c(e)} \end{aligned} \quad (3.30)$$

$$\leq h(\mathbf{s}^{t-1}) + \text{opt} \max_{e \in \text{supp}(\mathbf{o}^{t-1}) \setminus \text{supp}(\mathbf{s}^{t-1})} \frac{\Delta_{(e, \mathbf{o}(e))} h(\mathbf{s}^{t-1})}{c(e)} \quad (3.31)$$

$$\leq h(\mathbf{s}^{t-1}) + \text{opt} \frac{h(\mathbf{s}^t) - h(\mathbf{s}^{t-1})}{c(e^t)}. \quad (3.32)$$

Trong đó, phép biến đổi từ (3.30) sang (3.31) dựa trên bất đẳng thức quen thuộc

$$\min_{i=1,\dots,l} \frac{a_i}{b_i} \leq \frac{\sum_{i=1}^l a_i}{\sum_{i=1}^l b_i} \leq \max_{i=1,\dots,l} \frac{a_i}{b_i},$$

đúng với mọi số dương $a_1, \dots, a_l$ và $b_1, \dots, b_l$. Bất đẳng thức (3.32) có được từ quy tắc lựa chọn của thuật toán tham lam mở rộng, nghĩa là thuật toán luôn chọn phần tử có tỷ lệ lợi ích biên trên chi phí lớn nhất. Từ (3.32), ta suy ra

$$h(\mathbf{s}^t) - h(\mathbf{s}^{t-1}) \geq c(e^t) \frac{T - 2h(\mathbf{s}^{t-1})}{\text{opt}}.$$

Do đó

$$T - 2h(\mathbf{s}^t) = T - 2h(\mathbf{s}^{t-1}) - 2(h(\mathbf{s}^t) - h(\mathbf{s}^{t-1})) \quad (3.33)$$

$$\leq T - 2h(\mathbf{s}^{t-1}) - 2c(e^t) \frac{T - 2h(\mathbf{s}^{t-1})}{\text{opt}} \quad (3.34)$$

$$\leq \left(1 - \frac{2c(e^t)}{\text{opt}}\right) (T - 2h(\mathbf{s}^{t-1})) \quad (3.35)$$

$$\leq e^{-\frac{2c(e^t)}{\text{opt}}} (T - 2h(\mathbf{s}^{t-1})). \quad (3.36)$$

Áp dụng liên tiếp bất đẳng thức (3.36), ta thu được

$$T - 2h(\mathbf{s}^t) \leq e^{-\frac{2c(\mathbf{s}^t)}{\text{opt}}} T.$$

Do t là vòng lặp cuối cùng của vòng `while`, theo điều kiện dừng của thuật toán, ta có

$$g(\mathbf{s}) \geq h(\mathbf{s}) = h(\mathbf{s}^t) \geq (1 - \epsilon)T/2 \quad \text{và} \quad h(\mathbf{s}^{t-1}) < (1 - \epsilon)T/2.$$

Kết hợp với bất đẳng thức

$$T - 2h(\mathbf{s}^{t-1}) \leq e^{-\frac{2c(\mathbf{s}^{t-1})}{\text{opt}}} T,$$

ta suy ra

$$c(\mathbf{s}^{t-1}) \leq \frac{\text{opt}}{2} \ln \left(\frac{T}{T - 2h(\mathbf{s}^{t-1})} \right) \quad (3.37)$$

$$\leq \frac{\text{opt}}{2} \ln \left(\frac{1}{\epsilon} \right), \quad \text{do } h(\mathbf{s}^{t-1}) \leq (1 - \epsilon)T/2. \quad (3.38)$$

Mặt khác, từ (3.32) ta có thể chặn chi phí của phần tử được chọn ở vòng lặp cuối

như sau:

$$\begin{aligned}
c(e^t) &\leq \frac{h(\mathbf{s}^t) - h(\mathbf{s}^{t-1})}{T - 2h(\mathbf{s}^{t-1})} \text{opt} \\
&\leq \frac{T - h(\mathbf{s}^{t-1})}{T - 2h(\mathbf{s}^{t-1})} \text{opt} \quad (\text{vì } h(\mathbf{s}^t) \leq T) \\
&\leq \left(1 + \frac{h(\mathbf{s}^{t-1})}{T - 2h(\mathbf{s}^{t-1})}\right) \text{opt} \\
&\leq \left(1 + \frac{\frac{(1-\epsilon)T}{2}}{T - (1-\epsilon)T}\right) \text{opt} \quad (\text{vì } h(\mathbf{s}^{t-1}) \leq (1-\epsilon)T/2) \\
&\leq \frac{1+\epsilon}{2\epsilon} \text{opt}.
\end{aligned}$$

Do đó, tổng chi phí của nghiệm cuối cùng được chặn bởi

$$c(\mathbf{s}^t) = c(\mathbf{s}^{t-1}) + c(e^t) \leq \left(\ln\left(\frac{1}{\epsilon}\right) + \frac{1+\epsilon}{\epsilon}\right) \frac{\text{opt}}{2}.$$

Xét tiếp trường hợp chi phí đồng nhất, tức là $c(e) = 1$ với mọi $e \in V$. Khi đó, việc phân tích trở nên đơn giản hơn. Nếu $\text{opt} = 1$ thì thuật toán luôn tìm được nghiệm tối ưu, vì nó chọn phần tử có lợi ích biên lớn nhất.

Nếu $\text{opt} \geq 2$, từ (3.38) ta có

$$|supp(\mathbf{s}^t)| = |supp(\mathbf{s}^{t-1})| + 1 \leq \ln\left(\frac{1}{\epsilon}\right) \frac{\text{opt}}{2} + 1 \leq \left(\ln\left(\frac{1}{\epsilon}\right) + 1\right) \frac{\text{opt}}{2}.$$

Trường hợp g là hàm không đơn điệu. Khi đó h cũng là hàm không đơn điệu. Ta điều chỉnh Bổ đề 3.3 và sử dụng cách lập luận tương tự như trong trường hợp đơn điệu. Áp dụng Bổ đề 3.3, ta được

$$\begin{aligned}
T - h(\mathbf{s}^t) &\leq h(\mathbf{o}) - h(\mathbf{s}^t) \\
&= h(\mathbf{o}) - h(\mathbf{o}^t) + h(\mathbf{o}^t) - h(\mathbf{s}^t) \\
&\leq 2h(\mathbf{s}^t) + \sum_{e \in supp(\mathbf{o}^t) \setminus supp(\mathbf{s}^t)} \Delta_{(e, \mathbf{o}(e))} h(\mathbf{s}^t) \\
&\leq 2h(\mathbf{s}^t) + \text{opt} \frac{h(\mathbf{s}^{t+1}) - h(\mathbf{s}^t)}{c(e^{t+1})}.
\end{aligned} \tag{3.39}$$

Biến đổi tương tự với trường hợp g đơn điệu ta được

$$\begin{aligned}
T - 3h(\mathbf{s}^t) &\leq \left(1 - \frac{3c(e^t)}{\text{opt}}\right) (T - 3h(\mathbf{s}^{t-1})) \\
&\leq e^{-\frac{3c(e^t)}{\text{opt}}} (T - 3h(\mathbf{s}^{t-1})) \\
&\leq e^{-\frac{3c(\mathbf{s}^{t-1})}{\text{opt}}} T.
\end{aligned}$$

Mặt khác, ta có

$$g(\mathbf{s}) = h(\mathbf{s}) = h(\mathbf{s}^t) \geq (1 - \epsilon)T/3 \quad \text{và} \quad h(\mathbf{s}^{t-1}) < (1 - \epsilon)T/3.$$

Từ đó suy ra

$$T - 3h(\mathbf{s}^{t-1}) \leq e^{-\frac{3c(\mathbf{s}^{t-1})}{\text{opt}}} T,$$

và do đó

$$\begin{aligned} c(\mathbf{s}^{t-1}) &\leq \frac{\text{opt}}{3} \ln \left(\frac{T}{T - 3h(\mathbf{s}^{t-1})} \right) \\ &\leq \frac{\text{opt}}{3} \ln \left(\frac{1}{\epsilon} \right), \quad \text{do } h(\mathbf{s}^{t-1}) \leq (1 - \epsilon)T/3. \end{aligned}$$

Tương tự, từ (3.39) ta có

$$\begin{aligned} c(e^t) &\leq \frac{h(\mathbf{s}^t) - h(\mathbf{s}^{t-1})}{T - 3h(\mathbf{s}^{t-1})} \text{opt} \\ &\leq \frac{T - h(\mathbf{s}^{t-1})}{T - 3h(\mathbf{s}^{t-1})} \text{opt} \quad (\text{vì } h(\mathbf{s}^t) \leq T) \\ &\leq \left(1 + \frac{2h(\mathbf{s}^{t-1})}{T - 3h(\mathbf{s}^{t-1})} \right) \text{opt} \\ &\leq \left(1 + \frac{2(1 - \epsilon)T/3}{T - 3(1 - \epsilon)T/3} \right) \text{opt} \quad (\text{vì } h(\mathbf{s}^{t-1}) \leq (1 - \epsilon)T/3) \\ &\leq \frac{2 + \epsilon}{3\epsilon} \text{opt}. \end{aligned}$$

Vì vậy, tổng chi phí của nghiệm cuối cùng được chặn bởi

$$c(\mathbf{s}^t) = c(\mathbf{s}^{t-1}) + c(e^t) \leq \left(\ln \left(\frac{1}{\epsilon} \right) + \frac{2 + \epsilon}{\epsilon} \right) \frac{\text{opt}}{3}.$$

Trong trường hợp chi phí đồng nhất, ta tiếp tục phân tích tương tự như trong trường hợp đơn điệu. Nếu $\text{opt} \leq 2$, thuật toán luôn chọn phần tử đơn có giá trị f lớn nhất, ký hiệu là $(e_{\max}, i_{\max})$. Khi đó,

$$T \leq h(\mathbf{o}) \leq 2h(e_{\max}, i_{\max}) \leq 2h(\mathbf{s}^1),$$

suy ra $g(\mathbf{s}^1) \geq h(\mathbf{s}^1) \geq T/2$. Vì vậy, thuật toán dừng ngay sau vòng lặp đầu tiên và trả về một nghiệm thỏa mãn

$$|\text{supp}(\mathbf{s}^t)| = 1 \leq \text{opt}.$$

Nếu $\text{opt} \geq 3$, từ lập luận trên ta thu được

$$|\text{supp}(\mathbf{s}^t)| = |\text{supp}(\mathbf{s}^{t-1})| + 1 \leq \ln \left(\frac{1}{\epsilon} \right) \frac{\text{opt}}{3} + \frac{\text{opt}}{3} \leq \left(\ln \left(\frac{1}{\epsilon} \right) + 1 \right) \frac{\text{opt}}{3}.$$

□

Các phân tích trên cho thấy một số đặc điểm quan trọng của thuật toán *Tham lam mở rộng* đối với bài toán kSC:

- *Ý tưởng phân tích dựa trên sự suy giảm theo hàm mũ.* Một kết quả then chốt trong quá trình chứng minh là việc thiết lập được bất đẳng thức

$$T - \beta h(\mathbf{s}^t) \leq e^{-\frac{\beta c(\mathbf{s}^t)}{\text{opt}}} T,$$

trong đó $\beta = 2$ đối với trường hợp hàm mục tiêu đơn điệu và $\beta = 3$ đối với trường hợp không đơn điệu. Bất đẳng thức này cho thấy khoảng cách giữa nghiệm hiện tại và ngưỡng mục tiêu T giảm theo hàm mũ theo tổng chi phí của nghiệm đã chọn. Đây là cơ sở để suy ra các chặn trên đối với chi phí của nghiệm thu được.

- *Vai trò của quy tắc lựa chọn theo mật độ lợi ích.* Tại mỗi vòng lặp, thuật toán lựa chọn phần tử có tỷ lệ giữa lợi ích biên và chi phí lớn nhất. Quy tắc này bảo đảm mỗi đơn vị chi phí bỏ ra mang lại mức cải thiện lớn nhất cho hàm mục tiêu, đồng thời tạo cầu nối giữa nghiệm hiện tại và nghiệm tối ưu trong quá trình phân tích lý thuyết.
- *Sự khác biệt giữa trường hợp đơn điệu và không đơn điệu.* Đối với hàm mục tiêu đơn điệu, tính chất tăng của hàm cho phép đạt được bất đẳng thức mạnh hơn với hệ số $\beta = 2$, từ đó dẫn đến tỷ lệ xấp xỉ tốt hơn. Ngược lại, trong trường hợp không đơn điệu, việc kiểm soát lợi ích biên trở nên khó khăn hơn và dẫn tới hệ số $\beta = 3$, phản ánh độ khó cao hơn của bài toán.
- *Trường hợp chi phí đồng nhất.* Khi mọi phần tử có cùng chi phí, việc phân tích được đơn giản hóa đáng kể do chi phí của nghiệm tỷ lệ thuận với số lượng phần tử được chọn. Nhờ đó, có thể thiết lập trực tiếp các chặn đối với kích thước nghiệm mà vẫn duy trì được các đảm bảo lý thuyết về chất lượng lời giải.
- *Ưu điểm của phương pháp đề xuất.* Thuật toán *Tham lam mở rộng* đạt được các đảm bảo xấp xỉ tiêu chuẩn kép cho cả hai trường hợp hàm mục tiêu đơn điệu và không đơn điệu. Đồng thời, độ phức tạp truy vấn chỉ ở mức đa thức $O(n^2k)$, thấp hơn đáng kể so với nhiều phương pháp trước đây. Bên cạnh đó, thuật toán có cấu trúc đơn giản, dễ triển khai và có khả năng mở rộng tốt trên các bộ dữ liệu lớn.

3.4.2. Thuật toán xấp xỉ nhanh FBi cho hàm đơn điệu

Mặc dù Thuật toán tham lam mở rộng đạt được bảo đảm xấp xỉ tốt, độ phức tạp truy vấn $O(n^2k)$ có thể trở nên tốn kém trên dữ liệu lớn. Vì vậy, luận án đề xuất thuật toán **Xấp xỉ nhanh (Fast Bicriteria Approximation Algorithm, ký hiệu FBi)** cho trường

hợp hàm ràng buộc lợi ích g đơn điệu và $T \leq \max_{\mathbf{x}' \in (k+1)^V} g(\mathbf{x}')$. Thuật toán FBi đạt xấp xỉ gần tương đương Tham lam mở rộng nhưng chỉ cần $O(nk \log n)$ truy vấn (bỏ qua các hằng số phụ thuộc ϵ, δ), qua đó giảm mạnh chi phí tính toán.

Một kỹ thuật phổ biến để tăng tốc tham lam là dùng *threshold greedy* kết hợp với việc đoán giá trị tối ưu (chẳng hạn qua dãy $\{(1 + \epsilon)^i\}$ trên miền $[c_{\min}, c(V)]$) [24]. Tuy nhiên, cách làm này có thể cần

$$\frac{1}{\epsilon} \log \left(\frac{c(V)}{c_{\min}} \right) = O \left(\frac{1}{\epsilon} \log \left(c \frac{c_{\max}}{c_{\min}} \right) \right)$$

lần đoán. Vì $c_{\max}, c_{\min}$ là các số thực dương bất kỳ nên tỉ lệ $c_{\max}/c_{\min}$ có thể lớn bất kỳ và do vậy số lần đoán đã nêu không phải là đa thức theo kích thước dữ liệu (tức là đa thức theo n và k , tức là $\text{poly}(n, k)$).

Để khắc phục, luận án xây dựng một khung thuật toán gồm 2 thành phần: (i) trích chọn từ V hai tập con quan trọng V_0 (gồm các phần tử chi phí rất nhỏ) và V' (gồm các phần tử “triển vọng” để ước lượng giá trị tối ưu với số lần đoán chỉ cỡ $O(\log n)$); (ii) kết hợp TTX cho bài toán tối đa hóa k -submodular không ràng buộc [69] với phương pháp tham lam ngưỡng [24] để sinh các nghiệm ứng viên chất lượng. Để trình bày rõ ràng, trước hết luận án mô tả biến thể FBiOpt giả sử có một ước lượng v thỏa $(1 - \epsilon)v \leq \text{opt} \leq v$, sau đó trình bày phiên bản đầy đủ FBi loại bỏ giả định này.

3.4.2.1. Thuật toán FBiOpt (biết trước ước lượng tối ưu)

a. Mô tả thuật toán

Thuật toán FBiOpt nhận (V, g, k, T) , một ước lượng v thỏa $(1 - \epsilon)v \leq \text{opt} \leq v$, và hai tham số $\epsilon \in (0, 1/2)$, $\delta \in (0, 1)$. Thuật toán chạy tối đa $\lceil \ln(1/\delta)/\epsilon \rceil$ vòng lặp. Mỗi vòng lặp quét một lần tập V và chỉ thêm các phần tử có mật độ lợi ích vượt ngưỡng động

$$\theta \leftarrow \frac{(1 - \epsilon)(T - 2h(\mathbf{s}))}{v}.$$

Thuật toán dừng sớm nếu đạt $h(\mathbf{s}) \geq (1 - \delta)T/2$.

b. Phân tích lý thuyết

Tương tự như thuật toán tham lam mở rộng, đặt (e_j, i_j) là cặp được thêm thứ j và $\mathbf{s}^j$ là nghiệm sau j lần thêm; đặt $\mathbf{s}_j$ là nghiệm sau vòng lặp ngoài thứ j của Thuật toán 3.4. Các ký hiệu $\mathbf{o}^j, \mathbf{o}^{j-1/2}, \mathbf{u}^t, \mathbf{u}_i^t$ được định nghĩa tương tự.

Bổ đề 3.4. Sau vòng lặp ngoài thứ j của Thuật toán 3.4, ta có:

a) Nếu h đơn điệu thì $h(\mathbf{o}) - h(\mathbf{o}^j) \leq h(\mathbf{s}_j)$.

b) Nếu h không đơn điệu thì $h(\mathbf{o}) - h(\mathbf{o}^j) \leq 2h(\mathbf{s}_j)$.

Thuật toán 3.4: TTX nhanh với ước lượng tối ưu (FBiOpt)

Input: Thể hiện (V, g, k, T) , $\delta \in (0, 1)$, ước lượng v của lời giải tối ưu opt

Output: $\mathbf{s}$

```

1:  $\mathbf{s} \leftarrow \mathbf{0}$ 
2: Định nghĩa  $h : (k+1)^V \mapsto \mathbb{R}_+$  bởi  $h(\cdot) \leftarrow \min\{g(\cdot), T\}$ 
3: for  $l = 1$  to  $\lceil \ln(1/\delta)/\epsilon \rceil$  do
4:    $\theta \leftarrow (1 - \epsilon)(T - 2h(\mathbf{s}))/v$ 
5:   foreach  $e \in V \setminus \text{supp}(\mathbf{s})$  do
6:      $i \leftarrow \arg \max_{i \in [k]} \Delta_{(e,i)} h(\mathbf{s})$ 
7:     if  $\frac{\Delta_{(e,i)} h(\mathbf{s})}{c(e)} \geq \theta$  then
8:        $\mathbf{s} \leftarrow \mathbf{s} \sqcup (e, i)$ 
9:     end
10:    if  $h(\mathbf{s}) \geq (1 - \delta)T/2$  then
11:      return  $\mathbf{s}$ 
12:    end
13:  end
14: end
15: return  $\mathbf{s}$ 

```

Chứng minh. Chứng minh tương tự Bổ đề 3.3, do quy tắc chọn nhãn i (tối đa hóa độ tăng biên với phần tử e đang xét) trong Thuật toán 3.4 tương đồng với Thuật toán 3.3. $\square$

Định lý 3.4. Với mọi $\epsilon \in (0, 1/2)$, $\delta \in (0, 1)$, hàm g đơn điệu và giả sử $v(1 - \epsilon) \leq \text{opt} \leq v$, thuật toán 3.4 thực hiện nhiều nhất

$$O\left(\frac{nk}{\epsilon} \log\left(\frac{1}{\delta}\right)\right)$$

truy vấn và trả về một nghiệm đạt tỷ lệ xấp xỉ tiêu chuẩn kép

$$\left(\frac{1}{2}(1 + 6\epsilon) \left(\ln\left(\frac{1}{\delta}\right) + \frac{1 + \delta}{\delta}\right), \frac{1 - \delta}{2}\right).$$

Chứng minh. Thuật toán 3.4 bao gồm hai vòng lặp. Vòng lặp ngoài thực hiện $O\left(\frac{1}{\epsilon} \log\left(\frac{1}{\delta}\right)\right)$ lần lặp và trong mỗi lần lặp cần tối đa nk truy vấn tới hàm h . Do đó tổng số truy vấn của thuật toán nhiều nhất là $O\left(\frac{nk}{\epsilon} \log\left(\frac{1}{\delta}\right)\right)$. Tiếp theo, ta chứng minh đảm bảo xấp xỉ của thuật toán. Xét hai trường hợp sau.

Trường hợp 1. Giả sử vòng lặp chính kết thúc tại vòng lặp

$$l = \left\lceil \frac{1}{\epsilon} \ln\left(\frac{1}{\delta}\right) \right\rceil.$$

Ký hiệu $\mathbf{s}^{t_j}$ là $\mathbf{s}_j$ sau khi kết thúc vòng lặp thứ $j \leq l$ của vòng lặp chính. Do $h(\cdot)$ là hàm đơn điệu và có tính chất orthant-submodular, mọi phần tử không được chọn tại vòng lặp t đều có mật độ lợi ích nhỏ hơn θ . Do đó ta có

$$\begin{aligned} T - h(\mathbf{s}_j) &\leq h(\mathbf{o}) - h(\mathbf{s}_j) \\ &= h(\mathbf{o}) - h(\mathbf{o}^{t_j}) + h(\mathbf{o}^{t_j}) - h(\mathbf{s}^{t_j}) \end{aligned} \quad (3.40)$$

$$\leq h(\mathbf{s}^{t_j}) + \sum_{e \in \text{supp}(\mathbf{o}^{t_j}) \setminus \text{supp}(\mathbf{s}^{t_j})} \Delta_{(e, \mathbf{o}(e))} h(\mathbf{s}^{t_j}) \quad (3.41)$$

$$< h(\mathbf{s}^{t_j}) + \sum_{e \in \text{supp}(\mathbf{o}^{t_j}) \setminus \text{supp}(\mathbf{s}^{t_j})} \theta \cdot c(e) \quad (3.42)$$

$$\leq h(\mathbf{s}^{t_j}) + \text{opt} \frac{(1 - \epsilon)(T - 2h(\mathbf{s}_{j-1}))}{v} \quad (3.43)$$

$$\leq h(\mathbf{s}_j) + (1 - \epsilon)(T - 2h(\mathbf{s}_{j-1})). \quad (3.44)$$

Trong đó phép biến đổi từ (3.40) sang (3.41) sử dụng Bổ đề 3.4. Áp dụng lặp lại bất đẳng thức trên, ta thu được

$$T - 2h(\mathbf{s}_l) \leq (1 - \epsilon)^l T \leq e^{-\epsilon l} T \leq \delta T. \quad (3.45)$$

Do đó

$$g(\mathbf{s}_l) \geq h(\mathbf{s}_l) \geq (1 - \delta)T/2.$$

Trong trường hợp này, thuật toán không thỏa điều kiện dừng của thuật toán tại vòng lặp $l - 1$, suy ra $h(\mathbf{s}_{l-1}) < (1 - \delta)T/2$. Mặt khác, theo quy tắc lựa chọn phần tử mới đưa vào $\mathbf{s}_i$, ta có

$$h(\mathbf{s}_i) - h(\mathbf{s}_{i-1}) \geq \sum_{e \in \text{supp}(\mathbf{s}_i) \setminus \text{supp}(\mathbf{s}_{i-1})} \Delta_{(e, i_e)} h(\mathbf{s}^{<e}) \quad (3.46)$$

$$\geq (c(\mathbf{s}_i) - c(\mathbf{s}_{i-1}))\theta \quad (3.47)$$

$$= \frac{(c(\mathbf{s}_i) - c(\mathbf{s}_{i-1}))(1 - \epsilon)(T - 2h(\mathbf{s}_{i-1}))}{v}. \quad (3.48)$$

Ở đây $\mathbf{s}^{<e}$ biểu diễn tập $\mathbf{s}$ tại thời điểm ngay trước khi phần tử e được thêm vào nghiệm.

Sắp xếp lại bất đẳng thức (3.48) ta được

$$T - 2h(\mathbf{s}_i) \leq \left(1 - \frac{2(1 - \epsilon)(c(\mathbf{s}_i) - c(\mathbf{s}_{i-1}))}{v}\right)(T - 2h(\mathbf{s}_{i-1})) \quad (3.49)$$

$$\leq e^{-\frac{2(1 - \epsilon)(c(\mathbf{s}_i) - c(\mathbf{s}_{i-1}))}{v}}(T - 2h(\mathbf{s}_{i-1})). \quad (3.50)$$

Áp dụng lặp lại bất đẳng thức trên với $i \leq l - 1$ ta có

$$\delta T < T - 2h(\mathbf{s}_{l-1}) \leq e^{-\frac{2(1 - \epsilon)c(\mathbf{s}_{l-1})}{v}} T \leq e^{-\frac{2(1 - \epsilon)^2 c(\mathbf{s}_{l-1})}{\text{opt}}} T. \quad (3.51)$$

Suy ra $c(\mathbf{s}_{l-1}) \leq \frac{\text{opt}}{2(1 - \epsilon)^2} \ln\left(\frac{1}{\delta}\right)$. Mặt khác, từ (3.48) và định nghĩa của hàm $h(\cdot)$ ta có thể chặn chi phí tăng thêm ở vòng lặp cuối:

$$c(\mathbf{s}_l) - c(\mathbf{s}_{l-1}) \leq \frac{h(\mathbf{s}_l) - h(\mathbf{s}_{l-1})}{T - 2h(\mathbf{s}_{l-1})} \frac{v}{1 - \epsilon} \quad (3.52)$$

$$\leq \frac{T - h(\mathbf{s}_{l-1})}{T - 2h(\mathbf{s}_{l-1})} \frac{v}{1 - \epsilon} \quad (3.53)$$

$$< \left(1 + \frac{(1 - \delta)T/2}{T - (1 - \delta)T}\right) \frac{v}{1 - \epsilon} \quad (3.54)$$

$$\leq \frac{1 + \delta}{2\delta(1 - \epsilon)^2} \text{opt} \quad (\text{Do } v \leq \frac{\text{opt}}{1 - \epsilon}) \quad (3.55)$$

Do đó

$$c(\mathbf{s}_l) \leq \frac{\text{opt}}{2(1 - \epsilon)^2} \ln\left(\frac{1}{\delta}\right) + \frac{1 + \delta}{2\delta(1 - \epsilon)^2} \text{opt} \quad (3.56)$$

$$= \frac{1}{2(1 - \epsilon)^2} \left(\ln\left(\frac{1}{\delta}\right) + \frac{1 + \delta}{\delta} \right) \text{opt} \quad (3.57)$$

$$\leq \frac{1}{2}(1 + 6\epsilon) \left(\ln\left(\frac{1}{\delta}\right) + \frac{1 + \delta}{\delta} \right) \text{opt}. \quad (3.58)$$

Bất đẳng thức cuối cùng suy ra từ $\frac{1}{1-\epsilon} \leq 1 + 2\epsilon$, $\frac{1}{(1-\epsilon)^2} \leq 1 + 6\epsilon$, với $\epsilon \in (0, 1/2)$.

Trường hợp 2. Giả sử vòng lặp chính kết thúc sớm hơn, tức là $l < \frac{1}{\epsilon} \ln\left(\frac{1}{\delta}\right)$, tức là thuật toán thỏa điều kiện dừng ở vòng lặp l . Khi đó

$$h(\mathbf{s}_l) \geq (1 - \delta)T/2, \quad h(\mathbf{s}_{l-1}) < (1 - \delta)T/2.$$

Tương tự như lập luận trước đó, ta vẫn có

$$c(\mathbf{s}_{l-1}) \leq \frac{\text{opt}}{2(1-\epsilon)^2} \ln\left(\frac{1}{\delta}\right)$$

và

$$c(\mathbf{s}_l) - c(\mathbf{s}_{l-1}) \leq \frac{1 + \delta}{2\delta(1-\epsilon)} \text{opt}.$$

Do đó cũng suy ra chặn về chi phí của lời giải. Kết hợp hai trường hợp trên, ta thu được đảm bảo xấp xỉ của nghiệm trả về bởi thuật toán. $\square$

3.4.2.2. Thuật toán FBI

Tiếp theo, luận án trình bày thuật toán FBI (Thuật toán 3.5) bằng cách loại bỏ giả định biết trước v . Thuật toán gồm hai pha.

Pha 1 (trích chọn tập con). Từ V , thuật toán xây dựng hai tập con:

- V_0 : tập các phần tử có chi phí rất nhỏ, sao cho tổng chi phí $c(V_0)$ bị chặn bởi một hằng số tỉ lệ ϵopt ;
- V' : tập các phần tử “triển vọng” có chi phí nằm trong một khoảng hợp lý để có thể tìm ước lượng của giá trị tối ưu với số lần đoán cỡ $O(\log n)$.

Thuật toán khởi tạo lời giải ban đầu $\mathbf{x} \leftarrow \mathbf{0}$, sau đó tìm lời giải cho bài toán Tối đa hàm k -submodular (unconstrained submodular maximization, ký hiệu UkSM), tức là tìm lời giải $\mathbf{x} \in (k+1)^V$ sao cho $g(\mathbf{x})$ lớn nhất. Do đó $g(\mathbf{x}) \geq T$. Sau khi áp dụng TTXX với tỷ lệ $1/2$ với khung đề xuất trong [69] và cụ thể đối với thuật toán tham lam trong [11]. Thuật toán [11] tìm lời giải cho bài toán tối đa hàm đơn điệu k -submodular với ràng buộc kích thước. Thuật toán này hoàn toàn có thể áp dụng tìm lời giải cho bài toán UkSM với kích thước đầu vào là n với hệ số tắt định là $1/2$. Đây cũng chính là lý do thuật toán FBI áp dụng được với điều kiện $T \leq \max_{\mathbf{x}' \in (k+1)^V} g(\mathbf{x}')$.

Tiếp theo, thuật toán lần lượt gán mỗi phần tử $e \in V \setminus \text{supp}(\mathbf{x})$ vào một vị trí được chọn ngẫu nhiên $i \in [k]$ và cập nhật $\mathbf{x} \leftarrow \mathbf{x} \sqcup (e, i)$. Bước này nhằm bảo đảm hai mục đích:

- Do g là hàm đơn điệu nên $g(\mathbf{x} \sqcup (e, i)) \geq g(\mathbf{x})$. Vì vậy, việc bổ sung các phần tử vào $\mathbf{x}$ không làm giảm giá trị hàm mục tiêu.

- Sau khi tất cả các phần tử của V được gán vào $\mathbf{x}$, ta có $\text{supp}(\mathbf{x}) = V$. Nếu $\mathbf{x}$ thỏa mãn ngưỡng yêu cầu thì $\mathbf{x}$ là một nghiệm khả thi. Do đó, chi phí của $\mathbf{x}$ cung cấp một chặn trên cho chi phí của nghiệm tối ưu.

Sau đó, thuật toán sắp xếp các phần tử theo chi phí không giảm dần tức là $\text{supp}(\mathbf{x}) = \{e_1, \dots, e_n\}$ với $c(e_1) \leq c(e_2) \leq \dots \leq c(e_n)$, sau đó tìm

$$j \leftarrow \min \left\{ i : g(\{(e_1, \mathbf{x}(e_1)), \dots, (e_i, \mathbf{x}(e_i))\}) \geq T \right\}.$$

Phần tử j luôn tồn tại vì sau khi áp dụng thuật toán trong [69], thuật toán luôn tìm thấy $\mathbf{x}$ sao cho $g(\mathbf{x}) \geq T$. Từ đó suy ra hai ngưỡng chi phí $c_m = \epsilon' c(e_j)/n$ và $c_M = j c(e_j)$ để xác định V_0 và V' . Cuối cùng, Pha này trả về các giá trị V_0, V' là các tập nền mới để tìm lời giải cho Pha sau.

Pha 2 (sinh tập nghiệm ứng viên và chọn nghiệm tốt nhất).

Thuật toán trước hết tìm một nghiệm nền $\mathbf{x}_0$ trên V_0 bằng các áp dụng thuật toán [69] như một ứng viên tiềm năng. Định nghĩa hàm biên $g'(\cdot) = g(\mathbf{x}_0 \sqcup \cdot) - g(\mathbf{x}_0)$ và ngưỡng còn lại T' . Tiếp theo, thuật toán gọi FBiOpt nhiều lần trên miền V' với các giá trị đoán $v_i = (c_m)/(1 - \epsilon')^i, \left\lceil \log_{\frac{1}{1-\epsilon'}} \left(\frac{jn^2}{\epsilon'} \right) \right\rceil$ để tìm các nghiệm ứng viên $\mathbf{s}_i$. Chú ý rằng với việc duyệt v_i trong khoảng trên sẽ tìm “quét” hết các chi phí trong V' và do vậy luôn đưa một ước lượng của lời giải tối ưu đối với (V', g', T') với sai số không quá $1 - \epsilon'$ lần. Cuối cùng, thuật toán chọn nghiệm có chi phí nhỏ nhất thỏa ngưỡng và trả về nghiệm kết hợp với $\mathbf{x}_0$.

3.4.2.3. Phân tích lý thuyết

Đặt $V_1 = V \setminus (V_0 \cup V')$. Bổ đề sau cho thấy $\mathbf{o}$ không chứa phần tử trong V_1 , đồng thời tổng chi phí của V_0 là rất nhỏ.

Bổ đề 3.5. Sau khi kết thúc dòng 8 trong Thuật toán 3.5, V_0, V' được xác định. Khi đó ta có $\text{supp}(\mathbf{o}) \cap V_1 = \emptyset$ và $c(V_0) \leq \epsilon' \text{opt}$.

Chứng minh. Sau bước “lắp đầy”, ta có $\text{supp}(\mathbf{x}) = V$ và dãy chi phí $c(e_1) \leq \dots \leq c(e_n)$. Chỉ số j là nhỏ nhất sao cho

$$g(\{(e_1, \mathbf{x}(e_1)), \dots, (e_j, \mathbf{x}(e_j))\}) \geq T.$$

Trước hết chứng minh $c(e_j) \leq c(o_p)$, với $o_p = \arg \max_{o \in \text{supp}(\mathbf{o})} c(o)$ là phần tử đắt nhất trong nghiệm tối ưu. Giả sử ngược lại $c(e_j) > c(o_p)$. Vì $\text{supp}(\mathbf{o}) \subseteq V = \text{supp}(\mathbf{x})$, tồn tại $j' < j$ sao cho $\text{supp}(\mathbf{o}) \subseteq \{e_1, \dots, e_{j'}\}$. Đặt

$$\mathbf{x}^{j'} = \{(e_1, \mathbf{x}(e_1)), \dots, (e_{j'}, \mathbf{x}(e_{j'}))\}.$$

Thuật toán 3.5: Thuật toán FBi (Fast Bicriteria)

Input: Thể hiện (V, g, k, T) , g đơn điệu; $\epsilon \in (0, 1/2)$, $\delta \in (0, 1)$

Output: Nghiệm $\mathbf{s}$

- 1: $\mathbf{x} \leftarrow \mathbf{0}$; $\epsilon' \leftarrow \epsilon/8$, $\mathbf{x} \leftarrow$ nghiệm của UkSM trên (g, V, k) bằng TTXX 1/2 trong [69]
 - 2: **foreach** $e \in V \setminus \text{supp}(\mathbf{x})$ **do**
 - 3: | chọn ngẫu nhiên $i \in [k]$ và $\mathbf{x} \leftarrow \mathbf{x} \sqcup (e, i)$
 - 4: **end**
 - 5: Sắp xếp các phần tử trong $\text{supp}(\mathbf{x}) = \{e_1, \dots, e_n\}$ theo thứ tự chi phí không giảm
 - 6: $j \leftarrow \min \left\{ i : g(\{(e_1, \mathbf{x}(e_1)), \dots, (e_i, \mathbf{x}(e_i))\}) \geq T \right\}$
 - 7: $c_m \leftarrow \epsilon' c(e_j)/n$; $c_M \leftarrow j c(e_j)$
 - 8: $V_0 \leftarrow \{e \in V : c(e) < c_m\}$; $V' \leftarrow \{e \in V : c_m \leq c(e) \leq c_M\}$
 - 9: $\mathbf{x}_0 \leftarrow$ nghiệm của UkSM trên (g, V_0, k) bằng TTXX 1/2 trong [11] với thiết lập kích thước bằng n
 - 10: $T_0 \leftarrow g(\mathbf{x}_0)$; định nghĩa $g'(\cdot) \leftarrow g(\mathbf{x}_0 \sqcup \cdot) - g(\mathbf{x}_0)$
 - 11: $T' \leftarrow T - 2T_0$
 - 12: **for** $i = 0$ **to** $\left\lceil \log_{\frac{1}{1-\epsilon'}} \left(\frac{jn^2}{\epsilon'} \right) \right\rceil$ **do**
 - 13: | $v_i \leftarrow c_m/(1 - \epsilon')^i$
 - 14: | $\mathbf{s}_i \leftarrow$ kết quả của FBiOpt với đầu vào $(V', g', k, T', \epsilon', \delta, v_i)$
 - 15: **end**
 - 16: $\mathbf{s} \leftarrow \arg \min \left\{ c(\mathbf{s}_i) : g(\mathbf{s}_i \sqcup \mathbf{x}_0) \geq (1 - \delta)T/2 \right\}$
 - 17: **return** $\mathbf{s} \sqcup \mathbf{x}_0$
-

Do tính đơn điệu của g ,

$$T \leq g(\mathbf{o}) \leq g(\mathbf{x}^{j'}) \leq g(\mathbf{x}^j),$$

mâu thuẫn với tính tối tiểu của j . Vậy $c(e_j) \leq c(o_p) \leq \text{opt}$.

Từ định nghĩa j , ta cũng có $\text{opt} \leq \sum_{i=1}^j c(e_i) \leq j c(e_j)$. Mọi phần tử trong V_1 có chi phí $> c_M = j c(e_j) \geq \text{opt}$, nên không thể thuộc $\text{supp}(\mathbf{o})$, tức $\text{supp}(\mathbf{o}) \cap V_1 = \emptyset$. Cuối cùng, với mọi $e \in V_0$ ta có $c(e) < c_m = \epsilon' c(e_j)/n$ nên

$$c(V_0) \leq |V_0| \cdot \frac{\epsilon' c(e_j)}{n} \leq \epsilon' c(e_j) \leq \epsilon' \text{opt},$$

□

Định lý 3.5. Với mọi $\epsilon \in (0, 1/2)$ và $\delta \in (0, 1)$, giả sử g là một hàm đơn điệu và k -submodular. Khi đó, Thuật toán 3.5 trả về một nghiệm đạt tỷ lệ xấp xỉ tiêu chuẩn kép

$$\left(\frac{1}{2}(1 + \epsilon) \left(\ln \left(\frac{1}{\delta} \right) + \frac{1 + \delta}{\delta} \right), \frac{1 - \delta}{2} \right),$$

và có độ phức tạp truy vấn

$$O \left(\frac{nk}{\epsilon^2} \log \left(\frac{n}{\epsilon} \right) \log \left(\frac{1}{\delta} \right) \right).$$

Chứng minh. Trước hết, ta chứng minh độ phức tạp truy vấn của thuật toán.

Phân tích độ phức tạp truy vấn. Trong giai đoạn thứ nhất, thuật toán cần $O(nk)$ truy vấn để tìm nghiệm $\mathbf{x}$ bằng cách gọi thuật toán UkSAlg trong [69]. Sau đó, thuật toán cần thêm nhiều nhất $O(nk)$ truy vấn để xác định chỉ số j . Vì vậy, tổng số truy vấn trong giai đoạn thứ nhất là $O(nk)$. Trong giai đoạn thứ hai, thuật toán thực hiện

$$\left\lceil \log_{1/(1-\epsilon')} \left(\frac{jn^2}{\epsilon'} \right) \right\rceil + 1 = O \left(\frac{1}{\epsilon} \log \left(\frac{n}{\epsilon} \right) \right)$$

vòng lặp. Ở mỗi vòng lặp, thuật toán gọi thủ tục FBiOpt, mà theo kết quả đã chứng minh trước đó có độ phức tạp truy vấn

$$O \left(\frac{nk}{\epsilon} \log \left(\frac{1}{\delta} \right) \right).$$

Do đó, tổng số truy vấn của giai đoạn thứ hai là

$$O \left(\frac{nk}{\epsilon^2} \log \left(\frac{n}{\epsilon} \right) \log \left(\frac{1}{\delta} \right) \right).$$

Kết hợp hai giai đoạn, ta thu được tổng độ phức tạp truy vấn của thuật toán:

$$O(nk) + O\left(\frac{nk}{\epsilon^2} \log\left(\frac{n}{\epsilon}\right) \log\left(\frac{1}{\delta}\right)\right) = O\left(\frac{nk}{\epsilon^2} \log\left(\frac{n}{\epsilon}\right) \log\left(\frac{1}{\delta}\right)\right).$$

Tiếp theo, ta chứng minh các chặn lý thuyết đối với nghiệm do thuật toán trả về. Giả sử T_0 được định nghĩa ở Dòng 10 của Thuật toán 3.5. Ta xét hai trường hợp sau.

Trường hợp 1. Giả sử $T_0 \geq T/2$.

Trong trường hợp này, theo cách xây dựng của thuật toán, ta có

$$g(\mathbf{s}) \geq g(\mathbf{x}_0) \geq T/2.$$

Mặt khác, chi phí của phần nghiệm ban đầu thỏa mãn

$$c(\mathbf{x}) \leq c(V_0) \leq \epsilon' \text{opt}.$$

Do đó, trong trường hợp này nghiệm thu được đã thỏa mãn ngay các đảm bảo cần thiết, và định lý được chứng minh.

Trường hợp 2. Giả sử $T_0 < T/2$.

Ký hiệu opt' là giá trị tối ưu của bài toán kSC trên thể hiện con (V', g', k, T') . Ta có

$$c_m = \min_{e \in V'} c(e) \leq \text{opt}' \leq c(V').$$

Ngoài ra, dãy giá trị

$$v_i = \frac{c_m}{(1 - \epsilon')^i}$$

được xây dựng sao cho tăng dần từ c_m đến $c(V')$ với công bội $1/(1 - \epsilon')$. Vì vậy, tồn tại một chỉ số i trong vòng lặp chính sao cho

$$\text{opt}' \leq v_i \leq \frac{\text{opt}'}{1 - \epsilon'}.$$

Xét hai tập con của nghiệm tối ưu $\mathbf{o}$:

$$\mathbf{o}' = \{(e, i) : e \in \text{supp}(\mathbf{o}) \setminus V_0, i = \mathbf{o}(e)\},$$

và

$$\mathbf{x}' = \{(e, i) : e \in \text{supp}(\mathbf{o}) \cap V_0, i = \mathbf{o}(e)\}.$$

Nói cách khác, $\mathbf{o}'$ là phần của nghiệm tối ưu nằm ngoài V_0 , còn $\mathbf{x}'$ là phần của nghiệm tối ưu nằm trong V_0 . Từ Bổ đề 3.5 và tính đơn điệu của hàm g , ta có

$$g(\mathbf{o}') + g(\mathbf{x}') \geq g(\mathbf{o}' \sqcup \mathbf{x}') \geq g(\mathbf{o}) \geq T.$$

Bất đẳng thức trên cho thấy tổng giá trị của hai phần nghiệm $\mathbf{o}'$ và $\mathbf{x}'$ ít nhất bằng ngưỡng yêu cầu T . Mặt khác, do $\mathbf{x}_0$ được tính bằng TTX cho bài toán tối đa hóa k -submodular không ràng buộc UkSM trong [69], ta có

$$T_0 = g(\mathbf{x}_0) \geq \frac{1}{2} \max_{\mathbf{u} \in (k+1)^{V_0}} g(\mathbf{u}).$$

Từ đó suy ra

$$g(\mathbf{o}') \geq T - g(\mathbf{x}') \quad (3.59)$$

$$\geq T - \max_{\mathbf{u} \in (k+1)^{V_0}} g(\mathbf{u}) \quad (3.60)$$

$$\geq T - 2T_0. \quad (3.61)$$

Bất đẳng thức này cho thấy $\mathbf{o}'$ là một nghiệm khả thi của thể hiện con (V', g', k, T') , trong đó $T' = T - 2T_0$. Hơn nữa, ta có

$$\text{opt}' \leq c(\mathbf{o}') \leq c(\mathbf{o}) = \text{opt}.$$

Nghĩa là chi phí tối ưu của bài toán con không vượt quá chi phí tối ưu của bài toán ban đầu. Theo Định lý 3.4, khi áp dụng thủ tục FBIOpt trên thể hiện con, ta thu được nghiệm $\mathbf{s}_i$ thỏa mãn

$$c(\mathbf{s}_i) \leq \frac{1}{2}(1 + 6\epsilon') \left(\log \left(\frac{1}{\delta} \right) + \frac{1 + \delta}{\delta} \right) \text{opt}'.$$

Do đó, khi ghép nghiệm $\mathbf{s}_i$ với $\mathbf{x}_0$, ta có

$$\begin{aligned} c(\mathbf{s}_i \sqcup \mathbf{x}_0) &\leq c(V_0) + c(\mathbf{s}_i) \\ &\leq \epsilon' \text{opt} + (1 + 6\epsilon') \left(\log \left(\frac{1}{\delta} \right) + \frac{1 + \delta}{\delta} \right) \frac{\text{opt}'}{2} \\ &< (1 + \epsilon) \left(\log \left(\frac{1}{\delta} \right) + \frac{1 + \delta}{\delta} \right) \frac{\text{opt}}{2}, \end{aligned}$$

vì $\epsilon = 8\epsilon'$ và $\text{opt}' \leq \text{opt}$. Tiếp theo, ta phân tích giá trị hàm mục tiêu của nghiệm ghép $\mathbf{s}_i \sqcup \mathbf{x}_0$. Do

$$g(\mathbf{s}_i \sqcup \mathbf{x}_0) - g(\mathbf{x}_0) = g'(\mathbf{s}_i) \geq \frac{(1 - \delta)T'}{2},$$

nên

$$\begin{aligned} g(\mathbf{s}_i \sqcup \mathbf{x}_0) &\geq \frac{(1 - \delta)T'}{2} + T_0 \\ &= \frac{(1 - \delta)(T - 2T_0)}{2} + T_0 \\ &= \frac{(1 - \delta)T}{2} + \delta T_0 \\ &\geq \frac{(1 - \delta)T}{2}. \end{aligned}$$

Như vậy, nghiệm ghép $\mathbf{s}_i \sqcup \mathbf{x}_0$ đạt được giá trị ít nhất bằng $(1 - \delta)T/2$. Cuối cùng, theo quy tắc lựa chọn ở Dòng 15, thuật toán luôn trả về một nghiệm $\mathbf{s}$ sao cho

$$g(\mathbf{s}) \geq \frac{(1 - \delta)T}{2},$$

đồng thời

$$c(\mathbf{s}) \leq c(\mathbf{s}_i \sqcup \mathbf{x}_0) \leq (1 + \epsilon) \left(\log \left(\frac{1}{\delta} \right) + \frac{1 + \delta}{\delta} \right) \frac{\text{opt}}{2}.$$

□

3.4.3. Thực nghiệm

Nhằm đánh giá các thuật toán được đề xuất trong luận án, bao gồm *Tham lam mở rộng* và *FBI*, tác giả tiến hành đối chiếu các phương pháp này với những thuật toán tiên tiến hiện có, được tổng hợp trong Bảng 3.2. Việc so sánh được thực hiện dựa trên ba tiêu chí chính: (i) chất lượng nghiệm thu được, (ii) số lần truy vấn tối hàm mục tiêu g , đại diện cho chi phí tính toán, và (iii) thời gian thực thi trong thực tế.

3.4.3.1. Thiết lập thực nghiệm

Luận án tiến hành đánh giá thực nghiệm hai thuật toán đề xuất, gồm *Tham lam mở rộng* (Extended Greedy) và *FBI*, thông qua việc so sánh với các phương pháp tiên tiến hiện có trên hai ứng dụng điển hình của bài toán kSC: (i) bài toán *ITk* và (ii) bài toán *Ngưỡng doanh thu - RT*. Thuật toán Tham lam mở rộng có bảo đảm tỷ lệ xấp xỉ trong cả hai trường hợp và các thuật toán đề xuất đều được đánh giá trên hai bài toán này. Riêng đối với RT, thuật toán *FBI* không có bảo đảm tỷ lệ xấp xỉ và chỉ được sử dụng như một thuật toán heuristic để phục vụ so sánh thực nghiệm.

Các thuật toán đối sánh được lựa chọn dựa trên những kết quả tốt nhất đã được công bố trong tài liệu (xem Bảng 3.2), bao gồm:

- **Thuật toán 2** trong [75]: một thuật toán dạng luồng cho bài toán kSC. Với hàm k -submodular đơn điệu, thuật toán đạt tỷ lệ xấp xỉ $(\frac{3-\epsilon}{2\epsilon(1-\epsilon)}, \frac{1-\epsilon}{2})$, trong khi đối với hàm không đơn điệu đạt $(\frac{4-\epsilon}{3\epsilon(1-\epsilon)}, \frac{1-\epsilon}{3})$. Độ phức tạp truy vấn tương ứng là $O\left(nk \frac{\log(\frac{c_{min}}{nc_{max}})}{\log(1-\epsilon)}\right)$, trong đó c_{min} và c_{max} lần lượt biểu thị chi phí nhỏ nhất và lớn nhất của các phần tử.
- **Thuật toán 3** trong [75]: một biến thể của Thuật toán 2 nhằm loại bỏ giả định về việc biết trước khoảng chi phí. Thuật toán này duy trì cùng tỷ lệ xấp xỉ như trên, nhưng xây dựng các ngưỡng dựa trên giá trị mục tiêu T . Độ phức tạp truy vấn được biểu diễn bởi $O\left(nk \frac{\log!(\epsilon T (B \max_{e \in V, i \in [k]} g((e, i)))^{-1})}{\log(1-\epsilon)}\right)$.

Trong toàn bộ các thí nghiệm, các tham số được cố định ở mức $\epsilon = 0.1$ và $\delta = 0.1$. Chi phí của phần tử được khởi tạo trong $(0, 1]$ [81, 93]. Giá trị T được lấy trong tập $\{0.1g(\mathbf{x}), 0.2g(\mathbf{x}), 0.3g(\mathbf{x}), 0.4g(\mathbf{x}), 0.5g(\mathbf{x})\}$, trong đó $\mathbf{x}$ là lời giải của UkSM trên tập V được xác định bằng thuật toán hệ số xấp xỉ $1/2$ trong [69]. Các tham số được trình bày trong Bảng 3.3. Trong phần thực nghiệm, LA sử dụng bốn bộ dữ liệu chuẩn về OSN tại

Bảng 3.3: Các tham số sử dụng trong thực nghiệm

Tham số	Giá trị
ϵ, δ	0.1 [95, 75]
Chi phí phần tử	Khởi tạo ngẫu nhiên trong khoảng $(0, 1]$ [81, 93]
Ngưỡng T	$\{0.1g(\mathbf{x}), 0.2g(\mathbf{x}), 0.3g(\mathbf{x}), 0.4g(\mathbf{x}), 0.5g(\mathbf{x})\}$ [81]

SNAP² được thống kê trong Bảng 3.4.

Mạng	Ứng dụng	Số đỉnh	Số cạnh	Nguồn
Barabási–Albert	ITk	1,000	$\sim 5,000$	Mô hình BA [106]
Facebook	ITk	4,039	88,234	SNAP [92]
Email-Eu-core	RT	1,005	25,571	SNAP [92]
GR-QC	RT	5,242	14,496	SNAP [92]

Bảng 3.4: Các bộ dữ liệu sử dụng trong thực nghiệm

3.4.3.2. Kết quả thực nghiệm

a. Chi phí lời giải

Từ các Hình 3.3 và 3.4 có thể quan sát rằng chi phí của lời giải thu được bởi các thuật toán có sự biến động theo các giá trị ngưỡng T . Khi T tăng, chi phí của tất cả các thuật toán đều tăng theo xu hướng tương tự. Tuy nhiên, nhìn chung các đường biểu diễn chi phí nằm khá gần nhau, cho thấy chất lượng lời giải của các phương pháp là tương đương.

Mặc dù tại một số giá trị của T có thể xuất hiện sự chênh lệch nhất định giữa các thuật toán, sự khác biệt này không quá đáng kể và xu hướng tổng thể của các đường biểu diễn vẫn tương tự nhau khi T tăng. Trong một số trường hợp, thuật toán của Wang có thể cho chi phí lớn hơn, trong khi Thuật toán Tham lam mở rộng đạt được chi phí nhỏ hơn. Nhìn chung, các phương pháp đều đạt được chất lượng lời giải tương đương về mặt chi

²<https://snap.stanford.edu>

phí. Đáng chú ý, Thuật toán Tham lam mở rộng và Thuật toán FBI được đề xuất trong luận án luôn cho chi phí nằm trong cùng một khoảng với các thuật toán đối sánh, cho thấy các phương pháp đề xuất vẫn đảm bảo chất lượng lời giải cạnh tranh trong các ứng dụng IT.

Tương tự, từ các Hình 3.5 và 3.6 có thể quan sát rằng chi phí của lời giải cũng tăng dần theo ngưỡng T . Tuy nhiên, các đường biểu diễn chi phí vẫn nằm khá gần nhau, cho thấy sự khác biệt giữa các thuật toán là không lớn.

Để đánh giá rõ hơn về hiệu quả của thuật toán đề xuất, Bảng 3.5 biểu diễn so sánh theo tỷ lệ giảm chi phí của thuật toán đề xuất. Trong đa số các trường hợp, Thuật toán

Bảng 3.5: Thống kê số so sánh theo trường hợp của thuật toán đề xuất với các thuật toán theo tổng số 15 trường hợp của T .

Thuật toán	Tham lam mở rộng	FBI	Tham lam mở rộng hoặc FBI
Số trường hợp tốt hơn cả Thuật toán 2 và Thuật toán 3 trong [75]	8/15	10/15	15/15
Tỷ lệ	53.3%	66.7%	100%

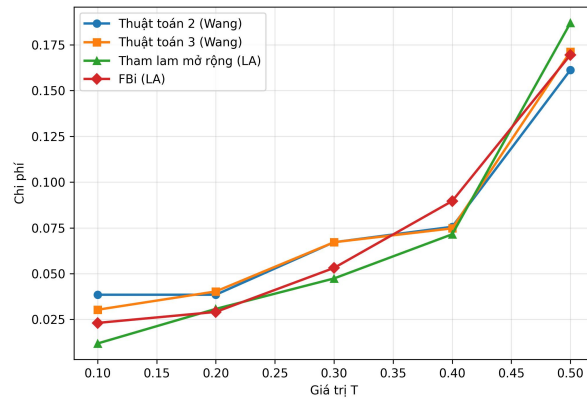
Tham lam mở rộng và FBI cho chi phí nhỏ hơn hoặc tương đương so với các thuật toán còn lại và ít nhất một trong hai thuật toán tốt hơn các thuật toán còn lại.

b. Số lượng truy vấn

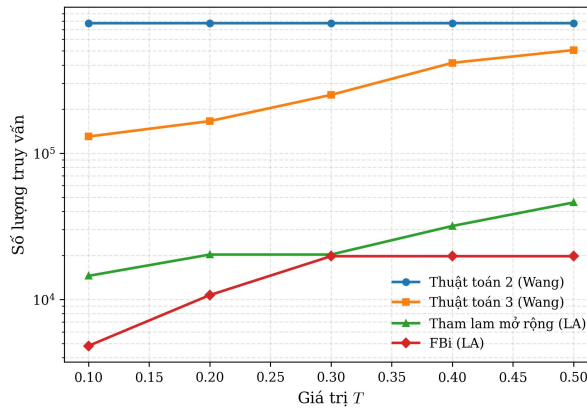
Từ các Hình 3.3(b) và 3.4(b), có thể nhận thấy sự chênh lệch đáng kể về số lượng truy vấn giữa các thuật toán trong ứng dụng IT. Cụ thể, Thuật toán 2 trong [75] sử dụng số truy vấn lớn nhất và hầu như không biến đổi theo các giá trị của ngưỡng T , duy trì ở mức vài triệu. Điều này cho thấy thuật toán phải đánh giá một lượng lớn ứng viên trong quá trình lựa chọn phần tử. Trong khi đó, Thuật toán 3 trong [75] giảm đáng kể số truy vấn so với Thuật toán 2, nhưng vẫn ở mức cao và có xu hướng tăng khi T tăng.

Trái lại, các thuật toán được đề xuất trong luận án thể hiện ưu thế rõ rệt về chi phí truy vấn. Thuật toán *Tham lam mở rộng* (LA) sử dụng ít truy vấn hơn đáng kể so với các phương pháp của Wang và chỉ tăng dần theo T . Đáng chú ý, thuật toán FBI (LA) thường đạt số lượng truy vấn thấp nhất trong phần lớn các trường hợp khảo sát. So với Thuật toán 2, các phương pháp đề xuất có thể giảm số truy vấn từ một đến hai bậc độ lớn, cho thấy hiệu quả vượt trội về mặt tính toán.

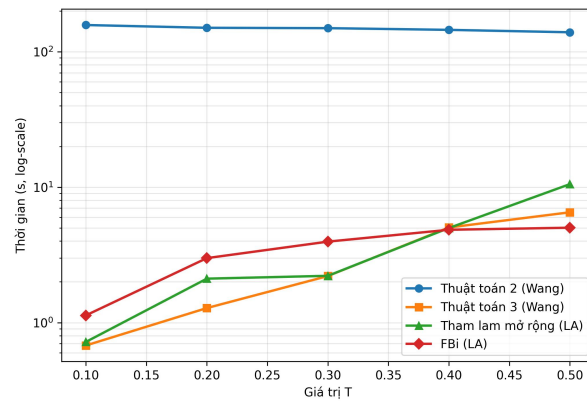
Xu hướng tương tự cũng được quan sát trong ứng dụng RT thông qua các Hình 3.5(b) và 3.6(b). Thuật toán 2 tiếp tục yêu cầu số truy vấn cao nhất và gần như không phụ thuộc vào T , trong khi Thuật toán 3 có cải thiện nhưng vẫn tăng đáng kể khi T lớn dần. Ngược lại, hai thuật toán *Tham lam mở rộng* và FBI duy trì mức truy vấn thấp hơn rõ rệt và chỉ



(a) Chi phí của lời giải

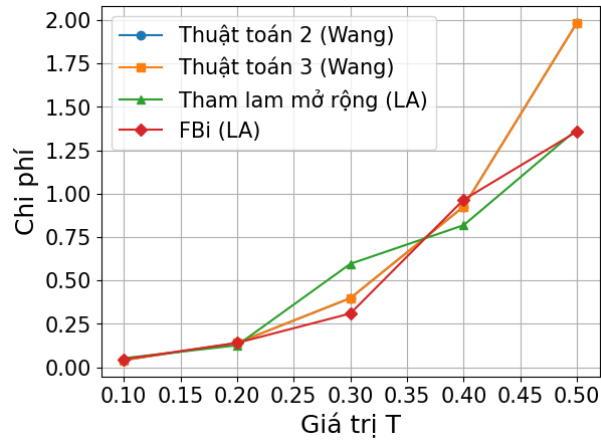


(b) Số lượng truy vấn (trục tung ở thang log)

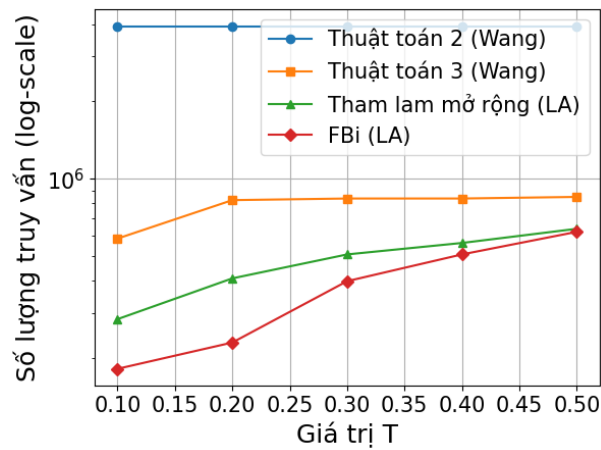


(c) Thời gian chạy (trục tung ở thang log)

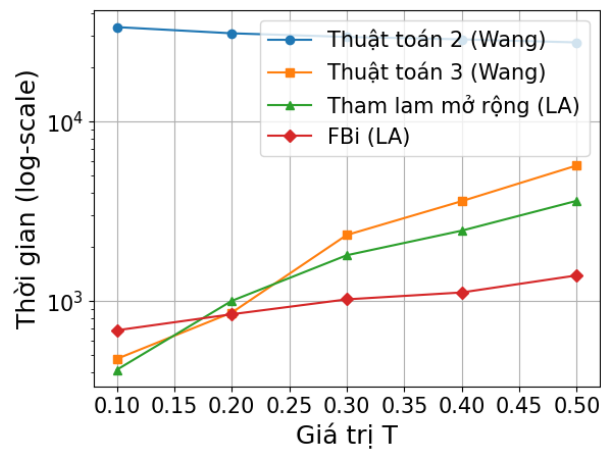
Hình 3.3: So sánh hiệu năng của các thuật toán trên mạng Barabási–Albert (BA) theo ngưỡng T với ứng dụng IT.



(a) Chi phí của lời giải

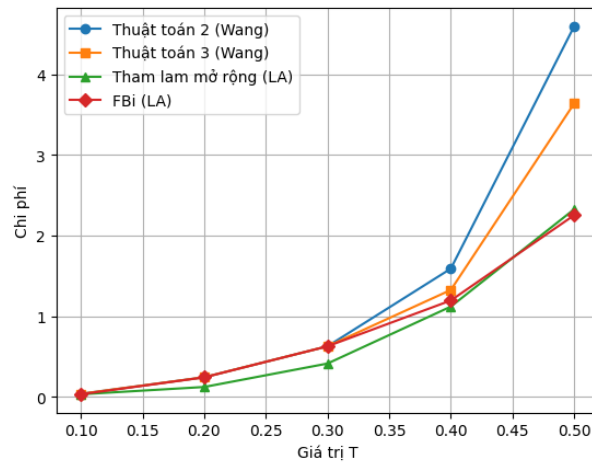


(b) Số lượng truy vấn (log-scale)

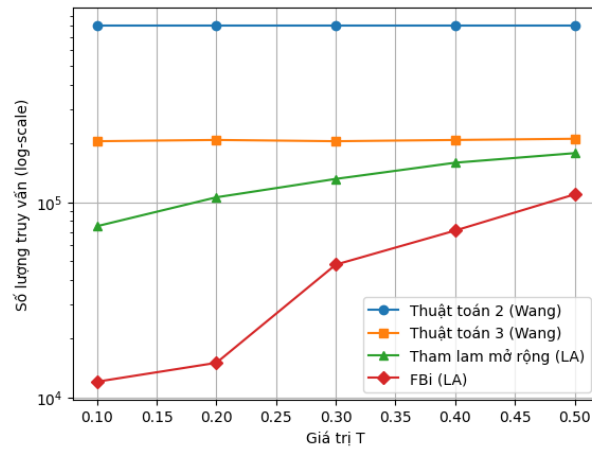


(c) Thời gian chạy (log-scale)

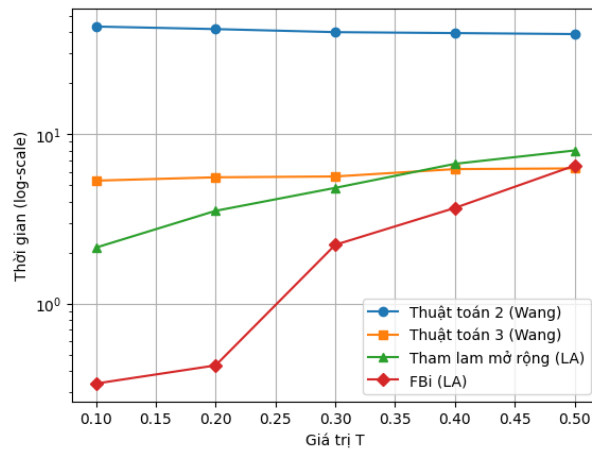
Hình 3.4: So sánh hiệu năng của các thuật toán theo ngưỡng T với ứng dụng IT: (a) chi phí của lời giải; (b) số lượng truy vấn; (c) thời gian chạy.



(a) Chi phí của lời giải

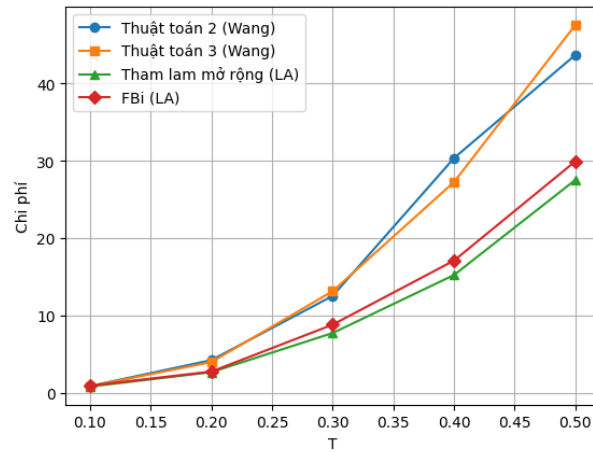


(b) Số lượng truy vấn (trục tung ở thang log)

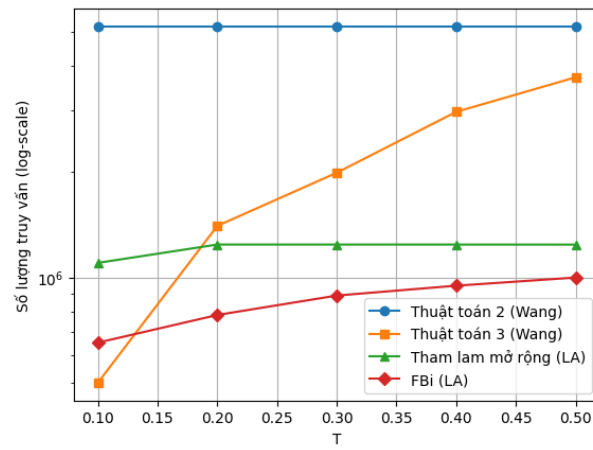


(c) Thời gian chạy (trục tung ở thang log)

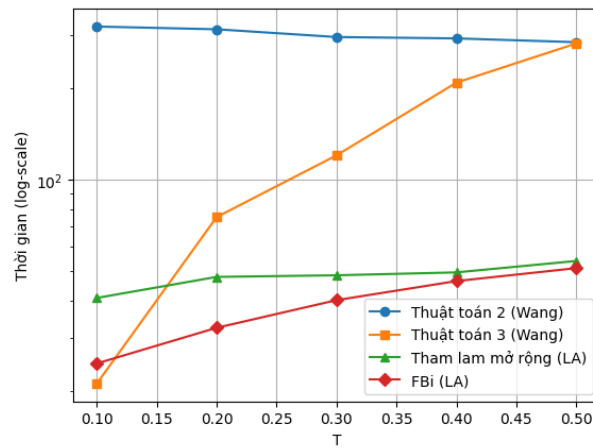
Hình 3.5: So sánh hiệu năng của các thuật toán trên bộ dữ liệu Email-Eu đối với ứng dụng RT theo ngưỡng T : (a) chi phí của lời giải; (b) số lượng truy vấn; (c) thời gian chạy.



(a) Chi phí của lời giải



(b) Số lượng truy vấn (trục tung ở thang log)



(c) Thời gian chạy (trục tung ở thang log)

Hình 3.6: So sánh hiệu năng của các thuật toán trên bộ dữ liệu Email-Eu đối với ứng dụng RT theo các giá trị ngưỡng T : (a) chi phí của lời giải; (b) số lượng truy vấn; (c) thời gian chạy.

tăng chậm theo T .

Tổng thể, số lượng truy vấn của các thuật toán đề xuất trong luận án được giữ ở mức thấp và ổn định hơn so với các phương pháp hiện có, ngay cả khi ngưỡng mục tiêu tăng lên.

Những quan sát thực nghiệm này cũng phù hợp với phân tích lý thuyết về độ phức tạp truy vấn được trình bày trong Bảng 3.1. Cụ thể, các thuật toán của Wang có độ phức tạp phụ thuộc vào kích thước tập nền và các tham số liên quan đến ngưỡng, dẫn đến chi phí truy vấn cao trong thực nghiệm. Ngược lại, Thuật toán *Tham lam mở rộng* và FBI được thiết kế nhằm hạn chế số lần đánh giá hàm thông qua các chiến lược lựa chọn phần tử hiệu quả, từ đó đạt được chi phí truy vấn thấp hơn đáng kể trên các bộ dữ liệu khảo sát.

c. Thời gian

Từ các Hình 3.3(c) và 3.4(c), có thể nhận thấy xu hướng thời gian chạy tương đồng với kết quả về số lượng truy vấn. Cụ thể, Thuật toán 2 trong [75] có thời gian thực thi lớn nhất trong tất cả các phương pháp so sánh và gần như không biến đổi theo ngưỡng T . Điều này phù hợp với việc thuật toán phải thực hiện một số lượng truy vấn rất lớn trong quá trình xử lý.

Thuật toán 3 trong [75] giúp rút ngắn đáng kể thời gian thực thi so với Thuật toán 2, tuy nhiên chi phí này vẫn có xu hướng gia tăng theo T . Trái lại, các thuật toán được đề xuất trong luận án cho thấy hiệu năng vượt trội hơn. Cụ thể, *Tham lam mở rộng* (LA) và FBI duy trì thời gian chạy thấp hơn rõ rệt so với Thuật toán 2, đồng thời trong nhiều trường hợp đạt hiệu quả tương đương, thậm chí tốt hơn Thuật toán 3.

Xu hướng này cũng được ghi nhận trong ứng dụng RT qua các Hình 3.5(c) và 3.6(c). Thuật toán 2 tiếp tục có thời gian chạy cao nhất và ít chịu ảnh hưởng của T , trong khi Thuật toán 3 tuy nhanh hơn nhưng vẫn tăng đáng kể theo ngưỡng. Trái lại, hai thuật toán *Tham lam mở rộng* và FBI duy trì thời gian chạy thấp và chỉ tăng nhẹ khi T tăng.

Một quan sát đáng chú ý là sự thay đổi của thời gian thực thi gần như tương ứng trực tiếp với xu hướng của số lượng truy vấn. Điều này cho thấy phần lớn chi phí tính toán của các thuật toán xuất phát từ quá trình đánh giá hàm mục tiêu. Do đó, trong thực nghiệm, thời gian chạy gần như tỉ lệ thuận với số lượng truy vấn, phù hợp với các phân tích lý thuyết về độ phức tạp truy vấn được trình bày trong Bảng 3.1. Các thuật toán của Wang, do cần thực hiện nhiều truy vấn hơn, dẫn đến thời gian thực thi cao hơn; ngược lại, các phương pháp đề xuất đạt hiệu quả tốt hơn nhờ giảm đáng kể số lần đánh giá hàm.

Trên mạng Barabási–Albert (BA), FBI chưa đạt thời gian chạy thấp nhất tại một số ngưỡng T nhỏ, mặc dù sử dụng ít truy vấn hơn các thuật toán đối sánh. Điều này cho

thấy việc giảm số truy vấn chưa bảo đảm cải thiện tương ứng về thời gian thực thi. Khi ngưỡng T cao, thuật toán Tham lam mở rộng cũng có thể chạy chậm hơn FBi và Thuật toán 3 của Wang trên BA. Về chi phí lời giải, FBi và Tham lam mở rộng lần lượt tốt hơn cả hai thuật toán đối sánh trong 10/15 và 8/15 trường hợp, cho thấy ưu thế về chất lượng chưa đồng đều giữa các thiết lập.

Kết quả tổng hợp trên cả hai ứng dụng IT và RT cho thấy các thuật toán đề xuất, bao gồm *Tham lam mở rộng* và FBi, đạt hiệu năng cạnh tranh so với các phương pháp của Wang. Xét về chất lượng nghiệm, các phương pháp này cho kết quả tương đương. Ngược lại, ở khía cạnh chi phí truy vấn, số lần đánh giá hàm được cắt giảm đáng kể, thường giảm từ một đến hai bậc độ lớn. Nhờ đó, thời gian thực thi cũng được rút ngắn rõ rệt trong phần lớn các kịch bản thực nghiệm. Nhìn chung, các thuật toán đề xuất vừa duy trì chất lượng lời giải, vừa cải thiện đáng kể hiệu quả tính toán.

3.5. KẾT LUẬN CHƯƠNG

Trong chương này, luận án tập trung nghiên cứu bài toán *Phủ k -Submodular* (kSC), một dạng mở rộng tự nhiên của bài toán phủ submodular cổ điển trong bối cảnh ra quyết định đa lựa chọn. Mô hình này cho phép biểu diễn nhiều ứng dụng quan trọng trong trí tuệ nhân tạo, nghiên cứu tác nghiệp và phân tích dữ liệu, như lan truyền ảnh hưởng theo nhiều chủ đề, tối ưu doanh thu với nhiều sản phẩm, hay lựa chọn đặc trưng đa lớp.

Xét trên phương diện lý thuyết, luận án đề xuất các TTX tiêu chuẩn kép hiệu quả cho cả hai kịch bản chi phí đồng nhất và chi phí tổng quát. Cụ thể, trong trường hợp chi phí đồng nhất, thuật toán FBi cung cấp bảo đảm xấp xỉ được cải thiện đáng kể so với các phương pháp trước đó. Đối với chi phí tổng quát, hai thuật toán *Tham lam mở rộng* và FBi được xây dựng, trong đó *Tham lam mở rộng* áp dụng cho cả hàm đơn điệu và không đơn điệu, còn FBi giúp cắt giảm đáng kể độ phức tạp truy vấn trong khi vẫn duy trì chất lượng nghiệm ở mức cạnh tranh. Về thực nghiệm, các kết quả trên nhiều bộ dữ liệu thực và tổng hợp cho thấy các thuật toán đề xuất luôn đạt chất lượng nghiệm tương đương với các phương pháp tiên tiến hiện có. Đồng thời, số lượng truy vấn và thời gian thực thi được cắt giảm đáng kể so với các thuật toán của Wang, cho thấy ưu thế rõ rệt về hiệu quả tính toán và khả năng mở rộng.

Bên cạnh những kết quả đạt được, nghiên cứu trong chương vẫn còn một số hạn chế. Các thuật toán đề xuất chủ yếu được xây dựng cho mô hình kSC tĩnh và chưa xem xét các bối cảnh dữ liệu động, luồng hoặc phân tán. Ngoài ra, các kết quả hiện tại mới tập trung vào mô hình phủ k -submodular cơ bản và chưa xét đến các ràng buộc thực tiễn như công bằng, matroid hay nhiều mức ngân sách.

CHƯƠNG 4

BÀI TOÁN TỐI ĐA HÀM SUBMODULAR VỚI RÀNG BUỘC CHI PHÍ CÔNG BẰNG

Ở Chương 2 và Chương 3, luận án tập trung nghiên cứu các bài toán tối thiểu hóa chi phí nhằm đạt một ngưỡng giá trị hàm mục tiêu cho trước. Cụ thể, Chương 2 đề xuất các thuật toán song song cho bài toán MCSC, còn Chương 3 phát triển các TTXS tiêu chuẩn kép cho bài toán k SC trong cả hai trường hợp chi phí đồng nhất và chi phí tổng quát. Tiếp nối các nội dung này, chương này tập trung vào Dạng 2 của SO là bài toán SM là một trong những bài toán trung tâm của trí tuệ nhân tạo và tối ưu tổ hợp, với nhiều ứng dụng quan trọng trong thực tiễn như tối đa ảnh hưởng trong OSN [1, 43, 107, 93], lựa chọn tập đại diện [108, 109], phân bổ tài nguyên và khai thác dữ liệu quy mô lớn [108, 4]. Do tính chất biểu diễn tốt hiệu ứng lợi ích giảm dần, các hàm submodular cho phép mô hình hóa tự nhiên nhiều bài toán tối ưu trong các hệ thống phức tạp.

Tuy nhiên, trong một số hướng nghiên cứu gần đây, yếu tố chi phí và sự phân bổ ngân sách giữa các nhóm người dùng trong lời giải thường bị bỏ qua hoặc chỉ được xét một cách gián tiếp. Điều này dẫn đến các lời giải tuy đạt giá trị hàm mục tiêu cao nhưng có thể thiếu công bằng hoặc không hợp lệ trong các kịch bản thực tế, nơi mỗi nhóm người dùng chịu các ràng buộc tài nguyên khác nhau. Một ví dụ tiêu biểu là bài toán tối đa hóa ảnh hưởng trong OSN [1]. Trong bối cảnh này, việc lựa chọn tập hạt giống thường được thực hiện trên nhiều nhóm người dùng khác nhau, chẳng hạn như phân theo khu vực địa lý, sở thích hoặc đặc điểm nhân khẩu học. Trên thực tế, mỗi nhóm thường có một mức ngân sách riêng, dẫn đến yêu cầu kiểm soát chi phí theo hướng cân bằng giữa các nhóm trong quá trình lựa chọn.

Từ những hạn chế đó, chương này tập trung nghiên cứu bài toán tối đa hóa ảnh hưởng dưới ràng buộc chi phí công bằng giữa các nhóm người dùng. Luận án đề xuất các mô hình tối ưu phù hợp, đồng thời phát triển các TTXS có bảo đảm lý thuyết nhằm dung hòa giữa giá trị ảnh hưởng đạt được và tính công bằng về chi phí trong nghiệm. Các kết quả liên quan đến bài toán SMFC đã được công bố tại hội nghị **Asian Conference on Intelligent Information and Database (ACIIDS)** (Rank B) năm 2025 và hội nghị **Symposium on Information and Communication Technology (SOICT)** năm 2024.

4.1. PHÁT BIỂU BÀI TOÁN

Định nghĩa 4.1. (Bài toán SMFC)

Input:

- Cho trước một tập nền V , một hàm submodular đơn điệu $f : 2^V \rightarrow \mathbb{R}_+$

- V được chia thành K nhóm $C_1, C_2, \dots, C_K$ với $C_i \subseteq V, \cup_i C_i = V, C_i \cap C_j = \emptyset$.
- Mỗi phần tử $u \in V$ có một chi phí dương $c(u)$.
- Chi phí tổng là $B > 0$.
- Tỷ lệ công bằng $\alpha \in (0, 1)$.

Output:

- Mục tiêu là tìm một tập $S \subseteq V$ sao cho giá trị $f(S)$ là lớn nhất, đồng thời thoả mãn các ràng buộc sau:
 1. Tổng chi phí của tập được chọn không vượt quá ngân sách tổng, tức là $c(S) \leq B$.
 2. Tổng chi phí các phần tử được chọn trong mỗi nhóm không vượt quá αB : $c(S \cap C_i) \leq \alpha B, \forall i \in [K]$.

Trong nhiều bối cảnh ứng dụng thực tế, đặc biệt là bài toán IM, các phần tử của tập nền thường được phân thành nhiều nhóm người dùng khác nhau, phản ánh các đặc trưng như vị trí địa lý, sở thích hoặc nhân khẩu học. Khi đó, việc lựa chọn tập hạt giống không chỉ bị ràng buộc bởi ngân sách tổng mà còn cần đáp ứng yêu cầu về tính công bằng giữa các nhóm.

- *Ràng buộc ngân sách tổng* xác định *nguồn lực tối đa* mà chiến lược tối ưu được phép sử dụng. Trong nhiều ứng dụng thực tế như quảng cáo, marketing, phân bổ tài nguyên hay lựa chọn tập hạt giống trong OSN, ngân sách luôn hữu hạn. Do đó, tập nghiệm phải được lựa chọn sao cho tổng chi phí không vượt quá mức ngân sách B đã được cấp.
- *Ràng buộc ngân sách theo nhóm nhằm đảm bảo* tính công bằng trong việc phân bổ chi phí giữa các nhóm. Khi tập dữ liệu được chia thành nhiều nhóm dựa trên các đặc trưng như khu vực địa lý, độ tuổi, giới tính, cộng đồng hoặc các thuộc tính nhân khẩu học khác, việc dồn quá nhiều ngân sách vào một nhóm có thể dẫn đến sự mất cân bằng và làm giảm cơ hội được đại diện của các nhóm còn lại. Vì vậy, mỗi nhóm chỉ được sử dụng tối đa một tỷ lệ α của ngân sách tổng. Cần lưu ý rằng ràng buộc này chỉ là *chặn trên* $c(S \cap C_i) \leq \alpha B$: nó giới hạn chi phí tối đa mà mỗi nhóm được phép sử dụng, nhưng *không* bảo đảm mỗi nhóm phải nhận được một mức ngân sách tối thiểu. Nói cách khác, một nhóm vẫn có thể không có phần tử nào được chọn trong nghiệm. Loại ràng buộc này xuất hiện phổ biến

trong các bài toán tối ưu công bằng (*fair optimization*), phúc lợi xã hội (*social welfare optimization*), phân bổ tài nguyên và tối đa ảnh hưởng công bằng trong OSN, nơi bên cạnh mục tiêu tối ưu hóa hiệu quả còn cần đảm bảo sự cân bằng và đại diện giữa các nhóm đối tượng khác nhau.

Ứng dụng trong tối đa ảnh hưởng Trong nhiều bài toán thực tiễn liên quan đến marketing lan truyền và phổ biến thông tin trên OSN, người dùng thường được phân thành các nhóm dựa trên những đặc trưng như vị trí địa lý, độ tuổi, sở thích hoặc các yếu tố nhân khẩu học. Trong các chiến dịch quảng bá hoặc truyền thông cộng đồng, bên cạnh mục tiêu tối đa hóa phạm vi lan truyền thông tin, việc phân bổ nguồn lực hợp lý và cân bằng giữa các nhóm người dùng cũng là một yếu tố quan trọng được quan tâm.

Do đó, bài toán đặt ra là lựa chọn một tập hạt giống có khả năng tối đa hóa mức độ lan tỏa thông tin trên OSN, đồng thời đảm bảo ngân sách được phân bổ một cách cân đối giữa các nhóm người dùng.

Xét một đồ thị có hướng $G = (V, E)$, trong đó V là tập người dùng trong OSN và E là tập các cạnh biểu diễn mối quan hệ ảnh hưởng giữa các người dùng. Khi một tập người dùng $S \subseteq V$ được chọn làm tập hạt giống ban đầu, quá trình lan truyền thông tin diễn ra theo một mô hình khuếch tán $\mathcal{M}$ cho trước, chẳng hạn như mô hình IC hoặc LT.

Ký hiệu $f(S)$ là số lượng kỳ vọng người dùng được kích hoạt sau khi quá trình lan truyền kết thúc. Hàm $f(\cdot)$ được biết là một *hàm đơn điệu và submodular* [1]. Trong thực tế, mỗi người dùng $u \in V$ có một chi phí lựa chọn $c(u) > 0$, phản ánh chi phí để chọn người dùng đó làm hạt giống. Tổng ngân sách dành cho chiến dịch lan truyền được ký hiệu là B .

Giả sử tập người dùng V được phân chia thành K nhóm $C_1, C_2, \dots, C_K$, trong đó mỗi $C_i \subseteq V$ biểu diễn một nhóm người dùng có chung đặc điểm (ví dụ cùng khu vực hoặc cùng phân khúc khách hàng). Để đảm bảo tính công bằng giữa các nhóm, ngân sách sử dụng cho mỗi nhóm không được vượt quá một tỷ lệ $\alpha \in (0, 1)$ của tổng ngân sách. Mục tiêu của bài toán là lựa chọn một tập hạt giống $S \subseteq V$ sao cho mức lan truyền ảnh hưởng trong mạng là lớn nhất, đồng thời thỏa mãn các ràng buộc ngân sách tổng và ràng buộc công bằng giữa các nhóm. Cụ thể, bài toán được phát biểu như sau

$$\begin{aligned} \max_{S \subseteq V} \quad & f(S) \\ \text{Thỏa mãn:} \quad & c(S) \leq B, \\ & c(S \cap C_i) \leq \alpha B, \quad \forall i \in [K]. \end{aligned}$$

Bài toán trên là một trường hợp của bài toán tối đa hóa *hàm submodular đơn điệu* dưới nhiều ràng buộc *knapsack*. Các ràng buộc theo nhóm có thể được hiểu như các ràng buộc

ngân sách bổ sung nhằm đảm bảo tính công bằng trong việc lựa chọn các hạt giống.

Vì các phần tử thuộc lời giải khả thi e luôn phải đảm bảo $c(e) \leq \alpha B$. Do vậy ta giả sử đã loại hết các phần tử có chi phí $c(e) \leq \alpha B$ ra khỏi tập V .

Để dễ hình dung hơn trong bối cảnh phân bổ chi phí công bằng, ta xét ví dụ sau

Ví dụ 4.1. Giả sử mạng xã hội được chia thành ba nhóm người dùng:

C_1 = nhóm người dùng thành thị

C_2 = nhóm người dùng ngoại thành

C_3 = nhóm người dùng nông thôn.

Tổng ngân sách của chiến dịch là $B = 100$ triệu đồng và chọn $\alpha = 0.4$. Khi đó, ngân sách dành cho các hạt giống thuộc mỗi nhóm không được vượt quá

$$\alpha B = 0.4 \times 100 = 40 \text{ triệu đồng.}$$

Giả sử thuật toán lựa chọn tập hạt giống S với phân bổ chi phí:

$$c(S \cap C_1) = 38, \quad c(S \cap C_2) = 34, \quad c(S \cap C_3) = 28.$$

Khi đó,

$$c(S) = 38 + 34 + 28 = 100 \leq B$$

và

$$c(S \cap C_i) \leq 40 = \alpha B, \quad \forall i \in [3].$$

Vì vậy, S thỏa mãn cả ràng buộc ngân sách tổng và ràng buộc công bằng. Ngược lại, xét một tập hạt giống S' có phân bổ chi phí

$$c(S' \cap C_1) = 70, \quad c(S' \cap C_2) = 20, \quad c(S' \cap C_3) = 10.$$

Mặc dù

$$c(S') = 70 + 20 + 10 = 100 \leq B,$$

tập S' không thỏa mãn ràng buộc công bằng vì

$$c(S' \cap C_1) = 70 > 40 = \alpha B.$$

Do đó, S' không phải là một nghiệm khả thi. Như vậy, ràng buộc theo nhóm ngăn thuật toán dồn quá αB ngân sách vào bất kỳ nhóm nào.

Tuy nhiên, ràng buộc này chỉ giới hạn chi phí tối đa của mỗi nhóm và không bảo đảm mỗi nhóm nhận được một mức ngân sách tối thiểu. Chẳng hạn, xét tập hạt giống S'' có phân bổ chi phí

$$c(S'' \cap C_1) = 40, \quad c(S'' \cap C_2) = 40, \quad c(S'' \cap C_3) = 0.$$

Khi đó $c(S'') = 80 \leq B$ và $c(S'' \cap C_i) \leq 40 = \alpha B$ với mọi $i \in [3]$, nên S'' vẫn là một nghiệm khả thi, mặc dù nhóm người dùng nông thôn C_3 không có hạt giống nào. Việc bổ sung chặn dưới về chi phí cho từng nhóm là một mở rộng của mô hình và chưa được xem xét trong luận án.

4.2. CÁC NGHIÊN CỨU LIÊN QUAN

Đối với hàm mục tiêu đơn điệu, George L. Nemhauser và các cộng sự [14] là những người đầu tiên đề xuất TTXS tối ưu cho bài toán tối SM dưới ràng buộc kích thước (SMC). Thuật toán tham lam cổ điển đạt tỷ lệ xấp xỉ $1 - 1/e$ thông qua việc lựa chọn tuần tự phần tử có độ tăng biên lớn nhất, với độ phức tạp truy vấn $O(nk)$, trong đó n là kích thước tập nền và k là giới hạn lực lượng. Kể từ công trình nền tảng này, nhiều hướng tiếp cận đã được phát triển nhằm cải thiện hiệu quả tính toán.

Một nhánh nghiên cứu quan trọng tập trung vào việc giảm độ phức tạp. Công trình [24] đã đề xuất TTXS hằng số với thời gian $O(n \log n)$ trong mô hình streaming, dựa trên chiến lược tham lam theo ngưỡng (threshold greedy), vốn sau đó trở thành một kỹ thuật phổ biến. Hướng tiếp cận này đã thúc đẩy các nghiên cứu nhằm đưa độ phức tạp truy vấn về mức gần tuyến tính. Gần đây, các công trình [55] và [18] đã đạt được các TTXS gần tối ưu $1 - 1/e$ với thời gian tuyến tính. Tuy nhiên, các phương pháp này chỉ áp dụng cho trường hợp đơn điệu và không mở rộng được sang hàm không đơn điệu.

Bài toán SM cũng được mở rộng sang các ràng buộc phức tạp hơn, đặc biệt là ràng buộc chi phí (SMK), công trình nổi bật [57, 58] đưa ra thuật toán tham lam rời rạc đạt tỷ lệ xấp xỉ $1 - 1/e$ tối ưu nhưng có độ phức tạp truy vấn rất cao, lên tới $O(n^5)$. Những nghiên cứu gần đây tiếp tục cải thiện hướng này. Công trình [59] giảm số phép đoán xuống hai và hạ độ phức tạp xuống $O(n^3 \log n)$ trong khi vẫn giữ tỷ lệ $1 - 1/e$. Tuy nhiên, các thuật toán nhanh hơn như trong [24] sau đó được phát hiện có sai sót [60]. Công trình [60] khắc phục bằng cách sử dụng kỹ thuật đa tuyến, đạt xấp xỉ gần $1 - 1/e$ nhưng với chi phí tính toán rất cao $O((1/\epsilon)^{O(1/\epsilon^4)} n \log^2 n)$.

Trong bối cảnh dữ liệu lớn, các thuật toán tuyến tính trở nên đặc biệt quan trọng. Công trình [18] đề xuất thuật toán thời gian $O(n)$ với tỷ lệ xấp xỉ $1/2$, đồng thời chỉ ra các giới hạn lý thuyết đối với độ phức tạp truy vấn. Bên cạnh đó, mô hình streaming cũng được nghiên cứu rộng rãi nhờ khả năng xử lý dữ liệu lớn với bộ nhớ hạn chế. Các

thuật toán trong [61, 62] đạt tỷ lệ xấp xỉ $1/(2.5 + \epsilon)$ với một hoặc hai lượt quét dữ liệu, trong khi [63] cải thiện tỷ lệ lên $1/(2 + \epsilon)$ nhưng với chi phí cao hơn. Đáng chú ý, [18] đạt được thuật toán streaming với số truy vấn gần tuyến tính $O(n \log(1/\epsilon)/\epsilon)$.

Ngoài ra, bài toán còn được mở rộng sang trường hợp có d ràng buộc chi phí, mỗi ràng buộc tương ứng với một hàm chi phí khác nhau. Kulik và cộng sự [110] đã chứng minh rằng, khi d là hằng số, có thể đạt được tỷ lệ xấp xỉ $1 - 1/e - \epsilon$. Đối với thuật toán luôn, các thuật toán được đề xuất trong [40] chỉ đạt tỷ lệ xấp xỉ $\frac{1}{2+2d}$, phụ thuộc tuyến tính vào số ràng buộc d và do đó kém hiệu quả khi d lớn. Đặc biệt, bài toán tối đa hóa hàm submodular dưới các ràng buộc ngân sách riêng cho từng nhóm vẫn là một hướng nghiên cứu mở, chưa có thuật toán xấp xỉ hiệu quả.

Các nghiên cứu về tối ưu submodular có ràng buộc công bằng. Các nghiên cứu hiện có chủ yếu mô hình hóa tính công bằng thông qua các ràng buộc về lực lượng (xem [36]). Cụ thể, tập nền V được phân chia thành K nhóm $C_1, C_2, \dots, C_K$, và số phần tử được lựa chọn từ mỗi nhóm phải thỏa mãn đồng thời chặn dưới và chặn trên:

$$\ell_i \leq |S \cap C_i| \leq u_i, \quad \forall i \in [K].$$

Ràng buộc này bảo đảm số phần tử được lựa chọn từ nhóm C_i nằm trong khoảng cho phép $[\ell_i, u_i]$, qua đó duy trì sự cân bằng về số lượng giữa các nhóm.

Trong khi đó, luận án tiếp cận tính công bằng theo chi phí thay vì theo lực lượng. Cụ thể, chi phí của các phần tử được lựa chọn từ mỗi nhóm không được vượt quá một tỷ lệ α của tổng ngân sách:

$$c(S \cap C_i) \leq \alpha B, \quad \forall i \in [K].$$

Ràng buộc này ngăn việc tập trung phần lớn ngân sách vào một nhóm duy nhất, qua đó hướng tới sự phân bổ chi phí cân bằng hơn giữa các nhóm, ngay cả khi các phần tử có chi phí lựa chọn khác nhau.

Mô hình được xem xét trong luận án là một bước nghiên cứu ban đầu về công bằng theo chi phí và hiện mới tập trung vào ràng buộc chặn trên $c(S \cap C_i) \leq \alpha B$. Do đó, mô hình chỉ giới hạn chi phí tối đa của mỗi nhóm và không bảo đảm mỗi nhóm được phân bổ một mức ngân sách tối thiểu. Các mở rộng như chặn dưới về chi phí hoặc các ngưỡng chi phí khác nhau cho từng nhóm chưa được xem xét và có thể là những hướng nghiên cứu tiếp theo.

Đóng góp của Luận án. LA nghiên cứu bài toán SMFC trong hai trường hợp: (i) *chi phí đồng nhất*, trong đó chi phí của mỗi phần tử đều bằng 1, và (ii) *chi phí tổng quát*, trong đó mỗi phần tử có thể có một chi phí dương bất kỳ.

Bảng 4.1: Tóm tắt các kết quả chính của luận án cho bài toán SMFC. $n = |V|$ là kích thước tập nền, k là ngân sách trong trường hợp chi phí đồng nhất, $\alpha \in (0, 1)$ là tỷ lệ công bằng giữa các nhóm, và $\epsilon > 0$ là tham số đánh đổi giữa chất lượng nghiệm và thời gian chạy của thuật toán.

Thuật toán	Trường hợp	Hệ số xấp xỉ	Độ phức tạp
TGFastU	Chi phí đồng nhất ($c(e) = 1$)	$1/2 - \epsilon$	$O\left(\frac{n}{\epsilon} \log \frac{k}{\epsilon}\right)$
TGFast	Chi phí tổng quát ($c(e) > 0$)	$\frac{\alpha}{\alpha+2} - \epsilon$	$O\left(\frac{n}{\epsilon} \log\left(n \frac{\alpha+2}{2\alpha}\right)\right)$

4.3. THUẬT TOÁN ĐỀ XUẤT

Trước hết, luận án xem xét trường hợp đặc biệt khi các phần tử có chi phí đồng nhất. Trường hợp này giúp làm rõ các ý tưởng cốt lõi của thuật toán và cho phép phân tích đơn giản hơn về mặt lý thuyết. Sau đó, các kết quả và kỹ thuật được mở rộng để xử lý trường hợp chi phí tổng quát, trong đó chi phí của các phần tử có thể khác nhau. Sự mở rộng này cho phép mô hình hóa chính xác hơn nhiều bài toán thực tế, nơi chi phí lựa chọn các phần tử không đồng nhất.

4.3.1. Thuật toán cho trường hợp đặc biệt khi chi phí đồng nhất

Trước hết, luận án xem xét một trường hợp đặc biệt nhưng quan trọng của bài toán SMFC, trong đó mỗi phần tử $e \in V$ có chi phí đồng nhất, tức là $c(e) = 1$, và tổng ngân sách B là một số nguyên. Trong trường hợp này, ràng buộc chi phí có thể được diễn giải tương đương như một ràng buộc lực lượng, từ đó cho phép thiết kế các thuật toán hiệu quả hơn.

Đối với thiết lập này, luận án đề xuất thuật toán TGFastU (*Tham lam ngưỡng nhanh cho chi phí đồng nhất- Threshold Greedy for Uni-cost*). Thuật toán TGFastU kế thừa và điều chỉnh ý tưởng của phương pháp tham lam ngưỡng [24] nhằm lựa chọn các phần tử có đóng góp lớn, đồng thời bảo đảm không vi phạm các ràng buộc phân bổ theo nhóm. Cụ thể, thuật toán khởi tạo tập nghiệm $S = \emptyset$ và trước hết xác định giá trị ước lượng ảnh hưởng lớn nhất của một phần tử đơn lẻ, ký hiệu là $M = \max_{e \in V} f(\{e\})$.

Phần chính của thuật toán được thực hiện trong một vòng lặp ngoài, với số vòng lặp bị chặn trên bởi $\log_{1/(1-\epsilon)}(k/\epsilon) + 1$. Ở vòng lặp thứ i , thuật toán duyệt qua tất cả các phần tử $e \in V \setminus S$ và thêm e vào tập nghiệm nếu việc thêm này không vi phạm ràng buộc phân bổ theo nhóm và ràng buộc ngân sách tổng, đồng thời thỏa mãn điều kiện đóng góp biên của e không nhỏ hơn ngưỡng hiện tại θ , trong đó $\theta = (1 - \epsilon)^i M$. Sau mỗi vòng lặp

Thuật toán 4.1: Thuật toán TGFastU**Input:** Tập nền V , hàm submodular $f : 2^V \rightarrow \mathbb{R}_+$, tập các nhóm $\mathcal{C} = \{C_1, C_2, \dots, C_K\}$, tỷ lệ công bằng $\alpha \in (0, 1)$, ngân sách $B = k$, tham số chính xác $\epsilon > 0$ **Output:** Tập nghiệm S

```

1:  $S \leftarrow \emptyset, M \leftarrow \max_{e \in V} f(\{e\}), \theta \leftarrow M;$ 
2: while  $\theta \geq \epsilon M/k$  do
3:   foreach  $e \in V$  do
4:     if  $|S \cap C(e)| < \alpha k$  and  $|S| < k$  then
5:       if  $f(S \cup \{e\}) - f(S) \geq \theta$  then
6:          $S \leftarrow S \cup \{e\};$ 
7:       end
8:     end
9:   end
10:   $\theta \leftarrow (1 - \epsilon)\theta;$ 
11: end
12: return  $S;$ 

```

ngoài, giá trị ngưỡng θ được giảm theo hệ số $(1 - \epsilon)$. Thuật toán kết thúc khi $\theta < \epsilon M/k$ và trả về tập nghiệm S . Giả mã chi tiết của thuật toán TGFastU được trình bày trong Thuật toán 4.1.

Định lý 4.1. Thuật toán 4.1 có độ phức tạp truy vấn là $O\left(\frac{n}{\epsilon} \log \frac{k}{\epsilon}\right)$ và trả về một nghiệm có tỷ lệ xấp xỉ $1/2 - \epsilon$ cho bài toán SMFC với trường hợp chi phí đồng nhất.

Chứng minh. Trước hết, xét số vòng lặp ngoài của Thuật toán 4.1. Do ngưỡng θ được giảm theo hệ số $(1 - \epsilon)$ và thuật toán dừng khi $\theta < \epsilon M/k$, số vòng lặp ngoài được chặn trên bởi

$$\begin{aligned}
\log_{\frac{1}{1-\epsilon}} \left(\frac{k}{\epsilon} \right) + 1 &= \frac{\log \left(\frac{k}{\epsilon} \right)}{\log \left(\frac{1}{1-\epsilon} \right)} + 1 \\
&\leq \frac{\log \left(\frac{k}{\epsilon} \right)}{\log(1 + \epsilon)} + 1 \quad \left(\text{vì } \frac{1}{1-\epsilon} \geq 1 + \epsilon \right) \\
&\leq \frac{2}{\epsilon} \log \left(\frac{k}{\epsilon} \right) + 1 \quad \left(\text{do } \log(1 + \epsilon) \geq \frac{\epsilon}{2} \right) \\
&= O\left(\frac{1}{\epsilon} \log \frac{k}{\epsilon} \right).
\end{aligned}$$

suy ra số vòng lặp ngoài là $O\left(\frac{1}{\epsilon} \log \frac{k}{\epsilon}\right)$. Trong mỗi vòng lặp, thuật toán duyệt qua toàn

bộ tập $\mathcal{R}$ để đánh giá đóng góp biên, do đó mỗi vòng lặp tốn $O(n)$ thời gian. Vì vậy, tổng độ phức tạp thời gian của thuật toán là $O\left(\frac{n}{\epsilon} \log \frac{k}{\epsilon}\right)$.

Tiếp theo, ta chứng minh tỷ lệ xấp xỉ. Giả sử $S = \{s_1, s_2, \dots, s_k\}$ là tập nghiệm thu được sau khi kết thúc vòng lặp chính (nếu $|S| < k$, ta bổ sung các phần tử rỗng để đủ kích thước). Ký hiệu s_i là phần tử được thêm vào ở bước thứ i , $S_i = \{s_1, \dots, s_i\}$, và $\theta(s_i)$ là giá trị ngưỡng tại thời điểm s_i được thêm vào S .

Gọi $O = \{o_1, o_2, \dots, o_k\}$ là tập nghiệm tối ưu. Với chi phí đồng nhất thì ràng buộc trở thành ràng buộc matroid [111]. Nhờ tính chất mở rộng của matroid, tồn tại một ánh xạ sao cho $\{s_1, \dots, s_{i-1}, o_i\}$ luôn tồn tại với mọi i . Ta chứng minh rằng

$$f(s_i | S_{i-1}) \geq (1 - \epsilon)f(o_i | S_{i-1}), \quad \forall i.$$

Bất đẳng thức này hiển nhiên đúng nếu s_i được chọn ở vòng lặp đầu tiên. Trong các trường hợp còn lại, nếu o_i chưa được chọn ở các vòng lặp trước, thì do quy tắc lựa chọn theo ngưỡng của thuật toán, ta có $f(o_i | S_{i-1}) < \theta(s_i)/(1 - \epsilon)$, từ đó suy ra bất đẳng thức trên. Nhờ tính submodular của f , ta có

$$\begin{aligned} f(S) &= \sum_{i=1}^k f(s_i | S_{i-1}) \\ &\geq \sum_{i=1}^k (1 - \epsilon)f(o_i | S_{i-1}) \\ &\geq (1 - \epsilon)(f(O) - f(S)) \end{aligned}$$

Suy ra

$$\begin{aligned} f(S) &\geq \frac{1 - \epsilon}{2 - \epsilon} f(O) \\ &= \left(\frac{1}{2} - \frac{\epsilon}{2(2 - \epsilon)} \right) f(O) \\ &\geq \left(\frac{1}{2} - \epsilon \right) f(O), \quad \left(\text{do } \frac{\epsilon}{2(2 - \epsilon)} \leq \epsilon, \epsilon \in (0, \frac{1}{2}) \right). \end{aligned}$$

Điều này chứng minh tỷ lệ xấp xỉ $1/2 - \epsilon$ của thuật toán và hoàn tất chứng minh. $\square$

4.3.2. Thuật toán cho trường hợp tổng quát

Trong mục này, LA đề xuất thuật toán TGFast (Tham lam ngưỡng nhanh- Threshold Greedy Fast), cho bài toán SMFC. Thuật toán đạt được tỷ lệ xấp xỉ $\frac{\alpha}{\alpha+2} - \epsilon$, trong đó $\epsilon \in (0, 1)$ là một hằng số nhỏ tùy ý. Ý tưởng của thuật toán TGFast mở rộng thuật toán cho trường hợp chi phí đồng nhất TGFastU với tiêu chí lựa chọn phần tử dựa trên *tỷ lệ mật độ* (density ratio), tức là tỷ lệ giữa độ tăng biên của hàm mục tiêu và chi phí của

Thuật toán 4.2: Thuật toán TGFast

Input: Tập nền V , hàm submodular $f : 2^V \rightarrow \mathbb{R}_+$, tập các nhóm

$\mathcal{C} = \{C_1, \dots, C_K\}$, tỷ lệ công bằng $\alpha \in (0, 1)$, ngân sách tổng B , tham số chính xác $\epsilon > 0$

Output: Tập nghiệm $S \subseteq V$

```

1:  $S \leftarrow \emptyset, e_{\max} \leftarrow \arg \max_{e \in V} f(e), M \leftarrow f(e_{\max}), \beta \leftarrow \frac{2\alpha}{\alpha+2}, \theta \leftarrow \frac{nM}{\alpha B};$ 
2: while  $\theta \geq \frac{\beta M}{\alpha B}$  do
3:   foreach  $e \in V \setminus S$  do
4:     if  $c(S) + c(e) \leq B$  then
5:       if  $c(S \cap C(e)) + c(e) \leq \alpha B$  then
6:         if  $\frac{f(S \cup \{e\}) - f(S)}{c(e)} \geq \theta$  then
7:            $S \leftarrow S \cup \{e\};$ 
8:         end
9:       end
10:    end
11:  end
12:   $\theta \leftarrow (1 - \epsilon)\theta;$ 
13: end
14:  $S' \leftarrow \arg \max_{X \in \{S, \{e_{\max}\}\}} f(X);$ 
15: return  $S'$ 

```

phần tử. Cách tiếp cận này cho phép lựa chọn các phần tử có đóng góp lớn trong khi vẫn đảm bảo không vi phạm các ràng buộc về chi phí tổng và chi phí theo nhóm trong mỗi vòng lặp.

Thuật toán khởi tạo tập nghiệm $S = \emptyset$ và trước hết tìm một phần tử $e_{\max} = \arg \max_{e \in V} f(e)$ có giá trị đơn lẻ lớn nhất, đồng thời đặt $M = f(e_{\max})$. Phần chính của thuật toán được thực hiện trong một vòng lặp ngoài, trong đó một ngưỡng θ được sử dụng để kiểm soát việc chọn phần tử. Ban đầu, θ được đặt tỉ lệ với M và sau mỗi vòng lặp ngoài, giá trị này được giảm theo hệ số $(1 - \epsilon)$.

Tại mỗi vòng lặp, thuật toán duyệt qua các phần tử trong $V \setminus S$. Một phần tử e được thêm vào tập S nếu việc thêm e không vi phạm ràng buộc ngân sách tổng, không vi phạm ràng buộc chi phí theo nhóm, và đồng thời thỏa mãn điều kiện tỷ lệ mật độ $f(e | S)/c(e) \geq \theta$. Thuật toán dừng khi ngưỡng θ nhỏ hơn một giá trị biên được xác định trước. Cuối cùng, thuật toán trả về nghiệm tốt hơn giữa tập S thu được và nghiệm đơn phần tử $\{e_{\max}\}$.

Định lý 4.2. *Thuật toán 4.2 có độ phức tạp truy vấn $O(\frac{n}{\epsilon} \log(n \frac{\alpha+2}{2\alpha}))$ và trả về một nghiệm hợp lệ có tỷ lệ xấp xỉ $\frac{\alpha}{\alpha+2} - \epsilon$ cho bài toán SMFC, trong đó $\alpha \in (0, 1)$ là hệ số công bằng của bài toán và ϵ là tham số chính xác của thuật toán thỏa mãn $\frac{\alpha}{\alpha+2} \geq \epsilon$.*

Chứng minh. Thuật toán trước hết tìm phần tử $e_{\max}$ bằng cách duyệt qua toàn bộ tập V , do đó cần $O(n)$ truy vấn hàm.

Ban đầu, ngưỡng θ được đặt bằng $\frac{nM}{\alpha B}$ và sau mỗi vòng lặp ngoài, giá trị này được giảm theo hệ số $(1 - \epsilon)$. Thuật toán dừng khi $\theta < \frac{\beta M}{\alpha B}$, trong đó $\beta = \frac{2\alpha}{\alpha+2}$. Do đó, số vòng lặp ngoài được chặn trên bởi

$$\begin{aligned} \log_{\frac{1}{1-\epsilon}} \left(\frac{n}{\beta} \right) + 1 &= \frac{\log \left(\frac{n}{\beta} \right)}{\log \left(\frac{1}{1-\epsilon} \right)} + 1 \\ &\leq \frac{\log \left(\frac{n}{\beta} \right)}{\log(1+\epsilon)} + 1 \quad \left(\text{vì } \frac{1}{1-\epsilon} \geq 1+\epsilon \right) \\ &\leq \frac{2}{\epsilon} \log \left(\frac{n}{\beta} \right) + 1 \quad \left(\text{do } \log(1+\epsilon) \geq \frac{\epsilon}{2}, \epsilon \in (0, 1) \right). \end{aligned}$$

Trong mỗi vòng lặp ngoài, thuật toán duyệt qua tối đa n phần tử. Với chú ý rằng β là

hằng số, tổng số truy vấn hàm tối đa là

$$\begin{aligned} n \left(\log_{\frac{1}{1-\epsilon}} \left(\frac{n}{\beta} \right) + 1 \right) &\leq n \left(\frac{2}{\epsilon} \log \left(\frac{n}{\beta} \right) + 1 \right) \\ &= O \left(\frac{n}{\epsilon} \log \left(\frac{n}{\beta} \right) \right) \\ &= O \left(\frac{n}{\epsilon} \log \left(n \frac{\alpha+2}{2\alpha} \right) \right). \end{aligned}$$

Ban đầu, $S = \emptyset$ nên hiển nhiên khả thi. Mỗi phần tử e chỉ được thêm vào S khi đồng thời thỏa mãn

$$c(S) + c(e) \leq B \quad \text{và} \quad c(S \cap C(e)) + c(e) \leq \alpha B.$$

Do đó, sau mỗi lần cập nhật, ràng buộc ngân sách tổng và ràng buộc chi phí của nhóm chứa e vẫn được bảo toàn; chi phí của các nhóm còn lại không thay đổi. Suy ra, theo quy nạp, tập S luôn khả thi. Ngoài ra, sau bước tiền xử lý, mọi phần tử $e \in V$ đều thỏa mãn $c(e) \leq \alpha B$, nên $\{e_{\max}\}$ cũng khả thi. Vì thuật toán trả về một trong hai nghiệm S và $\{e_{\max}\}$ nên nghiệm cuối cùng S' luôn khả thi.

Tiếp theo, ta phân tích tỷ lệ xấp xỉ. Gọi opt là giá trị tối ưu của bài toán. Do M là giá trị đơn lẻ lớn nhất, ta có $M \leq \text{opt} \leq Mn$. Vì vậy, tồn tại một số nguyên i sao cho

$$v = (1 - \epsilon)^i M \leq \text{opt} \leq (1 - \epsilon)^{i-1} M \leq \frac{v}{1 - \epsilon}.$$

Xét vòng lặp mà tại đó

$$\theta = \frac{\beta v}{\alpha B}.$$

Trường hợp 1. Không tồn tại phần tử $e \in O \setminus S$ sao cho $f(e \mid S) \geq \theta c(e)$ và việc thêm e không vi phạm ràng buộc theo nhóm.

Trường hợp 1.1. Không tồn tại phần tử e sao cho đồng thời

$$c(S \cup \{e\}) \leq B \quad \text{và} \quad f(e \mid S) \geq \theta c(e).$$

Khi đó, do tính submodular của f , ta có

$$f(O) - f(S) \leq \sum_{e \in O \setminus S} f(e \mid S) \tag{4.1}$$

$$< \sum_{e \in O \setminus S} \theta c(e) \tag{4.2}$$

$$\leq B\theta = \frac{\beta v}{\alpha}. \tag{4.3}$$

Suy ra

$$f(S) > \left(1 - \frac{\beta}{\alpha}\right) v,$$

và do đó $f(S') \geq f(S)$.

Trường hợp 1.2. Tồn tại một phần tử e thoả mãn

$$c(S \cup \{e\}) > B \quad \text{và} \quad f(e \mid S) \geq \theta c(e).$$

Khi đó,

$$\begin{aligned} f(S') &\geq \frac{f(S) + f(e)}{2} \\ &\geq \frac{f(S \cup \{e\})}{2} \\ &\geq \frac{c(S \cup \{e\}) \theta}{2} \\ &> \frac{B\theta}{2} = \frac{\beta v}{2\alpha}. \end{aligned}$$

Trường hợp 2. Tồn tại phần tử $e \in O \setminus S$ sao cho $f(e \mid S) \geq \theta c(e)$ và việc thêm e vi phạm ràng buộc theo nhóm. Vì lúc này chi phí tổng lớn hơn αB khi thêm phần tử e , nên ta có

$$\begin{aligned} f(S \cup \{e\}) &\geq f(S) + f(e \mid S) \\ &\geq \frac{(c(S) + c(e))\beta v}{\alpha B} \\ &\geq \beta v. \end{aligned}$$

Suy ra

$$f(S') \geq \frac{f(S \cup \{e\})}{2} \geq \frac{\beta v}{2}.$$

Kết hợp hai trường hợp trên, ta thu được

$$\begin{aligned} f(S') &\geq \min \left\{ 1 - \frac{\beta}{\alpha}, \frac{\beta}{2} \right\} v \\ &= \frac{\alpha}{\alpha + 2} v \\ &\geq \frac{\alpha}{\alpha + 2} (1 - \epsilon) \text{opt} \\ &\geq \left(\frac{\alpha}{\alpha + 2} - \epsilon \right) \text{opt}. \end{aligned}$$

Điều này hoàn tất chứng minh định lý. □

4.4. THỰC NGHIỆM VÀ KẾT QUẢ

Trong phần này, luận án tiến hành các thí nghiệm nhằm đánh giá hiệu năng của hai thuật toán đề xuất TGFastU và TGFast đối với bài toán SMFC trong bối cảnh tối đa hóa ảnh hưởng. Cụ thể, các thí nghiệm được thiết kế để đối chiếu các phương pháp đề xuất với những thuật toán hiện có dựa trên hai tiêu chí chính: (i) *chất lượng nghiệm*, được đo lường thông qua mức độ lan truyền ảnh hưởng đạt được, và (ii) *thời gian thực thi* của thuật toán.

4.4.1. Thiết lập thực nghiệm

Các thí nghiệm được thiết kế nhằm đánh giá hiệu năng của thuật toán đề xuất TGFastU và TGFast thông qua việc so sánh với phương pháp tiên tiến OPIMC [112] cho bài toán Tối đa ảnh hưởng (IM). Việc đối sánh được thực hiện dựa trên hai tiêu chí chính: (1) *chất lượng nghiệm* (mức độ lan truyền ảnh hưởng đạt được) và (2) *thời gian thực thi*. Mô tả ngắn gọn các thuật toán so sánh như sau.

- **OPIMC** [112]: OPIMC là một trong những TTXH hiệu quả nhất cho bài toán IM, với mục tiêu tìm tập hạt giống S có $|S| \leq k$ sao cho giá trị lan truyền ảnh hưởng $\mathbb{I}(S)$ là lớn nhất. Tuy nhiên, do lời giải của bài toán IM có thể không phải là lời giải hợp lệ đối với bài toán SMFC, LA điều chỉnh OPIM để áp dụng cho bài toán SMFC thông qua các sửa đổi hợp lý như sau:

- Trước hết, sinh $N_{\max}$ tập mẫu tương tự như trong [112].
- Với mỗi nhóm C_i , OPIMC được điều chỉnh để lần lượt lựa chọn người dùng có *tỷ lệ ảnh hưởng trên chi phí* lớn nhất trong khi tập nghiệm S thỏa mãn cả ràng buộc chi phí theo nhóm và ràng buộc chi phí tổng.

Nhờ cơ chế này, mọi tập hạt giống được tạo ra đều là nghiệm hợp lệ của bài toán SMFC, tức luôn bảo đảm tính công bằng theo ngân sách nhóm trong suốt quá trình xây dựng lời giải.

- **Max Degree**: Đây là một phương pháp heuristic đơn giản nhưng được sử dụng rộng rãi trong các bài toán lan truyền ảnh hưởng trên OSN [94, 1, 113]. Thuật toán thực hiện việc sắp xếp các đỉnh theo thứ tự giảm dần của bậc, sau đó lần lượt lựa chọn và thêm các đỉnh vào tập nghiệm, miễn là vẫn thỏa mãn các ràng buộc về chi phí tổng cũng như chi phí trên từng nhóm.

Dữ liệu. Các thí nghiệm sử dụng các tập dữ liệu OSN công khai từ SNAP¹, được liệt kê trong Bảng 4.2. Những bộ dữ liệu này đã được sử dụng phổ biến trong nhiều nghiên cứu trước đây [114, 112, 113, 100].

Thiết lập tham số. Tất cả các thí nghiệm được tiến hành dưới mô hình IC, trong đó xác suất lan truyền trên mỗi cạnh (u, v) được thiết lập là $p(u, v) = 1/|N_{in}(v)|$. Cách thiết lập này kế thừa từ các nghiên cứu trước [1, 113, 100, 112]. Trong quá trình thực nghiệm, ngân sách B được thay đổi trong tập 100, 150, 200, 250, 300, 350, 400 cho từng bộ dữ liệu. Đồng thời, số nhóm được cố định ở mức $K = 10$ và tỷ lệ công bằng $\alpha = 0.1$. Các tham

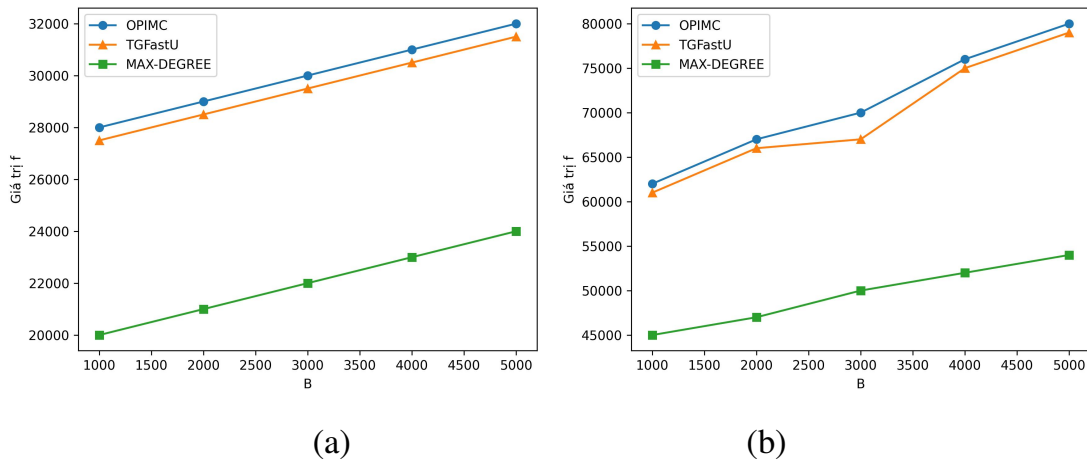
¹<https://snap.stanford.edu/data/>

Bảng 4.2: Các tập dữ liệu

Tập dữ liệu	Số đỉnh	Số cạnh	Loại
Musae-Github	37.700	289.003	Vô hướng
Gemsec-Deezer	41.773	125.826	Vô hướng
Gemsec-Facebook	50.515	819.306	Vô hướng
Google Plus	211.200	1.506.896	Có hướng

số $\epsilon = 0.1$ và $\delta = 0.1$ cũng được lựa chọn theo các thiết lập phổ biến trong các công trình trước [100, 112].

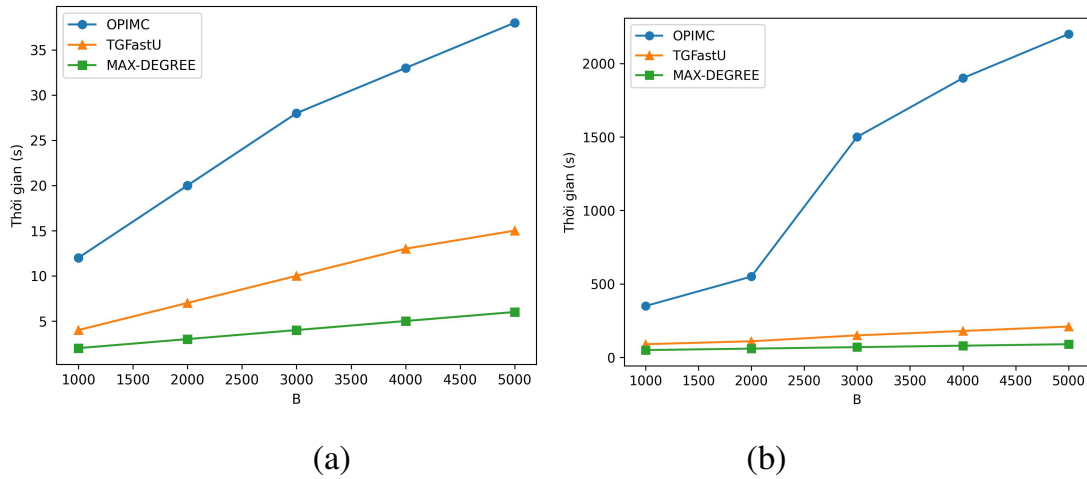
4.4.2. Kết quả thực nghiệm



Hình 4.1: Giá trị hàm f (ảnh hưởng) của các thuật toán với trường hợp chi phí đồng nhất trên dữ liệu (a) Facebook, (b) Google Plus.

a. Chất lượng lời giải Hình 4.1 và Hình 4.3 trình bày giá trị của hàm f (tương ứng với mức độ lan truyền ảnh hưởng) thu được bởi các thuật toán trên các bộ dữ liệu khác nhau.

Trong **trường hợp chi phí đồng nhất**, kết quả thể hiện trong Hình 4.1 cho thấy thuật toán OPIMC đạt giá trị lan truyền ảnh hưởng nhỉnh hơn so với các phương pháp còn lại trên bộ dữ liệu Facebook. Tuy nhiên, mức chênh lệch này không đáng kể, luôn dưới 5% so với thuật toán TGFastU, qua đó cho thấy hai phương pháp có chất lượng lời giải tương đương trong bối cảnh này. Ngược lại, trên mạng Google Plus, thuật toán TGFastU đạt giá trị lan truyền cao hơn khi ngân sách tăng, phản ánh khả năng lựa chọn tập hạt giống hiệu quả của phương pháp đề xuất. Kết quả thực nghiệm cho thấy TGFastU đạt giá trị mục tiêu rất gần với OPIMC trên cả hai bộ dữ liệu. Cụ thể, khoảng cách giữa hai thuật



Hình 4.2: Thời gian chạy của các thuật toán với trường hợp chi phí đồng nhất trên dữ liệu (a) Facebook, (b) Google Plus.

toán chỉ dao động từ khoảng 1.3% đến 4.3%, và trong phần lớn các trường hợp thấp hơn 2%. Điều này cho thấy TGFastU duy trì chất lượng nghiệm gần tương đương OPIMC trong khi vẫn thỏa mãn các ràng buộc công bằng của bài toán.

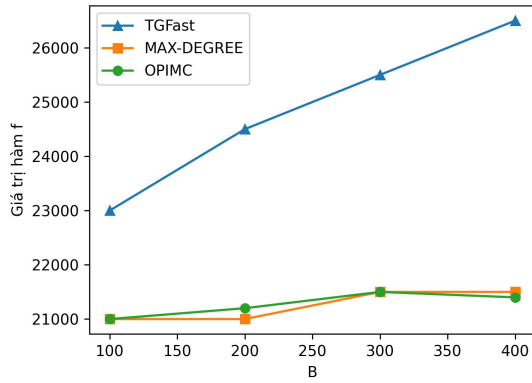
Trong trường hợp chi phí tổng quát, kết quả trong Hình 4.3 cho thấy thuật toán TGFastU luôn đạt giá trị hàm mục tiêu cao nhất trên tất cả các bộ dữ liệu, bao gồm Musae-Github, Gemsec-Deezer, Gemsec-Facebook và Google Plus. Cụ thể, khi ngân sách tăng, giá trị hàm f thu được bởi TGFast tăng ổn định và vượt trội so với hai thuật toán so sánh là OPIMC và MAX-DEGREE. Kết quả này phù hợp với các phân tích lý thuyết đã trình bày trước đó, trong đó TGFast được thiết kế với các bảo đảm xấp xỉ cho trường hợp chi phí tổng quát, trong khi các phương pháp còn lại không có các đảm bảo tương ứng. Kết quả thực nghiệm cho thấy TGFast đạt giá trị mục tiêu vượt trội so với OPIMC trên cả hai bộ dữ liệu. Cụ thể, TGFast cải thiện từ khoảng 6% đến 31% so với OPIMC và duy trì ưu thế nhất quán trên mọi mức ngân sách. Điều này cho thấy việc khai thác hiệu quả ngân sách toàn cục và ngân sách theo nhóm giúp TGFast đạt chất lượng nghiệm tốt hơn đáng kể so với các phương pháp tối đa ảnh hưởng truyền thống.

Nhìn chung, các kết quả thực nghiệm cho thấy thuật toán đề xuất không chỉ duy trì chất lượng lời giải cạnh tranh trong trường hợp chi phí đồng nhất mà còn thể hiện hiệu năng vượt trội trong bối cảnh chi phí tổng quát

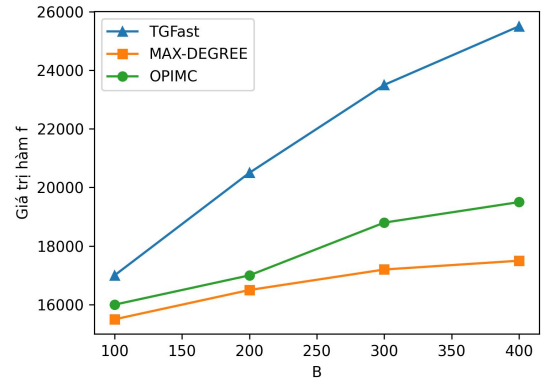
b. Thời gian

Hình 4.2 và Hình 4.4 trình bày thời gian thực thi của các thuật toán trên các bộ dữ liệu khác nhau.

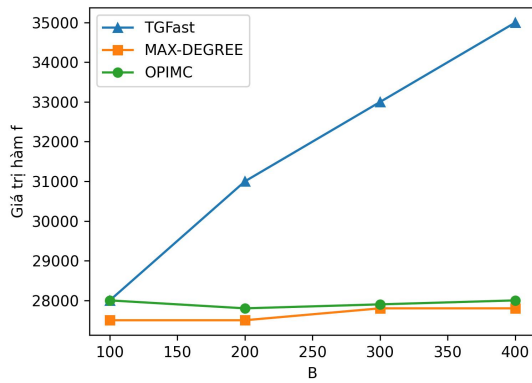
Trong **trường hợp chi phí đồng nhất**, kết quả ở Hình 4.2 cho thấy thời gian chạy



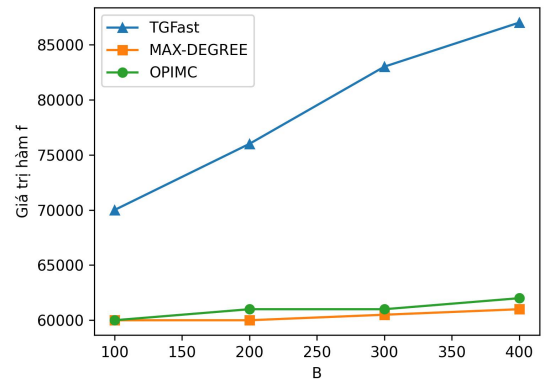
(a)



(b)

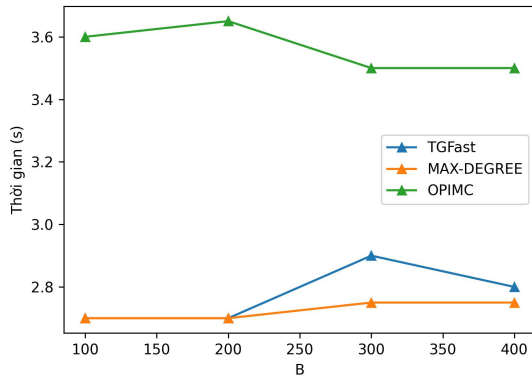


(c)

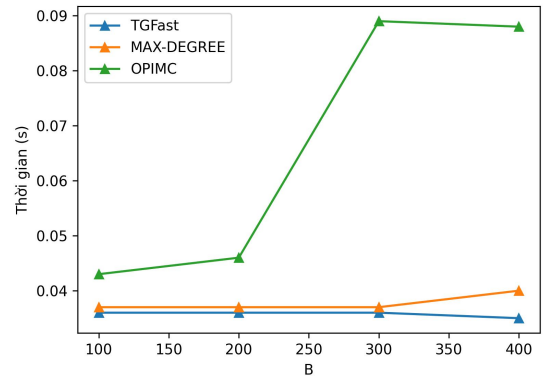


(d)

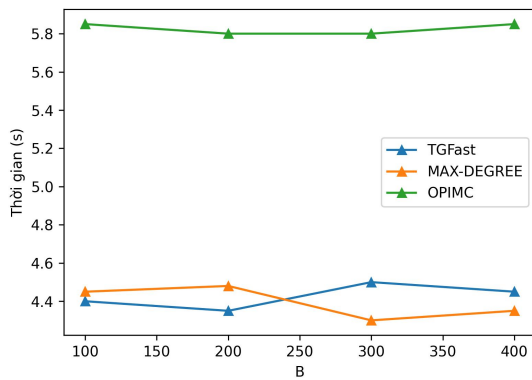
Hình 4.3: Giá trị hàm f (ảnh hưởng) của các thuật toán với trường hợp chi phí tổng quát trên các bộ dữ liệu: (a) Musae-Github, (b) Gemsec-Deezer, (c) Gemsec-Facebook và (d) Google Plus.



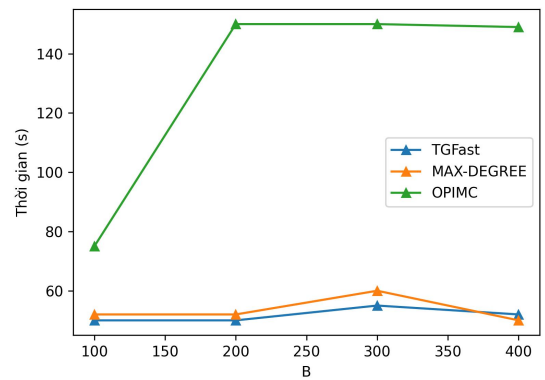
(a)



(b)



(c)



(d)

Hình 4.4: Thời gian chạy của các thuật toán với trường hợp chi phí tổng quát trên các bộ dữ liệu: (a) Musae-Github, (b) Gemsec-Deezer, (c) Gemsec-Facebook và (d) Google Plus.

của OPIMC tăng nhanh theo ngân sách B và luôn cao hơn đáng kể so với các phương pháp còn lại. Ngược lại, thuật toán đề xuất TGFastU duy trì thời gian thực thi thấp và tăng gần tuyến tính theo B . So với MAX-DEGREE, TGFastU có chi phí tính toán nhỉnh hơn nhưng vẫn thấp hơn rất nhiều so với OPIMC. Điều này cho thấy TGFastU đạt được sự cân bằng hợp lý giữa chất lượng nghiệm và chi phí tính toán.

Kết quả thực nghiệm cho thấy TGFastU có thời gian thực thi thấp hơn đáng kể so với OPIMC, với mức tăng tốc từ khoảng 2.5 lần đến hơn 10 lần trên các bộ dữ liệu khảo sát. Sự khác biệt này trở nên rõ rệt hơn khi ngân sách tăng, cho thấy TGFastU có khả năng mở rộng tốt hơn và phù hợp với các OSN quy mô lớn.

Trong **trường hợp chi phí tổng quát**, kết quả ở Hình 4.4 cho thấy thuật toán TGFast có thời gian chạy ổn định và tương đối thấp trên tất cả các bộ dữ liệu. Cụ thể, trên các bộ Musae-Github, Gemsec-Deezer và Gemsec-Facebook, thời gian thực thi của TGFast gần như không thay đổi khi ngân sách tăng và luôn thấp hơn rõ rệt so với OPIMC. Trên bộ Google Plus, sự khác biệt càng rõ khi OPIMC tăng rất nhanh theo B , trong khi TGFast vẫn giữ mức chi phí tính toán thấp. Kết quả thực nghiệm cho thấy TGFast có thời gian thực thi thấp hơn OPIMC trên tất cả các bộ dữ liệu, với mức tăng tốc dao động từ khoảng 1.2 lần đến 3 lần. Đặc biệt, trên các mạng có quy mô lớn hơn, TGFast thể hiện ưu thế rõ rệt khi thời gian chạy chỉ bằng khoảng 33%–67% so với OPIMC.

Nhìn chung, các kết quả thực nghiệm cho thấy các thuật toán đề xuất không chỉ đảm bảo chất lượng nghiệm tốt mà còn cải thiện đáng kể hiệu quả tính toán so với các phương pháp so sánh, đặc biệt trên các bộ dữ liệu lớn và trong bối cảnh chi phí tổng quát. Tổng hợp lại, các kết quả thực nghiệm cho thấy thuật toán đề xuất đạt được chất lượng lời giải tốt nhất trong khi vẫn duy trì thời gian chạy cạnh tranh. Đặc biệt, lợi thế của TGFast trở nên rõ rệt hơn khi kích thước tập dữ liệu tăng lên.

4.5. KẾT LUẬN CHƯƠNG

Trong chương này, luận án đã nghiên cứu bài toán *Tối đa hàm submodular với ràng buộc chi phí công bằng* (SMFC), một mô hình tối ưu mới có ý nghĩa thực tiễn trong nhiều ứng dụng như tối đa ảnh hưởng trong OSN, lựa chọn tập đại diện và phân bổ tài nguyên. Khác với các mô hình tối đa hóa submodular truyền thống chỉ xét ràng buộc ngân sách tổng, bài toán SMFC còn yêu cầu kiểm soát chi phí phân bổ trên từng nhóm phần tử, qua đó phản ánh tốt hơn các yêu cầu công bằng trong các hệ thống thực tế.

Về mặt lý thuyết, luận án đã xây dựng mô hình tối ưu cho bài toán và đề xuất hai TTTX tương ứng với hai thiết lập chi phí khác nhau. Trong **trường hợp chi phí đồng nhất**, luận án phát triển thuật toán TGFastU dựa trên chiến lược tham lam theo ngưỡng,

đạt tỷ lệ xấp xỉ $1/2 - \epsilon$ với độ phức tạp thời gian $O\left(\frac{n}{\epsilon} \log \frac{k}{\epsilon}\right)$. Đối với **trường hợp chi phí tổng quát**, luận án đề xuất thuật toán TGFast, mở rộng ý tưởng tham lam theo ngưỡng thông qua tiêu chí mật độ lợi ích trên chi phí, đồng thời chứng minh tỷ lệ xấp xỉ $\frac{\alpha}{\alpha+2} - \epsilon$ với độ phức tạp truy vấn $O\left(\frac{n}{\epsilon} \log(n \frac{\alpha+2}{\alpha})\right)$. Các kết quả này cho thấy phương pháp tiếp cận đề xuất không chỉ có các bảo đảm lý thuyết rõ ràng mà còn đạt hiệu quả tính toán tốt.

Về thực nghiệm, các thuật toán được đánh giá trên nhiều bộ dữ liệu OSN thực tế dưới mô hình lan truyền ảnh hưởng. Kết quả cho thấy, trong trường hợp chi phí đồng nhất, thuật toán TGFastU đạt chất lượng nghiệm cạnh tranh với phương pháp tiên tiến OPIMC, thậm chí vượt trội trên một số bộ dữ liệu, đồng thời vẫn duy trì thời gian thực thi thấp hơn đáng kể. Đối với chi phí tổng quát, thuật toán TGFast thể hiện ưu thế rõ rệt so với các phương pháp so sánh cả về giá trị hàm mục tiêu lẫn hiệu quả tính toán, đặc biệt trên các tập dữ liệu quy mô lớn. Những kết quả này phù hợp với các phân tích lý thuyết đã được thiết lập trước đó.

Mặc dù đã đạt được các kết quả khả quan, bài toán SMFC vẫn còn nhiều hướng nghiên cứu mở. Trong tương lai, có thể mở rộng các thuật toán đề xuất cho trường hợp hàm submodular không đơn điệu, các mô hình dữ liệu luồng hoặc phân tán, cũng như các ràng buộc công bằng phức tạp hơn. Bên cạnh đó, việc cải thiện tỷ lệ xấp xỉ và giảm độ phức tạp tính toán cho các mạng quy mô rất lớn vẫn là những thách thức quan trọng cả về mặt lý thuyết lẫn thực tiễn.

KẾT LUẬN VÀ KIẾN NGHỊ

Luận án tập trung nghiên cứu lớp bài toán SO bao ba bài toán: (1) Phủ submodular với chi phí tối thiểu (MCSC), (2) Phủ k -Submodular với chi phí tối thiểu (kSC) và (3) Tối đa hóa hàm submodular dưới ràng buộc chi phí công bằng (SMFC). Đây là một lớp bài toán có khả năng mô hình hóa linh hoạt nhiều ứng dụng thực tiễn, bao gồm tối đa hóa ảnh hưởng trên mạng xã hội, lựa chọn đặc trưng, hệ thống khuyến nghị, phân bổ ngân sách tiếp thị, cũng như các bài toán phân bổ tài nguyên dưới các ràng buộc công bằng. Trong các ứng dụng như lan truyền thông tin trên mạng xã hội, tiếp thị và phân bổ tài nguyên, tối ưu submodular với chi phí giúp lựa chọn phương án đạt hiệu quả cao trong điều kiện nguồn lực hữu hạn. Chẳng hạn, đối với hoạt động truyền thông, MCSC mô hình hóa yêu cầu đạt mức ảnh hưởng mong muốn với chi phí nhỏ nhất, còn kSC mở rộng yêu cầu này sang việc phân bổ các hạt giống cho nhiều chủ đề. SMFC hướng đến tối đa hóa hiệu quả truyền thông dưới các giới hạn chi phí, đồng thời bảo đảm công bằng trong phân bổ ngân sách giữa các nhóm người dùng. Ba bài toán vì thế gắn với những yêu cầu bổ sung cho nhau trong cùng một ứng dụng: tiết kiệm chi phí, phân bổ nguồn lực cho nhiều thành phần và bảo đảm công bằng. Việc nghiên cứu các mô hình này không chỉ có ý nghĩa về mặt lý thuyết mà còn đóng vai trò nền tảng trong thiết kế các hệ thống phân tích dữ liệu hiện đại. Trong khuôn khổ luận án, các kết quả chính đạt được bao gồm:

1. **Đối với bài toán MCSC:** Luận án đề xuất một thuật toán song song hiệu quả cho bài toán phủ submodular với chi phí tối thiểu. Thuật toán đạt tỷ lệ xấp xỉ tiệm cận với các kết quả tốt nhất hiện nay, đồng thời cải thiện đáng kể độ phức tạp thích nghi và khả năng song song hóa. Các phân tích lý thuyết chứng minh chất lượng nghiệm của thuật toán, trong khi các thực nghiệm trên dữ liệu thực cho thấy ưu thế rõ rệt về thời gian thực thi và khả năng mở rộng. Kết quả này góp phần thu hẹp khoảng cách giữa các đảm bảo lý thuyết và yêu cầu triển khai trong thực tế.

2. **Đối với bài toán kSC:** Luận án mở rộng mô hình phủ submodular sang miền k -submodular và đề xuất các thuật toán xấp xỉ tiêu chuẩn kép cho cả trường hợp chi phí đồng nhất và chi phí tổng quát. Các thuật toán đạt được hệ số xấp xỉ tốt hơn các kết quả trước đây, đồng thời giảm đáng kể độ phức tạp truy vấn hàm mục tiêu. Kết quả thực nghiệm cho thấy các phương pháp đề xuất duy trì được chất lượng nghiệm cao trong khi giảm đáng kể chi phí tính toán, qua đó nâng cao khả năng ứng dụng đối với các bài toán dữ liệu lớn.

3. **Đối với bài toán SMFC:** Luận án nghiên cứu một mô hình tối ưu mới kết hợp

giữa hiệu quả tối ưu và yêu cầu công bằng trong phân bổ chi phí. Hai thuật toán xấp xỉ được đề xuất tương ứng với trường hợp chi phí đồng nhất và chi phí tổng quát, kèm theo các bảo đảm lý thuyết về hệ số xấp xỉ. Các kết quả đạt được mở rộng phạm vi nghiên cứu của tối ưu submodular sang các hệ thống có ràng buộc công bằng, góp phần giải quyết các yêu cầu ngày càng quan trọng trong các ứng dụng dữ liệu và mạng xã hội hiện đại.

Mặc dù đã đạt được các kết quả lý thuyết và thực nghiệm quan trọng, các nghiên cứu trong luận án vẫn còn một số hạn chế nhất định. Các thuật toán được đề xuất chủ yếu được nghiên cứu trong các mô hình tối ưu tĩnh và chưa xem xét đầy đủ các đặc trưng của dữ liệu quy mô rất lớn hoặc dữ liệu thay đổi theo thời gian. Bên cạnh đó, các mô hình được khảo sát mới tập trung vào một số dạng ràng buộc chi phí và công bằng cơ bản, trong khi nhiều ứng dụng thực tế thường đòi hỏi các ràng buộc phức tạp hơn. Trong thời gian tới, các kết quả của luận án có thể được mở rộng theo một số hướng nghiên cứu chính sau.

Thứ nhất, phát triển các thuật toán tối ưu submodular và k -submodular trong các mô hình xử lý dữ liệu quy mô lớn như mô hình luồng (streaming), song song và phân tán. Đồng thời, nghiên cứu các kỹ thuật giảm số lần truy vấn hàm mục tiêu nhằm nâng cao khả năng áp dụng đối với các tập dữ liệu có kích thước rất lớn.

Thứ hai, mở rộng các mô hình tối ưu hiện tại sang các dạng ràng buộc phức tạp hơn, chẳng hạn như nhiều ràng buộc ngân sách đồng thời, các yêu cầu công bằng đa mức hoặc các ràng buộc kết hợp phát sinh trong các bài toán thực tế.

Thứ ba, nghiên cứu các bài toán tối ưu trong môi trường động và bất định, nơi dữ liệu, chi phí hoặc các tham số của hệ thống có thể thay đổi theo thời gian. Đây là hướng nghiên cứu có tiềm năng ứng dụng cao trong các hệ thống mạng xã hội, hệ thống khuyến nghị và các nền tảng dữ liệu hiện đại.

DANH MỤC CÔNG TRÌNH CÔNG BỐ LIÊN QUAN ĐẾN LUẬN ÁN

1. **Hue T. Nguyen**, Dung T.K. Ha, Canh V. Pham. *Efficient Parallel Algorithm for Minimum Cost Submodular Cover Problem with Lower Adaptive Complexity*. Asia-Pacific Journal of Operational Research, 41(6), 2450005 (2024) (SCIE).
2. **Hue T. Nguyen**, Nguyen Long Giang, Canh V. Pham. *Efficient Deterministic Bicriteria Approximation Algorithms for k -Submodular Cover Problem*. Asia-Pacific Journal of Operational Research, 2650014 (2026) (SCIE).
3. **Hue T. Nguyen**, Nguyen Long Giang, Tran D. Tran, Canh V. Pham. *Thuật toán song song hiệu quả cho bài toán Phủ Submodular với chi phí tối thiểu và ứng dụng*. Kỷ yếu Hội thảo Quốc gia lần thứ XXVII Một số vấn đề chọn lọc về Công nghệ thông tin và Truyền thông, Tr. 622-634, 2024.
4. **Hue T. Nguyen**, Bac D. Pham, Uyen T. Tran, Nguyen Long Giang, Canh V. Pham. *Influence Maximization with Fairness Allocation Constraint*. In: Proceedings of the 13th International Symposium on Information and Communication Technology (SoICT 2024), 401–414 (2025) (SCOPUS).
5. **Hue T. Nguyen**, Bac D. Pham, Dung T. K. Ha, Nguyen Long Giang, Canh V. Pham. *Influence Maximization with Fairness Cost on Groups in Online Social Networks*. In: Proceedings of the 17th Asian Conference on Intelligent Information and Database Systems (ACIIDS 2025), Kitakyushu, Japan, April 23–25, 2025, pp. 191–206 (2025) (SCOPUS).
6. **Hue T. Nguyen**, Tan D. Tran, Nguyen Long Giang, Canh V. Pham. *Fast Stochastic Greedy Algorithm for k -Submodular Cover Problem*. In: Proceedings of the 14th International Symposium on Information and Communication Technology (SOICT 2025), pp. 173–184 (2025) (SCOPUS). (SCOPUS).

DANH MỤC TÀI LIỆU THAM KHẢO

- [1] David Kempe, Jon M. Kleinberg, Éva Tardos, 2003, Maximizing the spread of influence through a social network, *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 24 - 27, 2003*, volume 11, 137–146.
- [2] David Kempe, Jon M. Kleinberg, Éva Tardos, 2015, Maximizing the Spread of Influence through a Social Network, *Theory of Computing*, 11, (1), 105–147.
- [3] Kai Han, Zongmai Cao, Shuang Cui, Benwei Wu, 2020, Deterministic Approximation for Submodular Maximization over a Matroid in Nearly Linear Time, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.
- [4] Ashwinkumar Badanidiyuru, Baharan Mirzasoleiman, Amin Karbasi, Andreas Krause, 2014, Streaming submodular maximization: massive data summarization on the fly, *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014*, 671–680, ACM.
- [5] Kai Han, Shuang Cui, Tianshuai Zhu, Enpei Zhang, Benwei Wu, Zhizhuo Yin, Tong Xu, Shaojie Tang, He Huang, 2021, Approximation Algorithms for Submodular Data Summarization with a Knapsack Constraint, *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 5, (1), 5:1–5:31.
- [6] Akbar Rafiey, Yuichi Yoshida, 2020, Fast and Private Submodular and k-Submodular Functions Maximization with Matroid Constraints, *Proceedings of the 37th International Conference on Machine Learning ICML*, volume 119 of *Proc. of MLR*, 7887–7897.
- [7] Yixin Chen, Tonmoy Dey, Alan Kuhnle, 2021, Best of Both Worlds: Practical and Theoretically Optimal Submodular Maximization in Parallel, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, 25528–25539.
- [8] Laurence A. Wolsey, 1982, An analysis of the greedy algorithm for the submodular set covering problem, *Combinatorica*, 2, (4), 385–393.

- [9] Laurence A. Wolsey, 1982, Maximising Real-Valued Submodular Functions: Primal and Dual Heuristics for Location Problems, *Mathematics of Operations Research*, 7, (3), 410–425.
- [10] Baharan Mirzasoleiman, Ashwinkumar Badanidiyuru, Amin Karbasi, 2016, Fast Constrained Submodular Maximization: Personalized Data Summarization, *Proceedings of the 33rd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, 1358–1367, JMLR.org.
- [11] Naoto Ohsaka, Yuichi Yoshida, 2015, Monotone k-Submodular Function Maximization with Size Constraints, *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems*, 694–702.
- [12] Tan D. Tran, Canh V. Pham, Dung T. K. Ha, Phuong N. H. Pham, 2024, Improved Parallel Algorithm for Non-Monotone Submodular Maximization under Knapsack Constraint, *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI-24*, 1961–1969, International Joint Conferences on Artificial Intelligence Organization, Main Track.
- [13] Canh V. Pham, Tan D. Tran, Dung T.K. Ha, My T. Thai, 2023, Linear Query Approximation Algorithms for Non-monotone Submodular Maximization under Knapsack Constraint, *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI-23*, 4127–4135, International Joint Conferences on Artificial Intelligence Organization, Main Track.
- [14] George L. Nemhauser, Laurence A. Wolsey, Marshall L. Fisher, 1978, An analysis of approximations for maximizing submodular set functions - I, *Math. Program.*, 14, (1), 265–294.
- [15] Arkadi S. Nemirovsky, David B. Yudin, 1983, *Problem Complexity and Method Efficiency in Optimization*, Wiley-Interscience Series in Discrete Mathematics, John Wiley & Sons, Translated from the Russian edition (1979).
- [16] Peng-Jun Wan, Ding-Zhu Du, Panos M. Pardalos, Weili Wu, 2010, Greedy approximations for minimum submodular cover with submodular cost, *Comput. Optim. Appl.*, 45, (2), 463–474.
- [17] Maxim Sviridenko, 2004, A note on maximizing a submodular set function subject to a knapsack constraint, *Oper. Res. Lett.*, 32, (1), 41–43.

- [18] Wenxin Li, Moran Feldman, Ehsan Kazemi, Amin Karbasi, 2022, Submodular Maximization in Clean Linear Time, *Advances in Neural Information Processing Systems 35 (NeurIPS 2022)*, 17473–17487.
- [19] Yingli Ran, Zhao Zhang, Shaojie Tang, 2022, Improved Parallel Algorithm for Minimum Cost Submodular Cover Problem, *Proc. of the Learning Theory, 2022*, volume 178 of *Proceedings of MLS*, 3490–3502, PMLR.
- [20] Alan Kuhnle, Victoria G. Crawford, My T. Thai, 2017, Scalable and Adaptive Algorithms for the Triangle Interdiction Problem on Billion-Scale Networks, *2017 IEEE International Conference on Data Mining, ICDM 2017, New Orleans, LA, USA, November 18-21, 2017*, 237–246, IEEE Computer Society.
- [21] Sebastian Tschieschek, Rishabh K. Iyer, Haochen Wei, Jeff A. Bilmes, 2014, Learning Mixtures of Submodular Functions for Image Collection Summarization, *Proceedings of the 28th International Conference on Neural Information Processing Systems*, 1413–1421, MIT Press.
- [22] Ashkan Norouzi-Fard, Abbas Bazzi, Ilija Bogunovic, Marwa El Halabi, Ya-Ping Hsieh, Volkan Cevher, 2016, An Efficient Streaming Algorithm for the Submodular Cover Problem, *Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS)*, volume 29, 4493–4501.
- [23] Alan Kuhnle, 2021, Quick Streaming Algorithms for Maximization of Monotone Submodular Functions in Linear Time, *Proceedings of the 24th International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 130, 1360–1368.
- [24] Ashwinkumar Badanidiyuru, Jan Vondrák, 2014, Fast algorithms for maximizing submodular functions, *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, 1497–1514, SIAM.
- [25] Niv Buchbinder, Moran Feldman, 2024, Constrained Submodular Maximization via New Bounds for DR-Submodular Functions, *Proceedings of the 56th Annual ACM Symposium on Theory of Computing, STOC 2024, Vancouver, BC, Canada, June 24-28, 2024*, 1820–1831, ACM.
- [26] Georgios Amanatidis, Federico Fusco, Philip Lazos, Stefano Leonardi, Rebecca Reiffenhäuser, 2020, Fast Adaptive Non-Monotone Submodular Maximization

Subject to a Knapsack Constraint, *Annual Conference on Neural Information Processing Systems*.

- [27] Xiaoming Sun, Jialin Zhang, Shuo Zhang, Zhijie Zhang, 2024, Improved Deterministic Algorithms for Non-monotone Submodular Maximization, *Theoretical Computer Science*, 984, 114293.
- [28] Yixin Chen, Ankur Nath, Chunli Peng, Alan Kuhnle, 2024, Discretely Beyond $1/e$: Guided Combinatorial Algorithms for Submodular Maximization, *Advances in Neural Information Processing Systems 37 (NeurIPS 2024)*, 108929–108973.
- [29] Yixin Chen, Tonmoy Dey, Alan Kuhnle, 2021, Best of Both Worlds: Practical and Theoretically Optimal Submodular Maximization in Parallel, *Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS)*, 25528–25539.
- [30] Eric Balkanski, Aviad Rubinstein, Yaron Singer, 2019, An optimal approximation for submodular maximization under a matroid constraint in the adaptive complexity model, *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*, 66–77, ACM.
- [31] Eric Balkanski, Aviad Rubinstein, Yaron Singer, 2019, An Exponential Speedup in Parallel Running Time for Submodular Maximization without Loss in Approximation, *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, 283–302, SIAM.
- [32] Eric Balkanski, Adam Breuer, Yaron Singer, 2018, Non-monotone Submodular Maximization in Exponentially Fewer Iterations, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, 2359–2370.
- [33] Ruiqi Yang, Dachuan Xu, Yukun Cheng, Chuangen Gao, Ding-Zhu Du, 2019, Streaming Submodular Maximization Under Noises, *Proc. of IEEE ICDCS 2019*, 348–357.
- [34] Eric Balkanski, Yaron Singer, 2018, The adaptive complexity of maximizing a submodular function, *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, 1138–1151, ACM.

- [35] Golnoosh Farnadi, Behrouz Babaki, Michel Gendreau, 2020, A Unifying Framework for Fairness-Aware Influence Maximization, *Companion of The 2020 Web Conference 2020, Taipei, Taiwan, April 20-24, 2020*, 714–722.
- [36] Marwa El Halabi, Federico Fusco, Ashkan Norouzi-Fard, Jakab Tardos, Jakub Tarnawski, 2023, Fairness in Streaming Submodular Maximization over a Matroid Constraint, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, 9150–9171, PMLR.
- [37] Alan Tsang, Bryan Wilder, Eric Rice, Milind Tambe, Yair Zick, 2019, Group-Fairness in Influence Maximization, *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, 5997–6005.
- [38] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein, 2009, *Introduction to Algorithms*, MIT Press, 3 edition.
- [39] Abhimanyu Das, David Kempe, 2018, Approximate Submodularity and its Applications: Subset Selection, Sparse Approximation and Dictionary Selection, *Jour. of MLR*, 19, (3), 1–34.
- [40] Qilian Yu, Easton Li Xu, Shuguang Cui, 2016, Submodular maximization with multi-knapsack constraints and its applications in scientific literature recommendations, *2016 IEEE Global Conference on Signal and Information Processing, GlobalSIP 2016, Washington, DC, USA, December 7-9, 2016*, 1295–1299, IEEE.
- [41] Andreas Krause, Ajit Paul Singh, Carlos Guestrin, 2008, Near-Optimal Sensor Placements in Gaussian Processes: Theory, Efficient Algorithms and Empirical Studies, *Journal of Machine Learning Research (JMLR)*, 9, 235–284.
- [42] Laurence A. Wolsey, 1982, An analysis of the greedy algorithm for the submodular set covering problem, *Combinatorica*, 2, (4), 385–393.
- [43] Wei Chen, Laks V. S. Lakshmanan, Carlos Castillo, 2013, *Information and Influence Propagation in Social Networks*, Synthesis Lectures on Data Management, Morgan & Claypool Publishers.
- [44] Jason D. Hartline, Vahab S. Mirrokni, Mukund Sundararajan, 2008, Optimal marketing strategies over social networks, *Proceedings of the 17th International Con-*

ference on World Wide Web, WWW 2008, Beijing, China, April 21-25, 2008, 189–198, ACM.

- [45] Alan Kuhnle, 2021, Nearly Linear-Time, Parallelizable Algorithms for Non-Monotone Submodular Maximization, *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 8200–8208.
- [46] Richard M. Karp, 2009, Reducibility among combinatorial problems, *50 Years of Integer Programming 1958–2008: From the Early Years to the State-of-the-Art*, 219–241, Springer, Reprint of the 1972 original.
- [47] Vijay V. Vazirani, 2003, *Approximation algorithms*, Springer.
- [48] Uriel Feige, 1998, A Threshold of $\ln n$ for Approximating Set Cover, *J. ACM*, 45, (4), 634–652.
- [49] Madhav V Marathe, R Ravi, Ravi Sundaram, S.S Ravi, Daniel J Rosenkrantz, Harry B Hunt, 1998, Bicriteria Network Design Problems, *Journal of Algorithms*, 28, (1), 142–171.
- [50] Noga Alon, Yossi Matias, Mario Szegedy, 1996, The Space Complexity of Approximating the Frequency Moments, *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing (STOC)*, 20–29, ACM.
- [51] Francoise Tisseur, Jack Dongarra, 1999, A Parallel Divide and Conquer Algorithm for the Symmetric Eigenvalue Problem on Distributed Memory Architectures, *SIAM Journal on Scientific Computing*, 20, (6), 2223–2236.
- [52] Yijie Han, 1990, Parallel Algorithms for Linked List and Beyond, *Proceedings of the International Symposium on Algorithms*, 86–100, Springer.
- [53] Mark Harris, Shubhabrata Sengupta, John D. Owens, 2007, Parallel Prefix Sum (Scan) with CUDA, *GPU Gems 3*, chapter 39, NVIDIA Corporation.
- [54] Georgios Amanatidis, Federico Fusco, Philip Lazos, Stefano Leonardi, Alberto Marchetti-Spaccamela, Rebecca Reiffenhäuser, 2021, Submodular Maximization subject to a Knapsack Constraint: Combinatorial Algorithms with Near-optimal Adaptive Complexity, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, 231–242, PMLR.

- [55] Alan Kuhnle, 2021, Quick Streaming Algorithms for Maximization of Monotone Submodular Functions in Linear Time, *Proceedings of the 24th International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 130 of *Proc. of MLR*, 1360–1368.
- [56] Adam Breuer, Eric Balkanski, Yaron Singer, 2020, The FAST Algorithm for Submodular Maximization, *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, 1134–1143, PMLR.
- [57] Maxim Sviridenko, 2004, A note on maximizing a submodular set function subject to a knapsack constraint, *Operations Research Letters*, 32, (1), 41–43.
- [58] Jure Leskovec, Andreas Krause, Carlos Guestrin, Christos Faloutsos, Jeanne M. VanBriesen, Natalie S. Glance, 2007, Cost-effective outbreak detection in networks, *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Jose, California, USA, August 12-15, 2007*, 420–429.
- [59] Moran Feldman, Zeev Nutov, Elad Shoham, 2023, Practical Budgeted Submodular Maximization, *Algorithmica*, 85, (5), 1332–1371.
- [60] Alina Ene, Huy L. Nguyen, 2019, A Nearly-Linear Time Algorithm for Submodular Maximization with a Knapsack Constraint, *Proceedings of the 46th International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 132 of *LIPIcs*, 53:1–53:12.
- [61] Chien-Chung Huang, Naonori Kakimura, Yuichi Yoshida, 2020, Streaming Algorithms for Maximizing Monotone Submodular Functions Under a Knapsack Constraint, *Algorithmica*, 82, (4), 1006–1032.
- [62] Chien-Chung Huang, Naonori Kakimura, 2021, Improved Streaming Algorithms for Maximizing Monotone Submodular Functions under a Knapsack Constraint, *Algorithmica*, 83, (3), 879–902.
- [63] Chien-Chung Huang, Naonori Kakimura, 2022, Multi-Pass Streaming Algorithms for Monotone Submodular Function Maximization, *Theory Comput. Syst.*, 66, (1), 354–394.
- [64] Amit Goyal, Francesco Bonchi, Laks V. S. Lakshmanan, Suresh Venkatasubrama-

- nian, 2013, On minimizing budget and time in influence propagation over social networks, *Social Network Analysis and Mining*, 3, (2), 179–192.
- [65] Tasuku Soma, Yuichi Yoshida, 2015, A Generalization of Submodular Cover via the Diminishing Return Property on the Integer Lattice, *Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS)*, 847–855.
- [66] Victoria G. Crawford, 2019, An Efficient Evolutionary Algorithm for Minimum Cost Submodular Cover, *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, 1227–1233, ijcai.org.
- [67] Uriel Feige, 1998, A Threshold of $\ln n$ for Approximating Set Cover, *J. ACM*, 45, (4), 634–652.
- [68] Justin Ward, Stanislav Zivný, 2014, Maximizing Bisubmodular and k -Submodular Functions, *Proceedings of the 2014 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 1468–1481.
- [69] Satoru Iwata, Shin-ichi Tanigawa, Yuichi Yoshida, 2016, Improved Approximation Algorithms for k -Submodular Function Maximization, *Proceedings of the 2016 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 404–413, SIAM.
- [70] Shinsaku Sakaue, 2017, On maximizing a monotone k -submodular function subject to a matroid constraint, *Discrete Optimization*, 23, 105–113.
- [71] Zhongzheng Tang, Chenhao Wang, Hau Chan, 2022, On maximizing a monotone k -submodular function under a knapsack constraint, *Operations Research Letters*, 50, (1), 28–31.
- [72] Baoxiang Wang, Huanjian Zhou, 2021, Multilinear extension of k -submodular functions, *arXiv preprint arXiv:2107.07103*.
- [73] Lan Nguyen, My Thai, 2020, Streaming k -Submodular Maximization under Noise subject to Size Constraint, *Proceedings of the 37th International Conference on Machine Learning ICML 2020*, 7338–7347.
- [74] Phuong N. H. Pham, Quy T. N. Co, Anh N. Su, Phuong T. Pham, Canh V. Pham, Václav Snásel, 2022, k -Submodular Cover Problem: Application and Algorithms, *The 11th International Symposium on Information and Communication Technology, SoICT 2022, Hanoi, Vietnam, December 1-3, 2022*, 442–449, ACM.

- [75] Wenqi Wang, Gregory Z. Gutin, Yaping Mao, Donglei Du, Xiaoyan Zhang, 2023, Streaming Algorithms for the k-Submodular Cover Problem, *arXiv preprint arXiv:2312.03593*.
- [76] Francis R. Bach, 2013, Learning with Submodular Functions: A Convex Optimization Perspective, *Found. Trends Mach. Learn.*, 6, (2-3), 145–373.
- [77] Shameem Ahamed Puthiya Parambath, Nishant Vijayakumar, Sanjay Chawla, 2018, SAGA: A Submodular Greedy Algorithm for Group Recommendation, *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), New Orleans, Louisiana, USA, February 2-7, 2018*, 3900–3908, AAAI Press.
- [78] Hui Lin, Jeff A. Bilmes, 2011, A Class of Submodular Functions for Document Summarization, *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Conference, 19-24 June, 2011, Portland, Oregon, USA*, 510–520, The Association for Computer Linguistics.
- [79] Jan Vondrák, 2008, Optimal approximation for the submodular welfare problem in the value oracle model, *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*, 67–74, ACM.
- [80] Canh V. Pham, Quat V. Phu, Huan X. Hoang, Jun Pei, My T. Thai, 2019, Minimum budget for misinformation blocking in online social networks, *J. Comb. Optim.*, 38, (4), 1101–1127.
- [81] Alan Kuhnle, Tianyi Pan, Md Abdul Alim, My T. Thai, 2017, Scalable bicriteria algorithms for the threshold activation problem in online social networks, *2017 IEEE Conference on Computer Communications, INFOCOM 2017, Atlanta, GA, USA, May 1-4, 2017*, 1–9, IEEE.
- [82] Alan Kuhnle, J. David Smith, Victoria G. Crawford, My T. Thai, 2018, Fast Maximization of Non-Submodular, Monotonic Functions on the Integer Lattice, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, 2791–2800, PMLR.

- [83] Victoria G. Crawford, Alan Kuhnle, My T. Thai, 2019, Submodular Cost Submodular Cover with an Approximate Oracle, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, 1426–1435, PMLR.
- [84] Ashkan Norouzi-Fard, Abbas Bazzi, Ilija Bogunovic, Marwa El Halabi, Ya-Ping Hsieh, Volkan Cevher, 2016, An Efficient Streaming Algorithm for the Submodular Cover Problem, *Advances in Neural Information Processing Systems*, volume 29, 4493–4501.
- [85] Baharan Mirzasoleiman, Amin Karbasi, Ashwinkumar Badanidiyuru, Andreas Krause, 2015, Distributed Submodular Cover: Succinctly Summarizing Massive Data, *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, 2881–2889.
- [86] Baharan Mirzasoleiman, Morteza Zadimoghaddam, Amin Karbasi, 2016, Fast Distributed Submodular Cover: Public-Private Data Summarization, *Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS)*, 3594–3602.
- [87] Canh V. Pham, Quat V. Phu, Huan X. Hoang, 2018, Targeted Misinformation Blocking on Online Social Networks, *Intelligent Information and Database Systems - 10th Asian Conference, ACIIDS 2018, Dong Hoi City, Vietnam, March 19-21, 2018, Proceedings, Part I*, volume 10751 of *Lecture Notes in Computer Science*, 107–116, Springer.
- [88] Guy E. Blelloch, Richard Peng, Kanat Tangwongsan, 2011, Linear-work greedy parallel approximate set cover and variants, *Proc. of the 23rd ACMSPAA 2011*, 23–32, ACM.
- [89] Matthew Fahrbach, Vahab S. Mirrokni, Morteza Zadimoghaddam, 2019, Submodular Maximization with Nearly Optimal Approximation, Adaptivity and Query Complexity, *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, 255–273, SIAM.
- [90] Arpit Agarwal, Sepehr Assadi, Sanjeev Khanna, 2019, Stochastic Submodular Cover with Limited Adaptivity, *Proceedings of the Thirtieth Annual ACM-SIAM*

Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019, 323–342, SIAM.

- [91] Matthew Fahrbach, Vahab S. Mirrokni, Morteza Zadimoghaddam, 2019, Non-monotone Submodular Maximization with Nearly Optimal Adaptivity and Query Complexity, *Proceedings of the 36th International Conference on Machine Learning (ICML)*, volume 97 of *Proceedings of Machine Learning Research*, 1833–1842, PMLR.
- [92] Jure Leskovec, Andrej Krevl, 2014, SNAP Datasets: Stanford Large Network Dataset Collection, <http://snap.stanford.edu/data>.
- [93] Xiang Li, J. David Smith, Thang N. Dinh, My T. Thai, 2019, TipTop: (Almost) Exact Solutions for Influence Maximization in Billion-Scale Networks, *IEEE/ACM Transactions on Networking*, 27, (2), 649–661.
- [94] Youze Tang, Xiaokui Xiao, Yanchen Shi, 2014, Influence maximization: near-optimal time complexity meets practical efficiency, *International Conference on Management of Data, SIGMOD 2014, Snowbird, UT, USA, June 22-27, 2014*, 75–86.
- [95] Dung T. K. Ha, Canh V. Pham, Tan D. Tran, 2024, Improved approximation algorithms for k -submodular maximization under a knapsack constraint, *Comput. Oper. Res.*, 161, 106452.
- [96] Chao Qian, Jing-Cheng Shi, Ke Tang, Zhi-Hua Zhou, 2018, Constrained Monotone k -Submodular Function Maximization Using Multiobjective Evolutionary Algorithms With Theoretical Guarantee, *IEEE Trans. Evol. Comput.*, 22, (4), 595–608.
- [97] Leqian Zheng, Hau Chan, Grigorios Loukides, Minming Li, 2021, Maximizing Approximately k -Submodular Functions, *Proceedings of the 2021 SIAM International Conference on Data Mining, SDM 2021, Virtual Event, April 29 - May 1, 2021*, 414–422, SIAM.
- [98] Canh V. Pham, Quang C. Vu, Dung K. Ha, Tai T. Nguyen, Nguyen D. Le, 2022, Maximizing k -submodular functions under budget constraint: applications and streaming algorithms, *Journal of Combinatorial Optimization*, 44, (1), 723–751.
- [99] Ajit P. Singh, Andrew Guillory, Jeff A. Bilmes, 2012, On Bisubmodular Maxi-

mization, *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics AISTATS*, 1055–1063.

- [100] Hung T. Nguyen, My T. Thai, Thang N. Dinh, 2016, Stop-and-Stare: Optimal Sampling Algorithms for Viral Marketing in Billion-scale Networks, *Proceedings of the 2016 International Conference on Management of Data, SIGMOD Conference 2016, San Francisco, CA, USA, June 26 - July 01, 2016*, 695–710.
- [101] Jesús Mario Bilbao, 2000, *Cooperative Games on Combinatorial Structures*, Theory and Decision Library, Springer.
- [102] Yvan Saeys, Iñaki Inza, Pedro Larrañaga, 2007, A review of feature selection techniques in bioinformatics, *Bioinformatics*, 23, (19), 2507–2517.
- [103] Hiroki Oshima, 2017, Derandomization for k-Submodular Maximization, *Proceedings of the 28th International Workshop on Combinatorial Algorithms*, 88–99.
- [104] Tasuku Soma, 2019, No-regret algorithms for online k-submodular maximization, *Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 89 of *Proceedings of Machine Learning Research*, 1205–1214.
- [105] Alina Ene, Huy L. Nguyen, 2022, Streaming Algorithm for Monotone k-Submodular Maximization with Cardinality Constraints, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, 5944–5967.
- [106] Albert-László Barabási, Réka Albert, 1999, Emergence of scaling in random networks, *Science*, 286, (5439), 509–512.
- [107] Hung T. Nguyen, My T. Thai, Thang N. Dinh, 2017, A Billion-Scale Approximation Algorithm for Maximizing Benefit in Viral Marketing, *IEEE/ACM Trans. Netw.*, 25, (4), 2419–2429.
- [108] Ran Haba, Ehsan Kazemi, Moran Feldman, Amin Karbasi, 2020, Streaming Submodular Maximization under a k-Set System Constraint, *Proceedings of the 37th International Conference on Machine Learning ICML 2020*, volume abs/2002.03352, 3939–3949.
- [109] Andreas Krause, 2013, Submodularity in Machine Learning and Vision, *British Machine Vision Conference, BMVC 2013, Bristol, UK, September 9-13, 2013*.

- [110] Ariel Kulik, Hadas Shachnai, Tami Tamir, 2013, Approximations for Monotone and Nonmonotone Submodular Maximization with Knapsack Constraints, *Mathematics of Operations Research*, 38, (4), 729–739.
- [111] Marwa El Halabi, Slobodan Mitrović, Ashkan Norouzi-Fard, Jakab Tardos, Jakub Tarnawski, 2020, Fairness in Streaming Submodular Maximization: Algorithms and Hardness, *Advances in Neural Information Processing Systems 33 (NeurIPS 2020)*.
- [112] Jing Tang, Xueyan Tang, Xiaokui Xiao, Junsong Yuan, 2018, Online Processing Algorithms for Influence Maximization, *Proceedings of the 2018 International Conference on Management of Data, SIGMOD Conference 2018, Houston, TX, USA, June 10-15, 2018*, SIGMOD '18, 991–1005, Association for Computing Machinery.
- [113] Youze Tang, Yanchen Shi, Xiaokui Xiao, 2015, Influence Maximization in Near-Linear Time: A Martingale Approach, *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, Melbourne, Victoria, Australia, May 31 - June 4, 2015*, 1539–1554.
- [114] Lan N. Nguyen, Kunxiao Zhou, My T. Thai, 2019, Influence Maximization at Community Level: A New Challenge with Non-submodularity, *39th IEEE International Conference on Distributed Computing Systems, ICDCS 2019, Dallas, TX, USA, July 7-10, 2019*, 327–337.