Báo cáo Phân tích Chuyên sâu: "Graph Summarization: Compactness Meets Efficiency"

Phần 1: Bối cảnh, Vấn đề và Thách thức Cốt lõi trong Tóm tắt Đồ thị

1.1. Tầm quan trọng và Sự gia tăng của Dữ liệu Đồ thị Lớn

Trong kỷ nguyên số hiện nay, khối lượng và sự phổ biến của dữ liệu đồ thị đang gia tăng với tốc độ chưa từng có. Như được nhấn mạnh trong tài liệu nghiên cứu, "Khi khối lượng và sự phổ biến của đồ thị tăng lên, một biểu diễn đồ thị cô đọng trở nên thiết yếu để cho phép lưu trữ, truyền tải và xử lý đồ thị hiệu quả".[1] Đồ thị, với khả năng trừu tượng hóa các mối quan hệ giữa các thực thể, đã trở thành một cấu trúc dữ liệu nền tảng trong vô số lĩnh vực, từ các liên kết web, mạng xã hội, đến các hệ thống sinh học và mạng lưới giao thông.[1] Sự bùng nổ này đặt ra một yêu cầu cấp thiết về các phương pháp biểu diễn dữ liệu đồ thị một cách cô đọng, không chỉ để tiết kiệm tài nguyên phần cứng quý giá như bộ nhớ, đĩa và băng thông mạng, mà còn để tăng tốc các thuật toán bằng cách cho phép một phần lớn hơn của đồ thị nằm trong bộ nhớ cache.[1]

Nhu cầu về các biểu diễn đồ thị cô đọng không chỉ dừng lại ở việc tiết kiệm không gian lưu trữ. Quan trọng hơn, nó mở ra khả năng thực hiện các phân tích phức tạp trên những tập dữ liệu khổng lồ mà nếu không có các kỹ thuật này, sẽ trở nên bất khả thi về mặt tính toán hoặc tốn kém đến mức không thể chấp nhận được. Điều này có tác động trực tiếp và sâu rộng đến nhiều lĩnh vực nghiên cứu và ứng dụng thực tiễn, bao gồm phân tích mạng xã hội, tin sinh học, tìm kiếm web, và các hệ thống gợi ý, nơi mà việc khai thác thông tin từ các cấu trúc đồ thị lớn là chìa khóa để đưa ra những quyết định và khám phá có giá trị.

Trong bối cảnh đó, bài báo định vị "tóm tắt đồ thị" (graph summarization) là một dòng kỹ thuật đặc biệt hiệu quả không chỉ trong việc nén đồ thị mà còn trong việc "khám phá các mẫu cấu trúc".[1] Điều này cho thấy một xu hướng rõ ràng hướng tới các kỹ thuật nén không chỉ đơn thuần giảm kích thước dữ liệu mà còn mang lại giá trị phân tích, giúp người dùng hiểu rõ hơn về cấu trúc tiềm ẩn bên trong đồ thị. Trong khi các phương pháp khác như gán nhãn lại nút hoặc nén danh sách kề bằng mã hóa tham chiếu tập trung chủ yếu vào việc giảm kích thước, tóm tắt đồ thị còn nhằm mục đích làm nổi bật các nhóm nút có hành vi tương tự hoặc các cấu trúc quan trọng, qua đó cung cấp một cái nhìn tổng quan hơn về dữ liệu.

1.2. Định nghĩa Bài toán Tóm tắt Đồ thị Không mất mát

Bài toán tóm tắt đồ thị, như được định nghĩa trong nghiên cứu, yêu cầu tìm kiếm một

biểu diễn cô đọng R=(S,C) cho một đồ thị G cho trước. Biểu diễn này bao gồm một đồ thị tóm tắt S và một tập hợp các hiệu chỉnh C, sao cho đồ thị gốc G có thể được tái tạo lại một cách chính xác từ R.[1] Đây là cốt lõi của "tóm tắt đồ thị không mất mát" (lossless graph summarization).

Các thành phần chính của biểu diễn này bao gồm:

- **Đồ thị tóm tắt (S):** Được xây dựng trên một tập hợp các "siêu nút" (super-nodes). Mỗi siêu nút $u \in P$ (trong đó P là một phân hoạch của tập nút V của G) đại diện cho một tập hợp các nút Pu trong G có cấu trúc lân cận tương tự nhau. Một "siêu cạnh" (super-edge) (u,v) trong S có nghĩa là tất cả các cặp nút trong tích Descartes $Pu \times Pv$ đều được liên kết.[1]
- **Hiệu chỉnh cạnh (C):** Lưu trữ danh sách các cạnh cần được thêm vào (ký hiệu là '+e') hoặc loại bỏ (ký hiệu là '-e') khi tái tạo G từ đồ thị tóm tắt. Các hiệu chỉnh này đảm bảo rằng quá trình tái tạo là hoàn toàn không mất mát thông tin.[1]

Mục tiêu của bài toán là tối thiểu hóa chi phí biểu diễn c(R), được định nghĩa là tổng số siêu cạnh trong S và số lượng hiệu chỉnh trong C, tức là $c(R) = |ES| + |C|$, trong đó ES là tập các siêu cạnh trong S.[1]

Khía cạnh "không mất mát" là một yêu cầu cực kỳ quan trọng trong nhiều ứng dụng, nơi mà việc bảo toàn tuyệt đối thông tin của đồ thị gốc là điều bắt buộc. Tập hợp hiệu chỉnh C chính là cơ chế đảm bảo tính không mất mát này. Tuy nhiên, bản thân C cũng đóng góp vào kích thước tổng thể của bản tóm tắt. Do đó, một thuật toán tóm tắt hiệu quả phải đưa ra quyết định thông minh về việc nhóm các nút nào vào siêu nút và tạo ra các siêu cạnh nào để có thể tối thiểu hóa cả số lượng siêu cạnh $|ES|$ và số lượng hiệu chỉnh $|C|$. Đây là một sự cân bằng tinh tế: nếu đồ thị tóm tắt S quá chi tiết (ít nhóm nút), số lượng siêu cạnh có thể lớn; ngược lại, nếu S quá thô (nhóm nhiều nút khác biệt vào cùng một siêu nút), số lượng hiệu chỉnh C sẽ tăng lên để bù đắp cho sự mất mát thông tin trong S. Việc tìm ra điểm cân bằng tối ưu này chính là thách thức trung tâm của bài toán.

1.3. Sự Đánh đổi Cốt lõi: Độ cô đọng (Compactness) vs. Hiệu quả (Efficiency)

Một trong những thách thức cơ bản và dai dẳng nhất trong lĩnh vực tóm tắt đồ thị là sự đánh đổi giữa chất lượng của bản tóm tắt (độ cô đọng) và thời gian cần thiết để tính toán ra bản tóm tắt đó (hiệu quả). Bài báo nêu rõ: "Mặc dù vấn đề này đã được nghiên cứu rộng rãi, các công trình hiện tại hoặc đánh đổi độ cô đọng của bản tóm tắt để lấy hiệu quả, hoặc ngược lại".[1]

Phân tích các giải pháp hiện có cho thấy rõ sự đánh đổi này:

- **Phương pháp Greedy (Navlakha et al.):** Phương pháp này nổi tiếng với việc cung cấp bản tóm tắt có độ cô đọng cao nhất ("most compact summary"). Tuy nhiên, nó phải trả giá bằng chi phí thời gian tính toán cực kỳ lớn ("prohibitive time cost"), khiến nó không thực tế đối với các đồ thị quy mô lớn.[1] Độ phức tạp thời gian của Greedy là
  $O(n \cdot davg3 \cdot (davg+\log m))$, trong đó n và m lần lượt là số nút và số cạnh, còn davg là bậc trung bình của đồ thị.[1] Một ví dụ điển hình là thuật toán này không thể hoàn thành trong vòng hai ngày trên đồ thị Amazon0601 chỉ với 3 triệu cạnh.[1]
- **Phương pháp SWeG (Shin et al.) và các thuật toán theo sau (ví dụ: LDME, Slugger):** Các thuật toán này được phát triển để giải quyết vấn đề hiệu quả của Greedy. Chúng có chi phí tính toán thực tế ("practical overheads") và có khả năng mở rộng đến các đồ thị hàng tỷ cạnh ("scale to billion-scale graphs"). Tuy nhiên, sự cải thiện về hiệu quả này thường đi kèm với sự suy giảm về độ cô đọng của bản tóm tắt. Ví dụ, các thuật toán này thường kém hơn Greedy về độ cô đọng, với bản tóm tắt lớn hơn 20% so với Greedy.[1] Cụ thể hơn, thực nghiệm cho thấy Greedy tạo ra bản tóm tắt nhỏ hơn trung bình 21.7% so với LDME và 30.2% so với Slugger trên các đồ thị mà Greedy có thể xử lý.[1]

Xung đột trung tâm ở đây là việc đạt được độ cô đọng cao thường đòi hỏi các quy trình tìm kiếm vét cạn hoặc các tinh chỉnh lặp đi lặp lại phức tạp (như trong Greedy), vốn rất tốn kém về mặt tính toán. Ngược lại, để đạt được hiệu quả, các thuật toán thường phải sử dụng các phương pháp heuristic, xấp xỉ, hoặc giới hạn không gian tìm kiếm (như trong mô hình chia để trị của SWeG), điều này có thể dẫn đến các bản tóm tắt không tối ưu về độ cô đọng. Bài báo này đặt mục tiêu phá vỡ sự phân đôi này.

Tình trạng này tạo ra một tình thế tiến thoái lưỡng nan cho các nhà thực hành: họ phải lựa chọn giữa một bản tóm tắt chất lượng cao nhưng mất quá nhiều thời gian để tính toán trên các đồ thị lớn, hoặc một bản tóm tắt được tạo ra nhanh chóng nhưng lại không cô đọng bằng. Điều này hạn chế đáng kể khả năng ứng dụng thực tế của tóm tắt đồ thị trên các "mạng lưới có kích thước đáng kể" (sizable networks).[1] Do đó, một giải pháp có thể dung hòa được cả hai yếu tố – độ cô đọng và hiệu quả – sẽ mang lại giá trị to lớn và mở rộng phạm vi ứng dụng của tóm tắt đồ thị.

Phần 2: Các Giải pháp Đột phá: Thuật toán Mags và Mags-DM

Để giải quyết sự đánh đổi cố hữu giữa độ cô đọng và hiệu quả, bài báo giới thiệu hai thuật toán mới: Mags và Mags-DM. Cả hai thuật toán này đều nhằm mục đích " bắc cầu giữa độ cô đọng và hiệu quả trong tóm tắt đồ thị".[1]

2.1. Thuật toán Mags: Tiếp cận Tham lam Cải tiến

Mục tiêu chính của Mags là "áp dụng mô hình tham lam hiện có vốn cung cấp độ cô đọng hàng đầu, nhưng cải thiện đáng kể hiệu quả của nó bằng một thiết kế thuật toán mới".[1]

Ý tưởng cốt lõi của Mags là khắc phục những điểm yếu về hiệu quả của thuật toán Greedy truyền thống do Navlakha et al. đề xuất.[1] Greedy truyền thống trở nên tốn kém do phải tính toán giá trị "saving" (mức độ tiết kiệm chi phí biểu diễn khi hợp nhất hai siêu nút) cho mọi cặp nút cách nhau 2-hop và cập nhật liên tục các giá trị saving này sau mỗi lần hợp nhất siêu nút.[1] Mags giải quyết vấn đề này bằng cách giảm thiểu đáng kể các tính toán và cập nhật không cần thiết. Thay vì xem xét tất cả các cặp nút 2-hop, Mags tập trung vào một tập hợp các "cặp ứng cử viên" (candidate pairs) có tiềm năng cao.[1] Đồng thời, Mags giới thiệu một thiết kế mới cho giai đoạn hợp nhất tham lam để giảm cả số lần cập nhật saving và số lượng cặp cần được cập nhật trong mỗi lần.[1]

Thuật toán Mags (Algorithm 1) bao gồm ba giai đoạn chính [1]:

- 2.1.1. Giai đoạn Tạo Ứng cử viên (Candidate Generation - Algorithm 2)
  Mục tiêu của giai đoạn này là tạo ra tối đa k·n cặp ứng cử viên, trong đó mỗi cặp được kỳ vọng sẽ mang lại giá trị "saving" cao khi được hợp nhất.1 Thách thức chính là làm thế nào để chọn lọc các cặp ứng cử viên này một cách hiệu quả từ không gian tìm kiếm rất lớn của tất cả các cặp nút cách nhau 2-hop.
  Để giải quyết thách thức này, Mags sử dụng các chiến lược cải tiến sau [1]:
  - **Lấy mẫu hàng xóm 2-hop (2Hop sampling):** Thay vì duyệt toàn bộ các nút trong vùng lân cận 2-hop của một nút u, Mags lấy mẫu một tập con gồm b hàng xóm của u (ký hiệu là S). Sau đó, tập 2Hop được tạo thành bằng cách hợp nhất tập hợp hàng xóm của u (Nu) và tập hợp hàng xóm của mỗi nút trong S. Cách tiếp cận này giúp giảm số lượng tập hợp cần thực hiện phép hợp từ $|Nu|+1$ xuống chỉ còn b+1, qua đó tiết kiệm thời gian tính toán.[1]
  - **Sử dụng MinHash để ước tính Jaccard Similarity:** Sau khi có được tập 2Hop, Mags sử dụng một độ đo dựa trên MinHash, ký hiệu là mh(u,v) (Phương trình 5), để chọn ra k nút từ 2Hop có độ tương đồng cao nhất với u. mh(u,v) là xác suất thực nghiệm mà u và v có cùng giá trị MinHash của tập hợp hàng xóm của chúng, tính trên h hàm băm khác nhau. Giá trị này là một ước lượng không chệch (unbiased estimator) của độ tương tự Jaccard (Jaccard similarity) giữa các tập lân cận Nu và Nv. Độ tương tự Jaccard lại có mối tương quan cao với giá trị "saving", do đó việc chọn các cặp dựa trên mh(u,v) giúp tập trung vào các ứng cử viên tiềm năng.[1]

Bản thân giai đoạn tạo ứng cử viên trong Mags đã thể hiện một sự đánh đổi có chủ ý: nó sử dụng MinHash (một phép xấp xỉ Jaccard similarity, vốn cũng là một phép xấp xỉ của "saving") và kỹ thuật lấy mẫu để tìm kiếm các ứng cử viên *một cách hiệu quả*. Đây là sự hy sinh tính toàn diện (duyệt tất cả các cặp) để đổi lấy tốc độ, với giả định rằng các ứng cử viên chất lượng cao cho giai đoạn hợp nhất tham lam vẫn có thể được tìm thấy. Hiệu quả của chiến lược này phụ thuộc vào mức độ tương quan giữa MinHash, Jaccard và giá trị "saving" thực tế, cũng như khả năng của việc lấy mẫu trong việc giữ lại các cặp tốt (được chứng minh qua Theorem 1 với xác suất cao).[1]Với các tham số b (số hàng xóm được lấy mẫu), h (số hàm băm MinHash) được chọn là các hằng số nhỏ và k (số ứng cử viên cho mỗi nút) được đặt là c·davg (với c là hằng số), giai đoạn tạo ứng cử viên có độ phức tạp thời gian là O(m·logdavg) (Theorem 2).[1]

- 2.1.2. Giai đoạn Hợp nhất Tham lam (Greedy Merge - Algorithm 3)
  Sau khi có được tập hợp các cặp ứng cử viên, giai đoạn hợp nhất tham lam của Mags sẽ lặp lại T lần. Trong mỗi lần lặp thứ t:
  - Các cặp ứng cử viên (u,v) có giá trị "saving" s(u,v) lớn hơn một ngưỡng ω(t) (được định nghĩa trong Phương trình 6) sẽ được xem xét để hợp nhất. Ngưỡng ω(t) này giảm dần khi t tăng, nghĩa là các cặp có "saving" cao sẽ được ưu tiên hợp nhất ở các vòng lặp đầu.[1]
  - Sau khi một loạt các cặp được hợp nhất, giá trị "saving" của các cặp ứng cử viên khác bị ảnh hưởng bởi những lần hợp nhất này sẽ được cập nhật lại.[1]

  Đây là một cải tiến quan trọng so với Greedy truyền thống. Trong khi Greedy phải cập nhật "saving" cho một số lượng rất lớn các cặp (lên đến n×davg2 cặp) sau *mỗi* lần hợp nhất (tối đa n lần hợp nhất), Mags chỉ cập nhật "saving" cho các cặp ứng cử viên (tối đa n×k cặp) trong *mỗi vòng lặp* (tổng cộng T vòng lặp).[1] Ngưỡng hợp nhấtω(t) được thiết kế cẩn thận, bắt đầu từ 0.5 (khi hai nút có tập lân cận giống hệt nhau) và giảm dần theo một chuỗi hình học đến một giá trị nhỏ (ví dụ 0.005) ở vòng lặp cuối cùng T. Điều này đảm bảo rằng các cặp có "saving" cao được hợp nhất trước, và quá trình hợp nhất diễn ra một cách có kiểm soát.[1]Tính chất lặp (T vòng lặp) và ngưỡng hợp nhất giảm dần ω(t) trong giai đoạn này thể hiện một cách tiếp cận "tham lam có kiểm soát". Khác với Greedy cổ điển luôn chọn cặp tốt nhất tuyệt đối trên toàn cục ở mỗi bước, Mags xử lý các lần hợp nhất theo lô trong mỗi vòng lặp dựa trên một ngưỡng. Việc xử lý theo lô và dựa trên ngưỡng này có khả năng đóng góp vào hiệu quả và khả năng song song hóa, trong khi T vòng lặp cho phép tinh chỉnh dần bản tóm tắt.Với k=c·davg, giai đoạn này có độ phức tạp thời gian là O(T·m·(davg+logm)) (Theorem 3).[1]
- 2.1.3. Giai đoạn Xuất Kết quả (Output - Algorithm 4)
  Cuối cùng, từ tập hợp các siêu nút P thu được sau T vòng lặp hợp nhất tham lam, giai đoạn xuất kết quả sẽ quyết định các siêu cạnh và các hiệu chỉnh tối ưu. Quy tắc được áp dụng là: đối với mỗi cặp siêu nút (u,v), nếu số cạnh thực tế giữa các nút gốc trong Pu và Pv, ký hiệu là $|Euv|$, lớn hơn $(|\Pi uv|+1)/2$ (trong đó $\Pi uv=Pu\times Pv$), thì một siêu cạnh (u,v) sẽ được thêm vào đồ thị tóm tắt S, và các

hiệu chỉnh âm ('-e') sẽ được thêm vào C cho các cạnh không tồn tại trong Euv nhưng được bao hàm bởi siêu cạnh. Ngược lại, nếu |Euv| nhỏ hơn hoặc bằng ngưỡng đó, không có siêu cạnh nào được thêm, và các hiệu chỉnh dương ('+e') sẽ được thêm vào C cho mỗi cạnh thực tế trong Euv.1 Giai đoạn này có độ phức tạp thời gian là
O(m) (Theorem 4).[1]

- Tổng thể Mags:
  Kết hợp độ phức tạp của ba giai đoạn, thuật toán Mags có độ phức tạp thời gian tổng thể là O(T·m·(davg+logm)).1 Đóng góp chính của Mags là cải thiện đáng kể độ phức tạp lý thuyết và hiệu quả thực tế so với thuật toán Greedy truyền thống, trong khi vẫn duy trì được độ cô đọng của bản tóm tắt ở mức tương đương.1

2.2. Thuật toán Mags-DM: Tối ưu hóa Phương pháp Chia để trị

Mục tiêu chính của Mags-DM là "theo một mô hình khác với hiệu quả thực tế và khắc phục những hạn chế của nó về độ cô đọng".[1] Mags-DM được xây dựng dựa trên mô hình chia để trị của SWeG nhưng nhằm cải thiện hiệu suất của nó.[1]

Nguyên lý hoạt động của Mags-DM (Algorithm 5) cũng bao gồm T vòng lặp. Trong mỗi vòng lặp thứ t [1]:

- **Giai đoạn Chia (Dividing):** Các nút (hoặc siêu nút từ các vòng lặp trước) trong P được chia thành các nhóm rời rạc S(1),...,S(d) dựa trên giá trị MinHash của chúng, tính toán từ một tập hợp h hàm băm. Một điểm cải tiến quan trọng là chiến lược chia này đảm bảo rằng kích thước của mỗi nhóm không quá lớn. Cụ thể, thuật toán sử dụng một hoán vị ngẫu nhiên của các hàm băm. Nó nhóm các nút theo giá trị MinHash của hàm băm đầu tiên. Nếu một nhóm nào đó có kích thước lớn hơn một hằng số M (ví dụ M=500), nhóm đó sẽ được chia nhỏ hơn nữa một cách đệ quy bằng các hàm băm tiếp theo trong hoán vị, cho đến khi kích thước mỗi nhóm con nhỏ hơn M hoặc đạt đến một độ sâu đệ quy giới hạn (ví dụ 10).[1]
- **Giai đoạn Hợp nhất (Merging):** Thuật toán xử lý từng nhóm S(i) một cách độc lập. Trong mỗi nhóm, nó lặp đi lặp lại việc chọn một nút u ngẫu nhiên. Sau đó, thay vì chỉ tìm một nút tương tự nhất, Mags-DM tìm b nút w∈S(i) có giá trị mh(u,w) (ước lượng Jaccard similarity dựa trên MinHash, Phương trình 5) cao nhất. Từ b nút này (tập Q), nó chọn ra nút v sao cho giá trị "saving" s(u,v) là lớn nhất. Cuối cùng, cặp (u,v) được hợp nhất nếu s(u,v) lớn hơn ngưỡng hợp nhất ω(t) (Phương trình 6, giống như trong Mags).[1] Khi hai siêu nút
  u và v được hợp nhất thành w, giá trị MinHash của w cho mỗi hàm băm được cập nhật bằng minfmin(u),fmin(v).[1]

Mags-DM giới thiệu bốn chiến lược cải tiến đột phá so với SWeG để nâng cao độ

chính xác và hiệu quả của các phép đo xấp xỉ được sử dụng trong quá trình chia và hợp nhất [1]:

- 2.2.1. Chiến lược Hợp nhất 1 (Lựa chọn nút - Node Selection):
  Trong khi SWeG chỉ chọn một nút v duy nhất được cho là tương tự nhất với u để xem xét hợp nhất, Mags-DM có một quy trình lựa chọn tinh vi hơn. Nó chọn ra b nút có độ tương đồng mh(·) cao nhất với u, sau đó tính toán giá trị "saving" s(u,v) thực tế cho từng nút v trong số b ứng cử viên này và chọn nút mang lại "saving" lớn nhất.1 Cách tiếp cận này tăng khả năng tìm được cặp hợp nhất thực sự tốt bằng cách đánh giá một tập nhỏ các ứng cử viên chất lượng cao thay vì chỉ dựa vào một lựa chọn duy nhất dựa trên độ tương đồng xấp xỉ.1
- 2.2.2. Chiến lược Hợp nhất 2 (Đo lường độ tương đồng - Similarity Measure):
  SWeG sử dụng Super-Jaccard, một phiên bản có trọng số của Jaccard similarity. Mags-DM thay thế nó bằng mh(u,v) (Phương trình 5), là một ước lượng không chệch của Jaccard similarity (không có trọng số) giữa các tập lân cận của siêu nút.1 Bài báo chỉ ra rằng Super-Jaccard có thể bị thiên vị, ưu tiên các siêu nút chứa nhiều nút hơn, ngay cả khi cấu trúc kết nối của chúng không thực sự tương đồng bằng các siêu nút nhỏ hơn. Ví dụ được đưa ra trong Hình 3 của bài báo minh họa điều này: Super-Jaccard có thể ưu tiên hợp nhất {b, c} với {f, g, h} (vì {f, g, h} có 3 nút) thay vì với {a} (chỉ có 1 nút), mặc dù {b, c} và {a} có cùng kiểu kết nối. Ngược lại,
  mh(·) (và Jaccard không trọng số) sẽ ưu tiên hợp nhất {b, c} với {a}, đây là lựa chọn tốt hơn về mặt cấu trúc. Do đó, việc sử dụng mh(·) giúp tạo ra bản tóm tắt cô đọng hơn và cũng có thể tính toán nhanh hơn.[1]
- 2.2.3. Chiến lược Hợp nhất 3 (Ngưỡng hợp nhất - Merge Threshold):
  SWeG sử dụng ngưỡng hợp nhất $\theta(t)=1/(1+t)$, giảm khá nhanh khi t tăng.
  Mags-DM áp dụng cùng ngưỡng hợp nhất ω(t) như trong Mags (Phương trình 6). Ngưỡng này bắt đầu từ 0.5 và giảm dần chậm hơn, đặc biệt là ở các vòng lặp đầu.1 Việc giảm ngưỡng chậm hơn cho phép thuật toán có nhiều cơ hội hơn để hợp nhất các cặp thực sự "hứa hẹn" (có "saving" cao) trước khi xem xét các cặp ít hứa hẹn hơn. Điều này dẫn đến một thứ tự hợp nhất tốt hơn và cuối cùng là một bản tóm tắt cô đọng hơn so với SWeG.1
- 2.2.4. Chiến lược Chia (Dividing Strategy):
  SWeG chia các nút chỉ bằng một hàm băm duy nhất trong mỗi vòng lặp. Mags-DM sử dụng một tập hợp các hàm băm và một chiến lược chia đệ quy để đảm bảo rằng kích thước tối đa của mỗi nhóm không quá lớn (ví dụ, không vượt quá hằng số M=500), đồng thời giới hạn độ sâu của việc chia đệ quy.1 Chiến lược này cải thiện đáng kể hiệu quả của thuật toán trên các đồ thị lớn bằng cách tạo ra các nhóm nhỏ hơn và dễ quản lý hơn, từ đó tăng tốc độ tính toán trong giai đoạn hợp

nhất, vì việc tìm kiếm các cặp hợp nhất tiềm năng chỉ diễn ra trong phạm vi các nhóm nhỏ này.1

Bốn chiến lược này trong Mags-DM không chỉ là những điều chỉnh độc lập mà chúng phối hợp với nhau một cách hiệp đồng. Chiến lược Chia cải tiến tạo ra các nhóm tinh tế hơn, dễ quản lý hơn. Trong các nhóm này, Độ đo Tương đồng chính xác hơn (mh(·)) và Lựa chọn Nút tốt hơn (chọn b ứng viên hàng đầu rồi chọn nút có "saving" tốt nhất) giúp xác định các ứng cử viên hợp nhất chất lượng cao hơn. Cuối cùng, Ngưỡng Hợp nhất phù hợp hơn (ω(t)) đảm bảo rằng các lần hợp nhất chất lượng cao này được ưu tiên. Cách tiếp cận tối ưu hóa đa hướng này là chìa khóa cho sự cải thiện về độ cô đọng của Mags-DM so với SWeG.

- Tổng thể Mags-DM:
  Với các hằng số b và h nhỏ, thuật toán Mags-DM có độ phức tạp thời gian là O(T·m) (Theorem 5).1 Đóng góp chính của Mags-DM là cải thiện đáng kể cả về độ cô đọng của bản tóm tắt lẫn hiệu quả thực tế so với các phương pháp chia để trị hiện có như SWeG.1

Phần 3: Đánh giá Thực nghiệm và Hiệu quả

Phần này trình bày chi tiết các kết quả thực nghiệm nhằm đánh giá hiệu suất của Mags và Mags-DM, so sánh chúng với các thuật toán hàng đầu khác và kiểm chứng hiệu quả của các kỹ thuật được đề xuất.

3.1. Thiết lập Môi trường Thực nghiệm

Các thực nghiệm được tiến hành một cách cẩn trọng để đảm bảo tính khách quan và độ tin cậy của kết quả [1]:

- **Tập dữ liệu:** Sử dụng 18 tập dữ liệu đồ thị thực tế, được liệt kê trong Bảng 2 của bài báo.[1] Các tập dữ liệu này rất đa dạng, bao gồm các loại đồ thị như mạng Internet, mạng E-Mail, mạng xã hội địa lý, mạng xã hội thông thường, mạng đồng tác giả, mạng đồng mua hàng, đồ thị web và mạng lưới hợp tác. Kích thước của chúng cũng phong phú, từ vài chục ngàn cạnh đến hơn một tỷ cạnh (ví dụ, tập dữ liệu IT). Tất cả các cạnh có hướng, cạnh trùng lặp và vòng lặp tự thân đều được loại bỏ để chuẩn hóa dữ liệu.
- **Thuật toán so sánh:** Hiệu suất của Mags và Mags-DM được so sánh với ba thuật toán tiêu biểu:
  - **Greedy (Navlakha et al.):** Đại diện cho các phương pháp cho độ cô đọng cao nhất nhưng hiệu quả thấp.
  - **LDME (Yong et al.) và Slugger (Lee et al.):** Đại diện cho các phương pháp

chia để trị có hiệu quả cao và khả năng mở rộng tốt, được coi là các thuật toán hàng đầu (state-of-the-art - SOTA) về mặt hiệu quả.

- **Tham số:** Số lần lặp T được đặt là 50 cho Mags, Mags-DM, LDME và Slugger. Các tham số cụ thể khác, chẳng hạn như k=5 (độ dài chữ ký) cho LDME, cũng được tuân theo như đề xuất của các tác giả gốc hoặc được xác định qua thực nghiệm. Mỗi thí nghiệm được lặp lại ba lần và kết quả trung bình được báo cáo. Giới hạn thời gian chạy là 24 giờ cho mỗi thử nghiệm.
- **Thước đo độ cô đọng:** Độ cô đọng của một biểu diễn R=(S,C) được đánh giá bằng kích thước tương đối của nó so với tập cạnh gốc E của đồ thị G, tức là (|ES|+|C|)/|E|, trong đó ES là tập siêu cạnh và C là tập hiệu chỉnh. Thước đo này được sử dụng rộng rãi, ngoại trừ Slugger sử dụng một thước đo khác do mô hình tóm tắt đồ thị phân cấp của nó.
- **Môi trường thực nghiệm:** Các thí nghiệm được thực hiện trên một máy chủ mạnh với hai CPU Intel Xeon Gold 6342, 512GB RAM, chạy Ubuntu 22.04. Mags, Mags-DM và Greedy được triển khai bằng C++, sử dụng OpenMP cho tính toán song song (40 luồng). Mã nguồn mở Java của LDME và Slugger được sử dụng và biên dịch với OpenJDK 11.

Thiết lập thực nghiệm này được xem là rất toàn diện. Việc sử dụng một loạt các tập dữ liệu đa dạng về kích thước và loại, cùng với việc so sánh với các phương pháp SOTA đã được công nhận, mang lại độ tin cậy cao cho các kết quả và kết luận được rút ra. Việc chia các đồ thị thành nhóm "nhỏ" và "lớn" để so sánh với Greedy là một chiến lược thực tế và công bằng, do chi phí tính toán cao của Greedy.

3.2. So sánh Hiệu suất Tổng thể

Kết quả thực nghiệm cho thấy Mags và Mags-DM đã thành công trong việc dung hòa giữa độ cô đọng và hiệu quả.[1]

- **Trên các đồ thị nhỏ (CA-DB, nơi Greedy có thể chạy trong 24 giờ):**
  - Độ cô đọng (Hình 4[1]):
    Greedy vẫn cho kết quả cô đọng nhất. Tuy nhiên, Mags đạt được độ cô đọng rất gần với Greedy, với sự khác biệt trung bình về kích thước tương đối nhỏ hơn 0.1%. Mags-DM cũng cho kết quả tốt, chỉ kém Greedy khoảng 2.1%. Cả Mags và Mags-DM đều vượt trội đáng kể so với LDME (tóm tắt lớn hơn Greedy 21.7%) và Slugger (lớn hơn Greedy 30.2%).
  - Thời gian chạy (Hình 6[1]):
    Mags nhanh hơn Greedy từ 2 đến 4 bậc độ lớn (orders of magnitude). So với các thuật toán nhanh khác, Mags nhanh hơn LDME trung bình 3.88 lần và nhanh hơn Slugger 3.84 lần. Mags-DM còn ấn tượng hơn nữa khi nhanh hơn

Mags trung bình 7.22 lần.

- **Trên các đồ thị lớn (AM-IT, nơi Greedy không thể hoàn thành):**
  - Độ cô đọng (Hình 5[1]):
    Mags thể hiện sự vượt trội, tạo ra các bản tóm tắt nhỏ hơn trung bình 24.9% so với LDME và 16.9% so với Slugger. Độ cô đọng của Mags-DM rất gần với Mags, chỉ lớn hơn khoảng 2.8%. Một trường hợp ngoại lệ là tập dữ liệu HO, nơi Slugger cho kết quả cô đọng hơn Mags, điều này được giải thích là do cấu trúc phân cấp đặc biệt của HO (chứa một clique lớn) phù hợp với mô hình của Slugger.
  - Thời gian chạy (Hình 7[1]):
    Mags tiếp tục cho thấy hiệu quả cao, nhanh hơn LDME trung bình 15.4 lần và nhanh hơn Slugger 4.4 lần. Mags-DM tiếp tục là thuật toán nhanh nhất, vượt qua Mags trung bình 16.4 lần. Slugger không thể hoàn thành trên hai tập dữ liệu lớn nhất là UK và IT trong giới hạn 24 giờ.
- Kết luận chung về hiệu suất:
  Trên toàn bộ các tập dữ liệu, Mags và Mags-DM đã chứng minh khả năng đạt được hiệu suất SOTA ở cả hai khía cạnh quan trọng: độ cô đọng và hiệu quả. Trong khi các thuật toán trước đây thường chỉ mạnh ở một trong hai khía cạnh, các thuật toán được đề xuất trong bài báo này đã thành công trong việc kết hợp cả hai. Mags đạt được độ cô đọng gần như Greedy nhưng với tốc độ cải thiện vượt bậc. Mags-DM mang lại tốc độ thậm chí còn lớn hơn, với độ cô đọng vẫn tốt hơn đáng kể so với các phương pháp nhanh trước đó (LDME, Slugger) và rất gần với Mags/Greedy. Đây là một bước tiến đáng kể, cho thấy có thể đạt được cả hai mục tiêu mà không cần phải hy sinh quá nhiều một trong hai.
  Để minh họa rõ hơn, bảng dưới đây tóm tắt hiệu suất so sánh trên một số tập dữ liệu đại diện, với dữ liệu được trích xuất từ các kết quả thực nghiệm của bài báo [1]:
  Bảng 1: Tóm tắt So sánh Hiệu suất trên Tập dữ liệu Đại diện
Thuật toán	Tập dữ liệu	Kích thước tương đối (Độ cô đọng)	Thời gian chạy (giây)
Greedy	DB	~0.48	~1000
Mags	DB	~0.48	~10
Mags-DM	DB	~0.49	~1
LDME	DB	~0.61	~40
Slugger	DB	~0.68	~40
Mags	YT	~0.68	~10
Mags-DM	YT	~0.70	~1
LDME	YT	~0.73	~100
Slugger	YT	~0.81	~40

Mags	UK	~0.07	~100
Mags-DM	UK	~0.09	~10
LDME	UK	~0.30	~1000
Slugger	UK	Không hoàn thành	Không hoàn thành
Bảng này cung cấp một cái nhìn tổng hợp, dễ so sánh về hiệu suất của các thuật toán trên các loại đồ thị khác nhau, trực quan hóa sự đánh đổi và ưu điểm của Mags/Mags-DM. Nó tập hợp dữ liệu từ nhiều biểu đồ vào một định dạng cô đọng, giúp người đọc nhanh chóng nắm bắt các xu hướng chính.

3.3. Đánh giá Hiệu quả Kỹ thuật của Mags

Để xác minh đóng góp của từng cải tiến kỹ thuật trong Mags, bài báo tiến hành so sánh Mags với một biến thể gọi là "Mags (naive CG)" (sử dụng phương pháp tạo ứng cử viên đơn giản, gần như vét cạn trong phạm vi 2-hop) và với thuật toán Greedy gốc.[1]

- **Cải tiến từ Greedy sang Mags (naive CG):** Việc thay đổi thiết kế thuật toán cốt lõi của Greedy (như cách Mags xử lý việc hợp nhất và cập nhật "saving", nhưng vẫn dùng cách tạo ứng cử viên đơn giản) đã mang lại hiệu quả đáng kể. "Mags (naive CG)" nhanh hơn Greedy từ 2 đến 4 bậc độ lớn về thời gian chạy, trong khi độ cô đọng của bản tóm tắt gần như tương đương, với sự khác biệt trung bình nhỏ hơn 0.5% (Hình 8a, 8c[1]). Điều này cho thấy rằng cấu trúc vòng lặp hợp nhất và cách quản lý cập nhật "saving" mới của Mags (Phần 3.2 trong bài báo gốc) tự nó đã là một cải tiến lớn, ngay cả khi không có giai đoạn tạo ứng cử viên tinh vi.
- **Cải tiến từ Mags (naive CG) sang Mags (với Candidate Generation cải tiến):** Khi áp dụng giai đoạn Tạo Ứng cử viên tiên tiến của Mags (sử dụng MinHash và lấy mẫu 2Hop, như mô tả ở Phần 3.1 trong bài báo gốc), hiệu quả còn được cải thiện hơn nữa. Giai đoạn Candidate Generation cải tiến này nhanh hơn trung bình 3.68 lần so với phương pháp "naive CG" trong việc tạo ra các cặp ứng cử viên, mà không làm suy giảm độ cô đọng của bản tóm tắt cuối cùng (sự khác biệt không đáng kể). Điều này dẫn đến việc Mags hoàn chỉnh có tốc độ tổng thể nhanh hơn "Mags (naive CG)" khoảng 2 lần trên tất cả các tập dữ liệu, và tốc độ tăng này còn cao hơn trên các đồ thị lớn hoặc đồ thị có bậc trung bình cao (Hình 8b, 8d[1]).

Nghiên cứu cắt lớp (ablation study) này cho Mags đã biện minh rõ ràng cho hai bước đổi mới chính: thứ nhất, sự thay đổi cấu trúc tổng thể so với Greedy cổ điển (được thể hiện qua "Mags (naive CG)" vẫn tốt hơn nhiều so với Greedy), và thứ hai, giai đoạn tạo ứng cử viên tinh vi. Mỗi bước đều mang lại sự gia tăng hiệu suất đáng kể mà không phải hy sinh chất lượng tóm tắt. Điều này chứng tỏ rằng các lựa chọn thiết kế của Mags là hợp lý và đóng góp vào hiệu suất vượt trội của nó.

3.4. Đánh giá Hiệu quả Kỹ thuật của Mags-DM

Tương tự, hiệu quả của các chiến lược mới trong Mags-DM cũng được đánh giá bằng cách so sánh Mags-DM với các biến thể của nó (loại bỏ từng chiến lược một) và với thuật toán SWeG cơ sở.[1]

- **Mags-DM vs. Mags-DM (no DS - không có Chiến lược Chia cải tiến):** Khi loại bỏ Chiến lược Chia tinh vi của Mags-DM (sử dụng nhiều hàm băm và chia đệ quy để giới hạn kích thước nhóm), biến thể "Mags-DM (no DS)" chạy chậm hơn đáng kể, cụ thể là Mags-DM nhanh hơn trung bình 14.4 lần, trong khi độ cô đọng của bản tóm tắt vẫn tương tự.[1] Điều này khẳng định tầm quan trọng của việc chia đồ thị thành các nhóm nhỏ, dễ quản lý để tăng tốc giai đoạn hợp nhất.
- **Mags-DM vs. Mags-DM (no MS - không có các Chiến lược Hợp nhất cải tiến):** Khi loại bỏ cả ba Chiến lược Hợp nhất mới của Mags-DM (Lựa chọn nút, Đo lường độ tương đồng mh(·), và Ngưỡng hợp nhất ω(t)), biến thể "Mags-DM (no MS)" không chỉ chậm hơn 21.3 lần mà còn tạo ra bản tóm tắt kém cô đọng hơn 12.8%.[1] Phân tích sâu hơn về đóng góp của từng chiến lược hợp nhất cho thấy:
  - **Lựa chọn nút** (chọn b ứng viên rồi chọn nút có saving tốt nhất) giúp tăng độ cô đọng thêm 3.1%.
  - **Đo lường độ tương đồng** (sử dụng mh(·) thay vì Super-Jaccard) là cải tiến quan trọng nhất, giúp tăng độ cô đọng 2.8% và đồng thời tăng hiệu quả lên đến 11.4 lần.
  - **Ngưỡng hợp nhất** (sử dụng ω(t) thay vì θ(t) của SWeG) đóng góp vào việc tăng độ cô đọng thêm 1%.
- **Mags-DM vs. SWeG:** So với SWeG (phiên bản được các tác giả triển khai lại), Mags-DM cho thấy sự vượt trội toàn diện: nhanh hơn trung bình 202 lần và luôn tạo ra bản tóm tắt cô đọng hơn trên mọi tập dữ liệu được thử nghiệm.[1]

Nghiên cứu cắt lớp này đối với Mags-DM chứng minh rằng các chiến lược chia và hợp nhất mới không phải là những điều chỉnh nhỏ nhặt mà có tác động sâu sắc đến cả hiệu quả và độ cô đọng so với cách tiếp cận kiểu SWeG cơ bản. Đặc biệt, tác động tích lũy của các chiến lược này (nhất là việc sử dụng mh(·) và chiến lược chia mới) dẫn đến sự tăng tốc đáng kể so với SWeG. Điều này xác nhận các lựa chọn thiết kế của Mags-DM là đúng đắn và hiệu quả.

3.5. Phân tích Tham số

Độ nhạy của Mags và Mags-DM đối với các tham số chính cũng được khảo sát.[1]

- Số lần lặp T (Hình 11, 12[1]):
  Kết quả cho thấy khi tăng T từ 10 lên 50, kích thước tương đối của bản tóm tắt

giảm không nhiều (trung bình 0.5% đối với Mags và 2% đối với Mags-DM). Tuy nhiên, thời gian chạy lại tăng đáng kể (35% đối với Mags và 37% đối với Mags-DM). Điều này ngụ ý rằng độ cô đọng của bản tóm tắt hội tụ khá nhanh, thường đạt mức tốt với T nhỏ (ví dụ T=20). Việc sử dụng T lớn hơn chỉ cải thiện một chút độ cô đọng nhưng phải trả giá bằng thời gian chạy cao hơn. Các thuật toán này tỏ ra mạnh mẽ (robust) với sự thay đổi của T, một phần là do ngưỡng hợp nhất ω(t) được thiết kế để tự động điều chỉnh tốc độ giảm dựa trên T.

- Số luồng song song p (Hình 13[1]):
  Mags-DM cho thấy khả năng song song hóa tốt hơn nhiều so với Mags. Với 40 luồng, Mags-DM đạt được tốc độ tăng trung bình 12.1 lần so với chạy tuần tự, trong khi Mags chỉ đạt 3.4 lần. Nguyên nhân là Mags gặp phải vấn đề tranh chấp dữ liệu (data race) khi hợp nhất đồng thời một lô các cặp nút, trong khi Mags-DM chia các nút thành các nhóm độc lập, cho phép xử lý song song hiệu quả hơn. Cả hai thuật toán đều cho thấy khả năng song song hóa tốt hơn trên các đồ thị lớn.
- Các tham số khác (b, h, k) (Hình 14-16[1]):
  Các tham số này (số hàng xóm lấy mẫu b, số hàm băm h, số ứng cử viên k cho mỗi nút) có tác động hạn chế đến độ cô đọng của bản tóm tắt cuối cùng (sự khác biệt trung bình dưới 0.5%). Chúng cũng có một phạm vi giá trị khả thi khá rộng, nghĩa là thuật toán không quá nhạy cảm với việc lựa chọn chính xác các giá trị này.

Sự tương đối không nhạy cảm với các tham số như T (sau một ngưỡng nhất định), b, h, và k, cùng với khả năng mở rộng song song tốt của Mags-DM, đóng góp vào tính "thực tế" của các giải pháp được đề xuất. Người dùng không cần phải tốn quá nhiều thời gian để tinh chỉnh nhiều tham số để có được kết quả tốt, và Mags-DM có thể tận dụng hiệu quả các kiến trúc đa lõi hiện đại.

3.6. Xử lý Truy vấn Đồ thị

Ngoài mục tiêu chính là tạo ra các biểu diễn đồ thị cô đọng, bài báo còn khảo sát khả năng sử dụng các bản tóm tắt này để tăng tốc các truy vấn đồ thị thông thường.[1]

- Truy vấn láng giềng (Algorithm 6[1]):
  Cho một nút truy vấn q, việc tìm kiếm tập láng giềng của nó trên đồ thị tóm tắt (bao gồm việc xem xét các siêu nút lân cận và áp dụng các hiệu chỉnh liên quan đến q) có độ phức tạp thời gian dự kiến là $O(1.12 \cdot davg)$, trong đó davg là bậc trung bình của đồ thị gốc.
- Tính toán PageRank (Algorithm 7[1]):
  Một thuật toán để tính PageRank trực tiếp trên đồ thị tóm tắt cũng được đề xuất. Về mặt tiệm cận, thời gian chạy của thuật toán này là $O(T \cdot (|ES| + |C|))$, có thể nhanh hơn so với $O(T \cdot m)$ khi chạy trên đồ thị gốc (do $|ES| + |C| \le m$). Kết quả thực

nghiệm (Bảng 3[1]) cho thấy phương pháp này nhanh hơn trên 8 trong số 18 tập dữ liệu. Đối với các tập dữ liệu còn lại, chi phí thời gian không tốt hơn do các yếu tố như hằng số lớn trong cài đặt hoặc chi phí giải nén ngầm định cho một số thao tác.

Điều này cho thấy rằng các bản tóm tắt không chỉ hữu ích cho việc nén dữ liệu mà còn có tiềm năng tăng tốc các tác vụ phân tích đồ thị sau đó. Tuy nhiên, kết quả PageRank cũng chỉ ra rằng việc tăng tốc này không phải lúc nào cũng được đảm bảo và có thể phụ thuộc vào bản chất của truy vấn cũng như chi phí phát sinh khi làm việc với cấu trúc tóm tắt (bao gồm cả siêu nút và hiệu chỉnh). Các thuật toán truy vấn trên đồ thị tóm tắt cần được thiết kế cẩn thận để có thể vượt trội so với việc truy vấn trực tiếp trên đồ thị gốc.

Phần 4: Kết luận và Định hướng Phát triển Tương lai

4.1. Tóm tắt Đóng góp Chính

Bài báo "Graph Summarization: Compactness Meets Efficiency" đã giới thiệu hai thuật toán mới là Mags và Mags-DM, mang lại những đóng góp quan trọng cho lĩnh vực tóm tắt đồ thị.[1] Đóng góp cốt lõi của công trình này là đã giải quyết một cách hiệu quả sự đánh đổi lâu nay giữa độ cô đọng của bản tóm tắt và hiệu quả tính toán.

- **Mags**, một thuật toán dựa trên phương pháp tham lam, có độ phức tạp thời gian $O(T \cdot m \cdot (davg + \log m))$. Nó đã cải thiện đáng kể hiệu quả so với phương pháp Greedy truyền thống mà không làm suy giảm chất lượng (độ cô đọng) của bản tóm tắt.
- **Mags-DM**, một thuật toán dựa trên mô hình chia để trị, có độ phức tạp thời gian $O(T \cdot m)$. Thuật toán này không chỉ cải thiện độ cô đọng của bản tóm tắt so với các phương pháp chia để trị trước đó như SWeG, mà còn cho thấy hiệu quả thực tế vượt trội.
- Cả Mags và Mags-DM đều được thiết kế để có thể tận dụng lợi thế của môi trường tính toán song song, giúp tăng tốc hơn nữa quá trình xử lý trên các hệ thống đa lõi.
- Các kết quả thực nghiệm sâu rộng trên nhiều tập dữ liệu đồ thị lớn đã chứng minh rằng Mags và Mags-DM đạt được hiệu suất hàng đầu (state-of-the-art) ở cả hai khía cạnh: độ cô đọng và hiệu quả. Điều này đánh dấu một bước tiến quan trọng, bởi các công trình trước đây thường chỉ có thể đạt được hiệu suất SOTA ở một trong hai khía cạnh này.

Kết luận của bài báo rằng "Đây là lần đầu tiên các thuật toán tóm tắt đồ thị được chứng minh là vừa thực tế vừa cung cấp một bản tóm tắt nhỏ gọn" [1] cho thấy một tiềm

năng thay đổi mô hình trong lĩnh vực này. Trước công trình này, cộng đồng nghiên cứu dường như chấp nhận một sự đánh đổi khắc nghiệt. Các thuật toán mới này chứng minh rằng sự đánh đổi đó có thể được giảm thiểu đáng kể, có khả năng mở ra ứng dụng của tóm tắt đồ thị cho một phạm vi rộng lớn hơn các bài toán quy mô lớn, nơi mà cả chất lượng tóm tắt và thời gian tính toán đều là yếu tố then chốt.

4.2. Định hướng Phát triển Tương lai

Dựa trên những thành tựu đã đạt được, bài báo cũng đề xuất một số hướng phát triển tiềm năng trong tương lai để tiếp tục nâng cao và mở rộng khả năng của các phương pháp tóm tắt đồ thị [1]:

- **Mở rộng sang tóm tắt có mất mát (Lossy Summarization):** Nghiên cứu việc mở rộng Mags và Mags-DM để hỗ trợ tóm tắt có mất mát. Trong nhiều trường hợp ứng dụng, việc chấp nhận một mức độ lỗi giới hạn trong biểu diễn có thể cho phép đạt được độ cô đọng cao hơn nữa, điều này đặc biệt hữu ích khi tài nguyên lưu trữ cực kỳ hạn chế hoặc khi tốc độ xử lý là ưu tiên hàng đầu.
- **Ứng dụng cho đồ thị động (Dynamic Graphs):** Một hướng quan trọng khác là nghiên cứu việc mở rộng Mags và Mags-DM cho các đồ thị động – những đồ thị thường xuyên được cập nhật với việc thêm hoặc xóa các nút và cạnh. Việc duy trì hiệu quả một bản tóm tắt chất lượng cao trên các đồ thị biến đổi liên tục là một thách thức lớn nhưng có ý nghĩa thực tiễn cao.

Các hướng phát triển được đề xuất này là những bước tiến tự nhiên và hợp lý. Tóm tắt có mất mát giải quyết các tình huống mà yêu cầu nén cao hơn và một số sai số có thể chấp nhận được. Đồ thị động phản ánh bản chất không ngừng phát triển của dữ liệu trong thế giới thực, đặt ra những thách thức mới cho việc duy trì các bản tóm tắt một cách hiệu quả theo thời gian. Việc giải quyết những vấn đề này sẽ làm cho các mô hình như Mags và Mags-DM trở nên linh hoạt và mạnh mẽ hơn nữa, đáp ứng nhu cầu ngày càng tăng về các kỹ thuật tóm tắt đồ thị tiên tiến và có khả năng thích ứng cao.

Nguồn trích dẫn

1. 2024_SIGMO_graphsum_cr.pdf