MỤC LỤC

[Lời cảm ơn 1](#_Toc378314079)

[MỤC LỤC 2](#_Toc378314080)

[DANH MỤC CÁC CHỮ VIẾT TẮT 4](#_Toc378314081)

[DANH MỤC CÁC BẢNG BIỂU 5](#_Toc378314082)

[DANH MỤC CÁC HÌNH VẼ, ĐỒ THỊ 6](#_Toc378314083)

[MỞ ĐẦU 7](#_Toc378314084)

[Chương 1. Tổng quan 8](#_Toc378314085)

[Chương 2. Cơ sở lý thuyết 13](#_Toc378314086)

[2.1 Đồ thị 13](#_Toc378314087)

[2.1.1 Khái niệm về đồ thị 13](#_Toc378314088)

[2.1.2 Phân hoạch đồ thị 14](#_Toc378314089)

[2.1.3 Xây dựng đồ thị trong phân đoạn ảnh 15](#_Toc378314090)

[2.2 Thuật toán Normalized cuts 16](#_Toc378314091)

[2.2.1 Giải thuật two-way-cut 18](#_Toc378314092)

[2.2.2 Giải thuật k-way cut 20](#_Toc378314093)

[2.3 Thuật toán gom nhóm trong phân đoạn ảnh 22](#_Toc378314094)

[2.3.1 Thuật toán k-Means 23](#_Toc378314095)

[2.3.2 Thuật toán FCM 24](#_Toc378314096)

[2.4 Tính toán song song 25](#_Toc378314097)

[2.4.1 Kiến trúc tính toán song song 28](#_Toc378314098)

[2.4.2 Phân loại tính toán song song 30](#_Toc378314099)

[2.4.3 Bộ nhớ trong tính toán song song 31](#_Toc378314100)

[2.4.4 Lập trình song song 33](#_Toc378314101)

[2.4.4.1 Mô hình lập trình song song 33](#_Toc378314102)

[2.4.4.2 Lập trình song song trên cùng một máy tính đa lõi và đa bộ xử lý 36](#_Toc378314103)

[2.4.4.3 Lập trình song song trên hệ thống cụm nhiều máy 37](#_Toc378314104)

[2.4.4.4 Lập trình song song trên thiết bị đồ họa 38](#_Toc378314105)

[2.4.4.5 Lập trình song song với mô hình lai trên CPU – GPU 41](#_Toc378314106)

[Chương 3. Cải tiến thuật toán Ncuts 43](#_Toc378314107)

[3.1 Xác định số nhóm k cần phân đoạn 43](#_Toc378314108)

[3.2 Gom nhóm bằng FCM 45](#_Toc378314109)

[3.3 Tính ma trận tương đồng trên GPU 45](#_Toc378314110)

[3.4 Tính trị riêng trên GPU 47](#_Toc378314111)

[Chương 4. Kết quả và hướng phát triển 49](#_Toc378314112)

[4.1 Cài đặt thử nghiệm 49](#_Toc378314113)

[4.1.1 Tập dữ liệu 49](#_Toc378314114)

[4.1.2 Môi trường cài đặt 49](#_Toc378314115)

[4.1.3 Kết quả cài đặt thực nghiệm 50](#_Toc378314116)

[4.2 Kết luận và hướng phát triển 56](#_Toc378314117)

[Tài liệu tham khảo 58](#_Toc378314118)

DANH MỤC CÁC CHỮ VIẾT TẮT

Ncuts Normalized cuts

CUDA Compute Unified Device Architecture

CPU Central Processing Unit

GPU Graphisc Processing Unit

SISD Single Instruction Single Data

SIMD Single Instruction Multiple Data

MISD Multiple Instruction Single Data

MIMD Multiple Instruction Multiple Data

SDK Software Development Kit

UMA Uniform Memory Access

NUMA Non-uniform Memory Access

DANH MỤC CÁC BẢNG BIỂU

Bảng 2.1: Phân loại kiến trúc máy tính song song 29

Bảng 4.1: Cấu hình phần cứng sử dụng trong thực nghiệm 49

Bảng 4.2: Một số phần mềm sử dụng 50

Bảng 4.3: Số phân vùng trong ảnh xác định trực quan và dùng lược đồ histogram 50

Bảng 4.4: Thời gian tìm ma trận tương đồng W trên CPU và song song trên GPU 52

Bảng 4.5: Thời gian tìm trị riêng trên CPU và song song trên GPU (với 3 segment)

54

Bảng 4.6: Thời gian tìm trị riêng trên CPU và GPU theo số phân vùng (với ảnh có kích cỡ 128x128) 55

DANH MỤC CÁC HÌNH VẼ, ĐỒ THỊ

Hình 2.1 Đồ thị vô hướng có trọng số 13

Hình 2.2. Cách giải quyết vấn đề của máy tính tuần tự 26

Hình 2.3. Cách thực hiện các phần song song trên máy tính song song 27

Hình 2.4. Kiến trúc CPU đa nhân và kiến trúc GPU 38

Hình 2.5. Kiến trúc bộ nhớ GPU 39

Hình 2.6. Mô hình bộ nhớ GPU trong mô hình lập trình CUDA 41

Hình 2.7. Hệ thống lai CPU+GPU 42

Hình 3.1. Lược đồ histogram minh họa cực trị với khoảng cách tương ứng 44

Hình 3.2. Hình minh họa đồ thị và ma trận tương đồng của ảnh 46

Hình 4.1. Gom nhóm bằng k-means trên tập vector riêng Unxk thành k phân vùng 51

Hình 4.2. Gom nhóm bằng FCM trên tập vector riêng Unxk thành k phân vùng 51

Hình 4.3. Thời gian tính ma trận tương đồng W trên CPU và song song trên GPU 53

Hình 4.4. Thời gian tìm trị riêng trên CPU và GPU theo kích cỡ ảnh (với 3 segment)

54

Hình 4.5. Thời gian tìm trị riêng trên CPU và GPU theo số phân vùng (ảnh 128x128) 56

MỞ ĐẦU

Trong gần thập kỉ qua đã có rất nhiều công trình và nghiên cứu khoa học đóng góp không ngừng trong lĩnh vực thị giác máy tính. Trong đó, phân đoạn ảnh là quá trình tiền xử lý quan trọng trong trong hầu hết các ứng dụng xử lý ảnh cũng như thị giác máy tính. Cho đến thời điểm này, có rất nhiều các phương pháp cũng như kỹ thuật phân đoạn ảnh đã và đang được các nhà nghiên cứu khoa học đóng góp. Trong đó, thuật toán Normalized cuts là thuật toán kinh điển khá phổ biến trong phân đoạn ảnh dựa trên đồ thị. Thuật toán Ncuts được đưa ra năm 2000 nhưng đã được áp dụng thành công và cho kết quả tối ưu cho nhiều ứng dụng xử lý ảnh cũng như các ứng dụng khoa học kĩ thuật. Nhận thấy tính cần thiết của phân đoạn ảnh, chúng tôi chọn nghiên cứu thuật toán kinh điển Normalized cuts trong phân đoạn ảnh và tìm cách tối ưu hóa thuật toán hơn, đặc biệt làm sao có thể áp dụng tốt hơn với hệ thống thiết bị ngày một phát triển hiện đại và tối ưu như hiện nay.

Nội dung đề tài được chia thành 4 chương như sau:

* Chương 1, chúng tôi giới thiệu các vấn đề tổng quan về phân đoạn ảnh trong xử lý ảnh số và những vấn đề chúng tôi tìm hiểu về thuật toán Normalized Cut phân đoạn ảnh.
* Chương 2, chúng tôi trình bày cơ sở lý thuyết có liên quan đến thuật toán Normalized Cut, các vấn đề về lý thuyết đồ thị trong phân đoạn ảnh, các thuật toán gom nhóm trong phân đoạn ảnh, và tiêu chuẩn sử dụng cho việc cực tiểu hóa trong phân vùng ảnh thông qua bài toán tìm trị riêng cho ma trận đặc trưng.
* Chương 3, chúng tôi trình bày một số cải tiến của thuật toán dựa trên những ý tưởng đã trình bày trong chương 1.
* Chương 4, chúng tôi đưa ra một số kết quả thực nghiệm của thuật toán với tiểu chuẩn Ncuts như trình bày ở chương 2, đồng thời đưa ra một số kết quả cài đặt thực nghiệm của chúng tôi và nêu một số đánh giá, nhận xét, hướng phát triển.

# Tổng quan

Trong những năm gần đây, ảnh số ngày càng trở nên phổ biến và phong phú hơn, đặc biệt là có liên quan đến nhiều ứng dụng khoa học kĩ thuật. Ảnh số được xem là một trong những phương tiện quan trọng nhất trong việc truyền tải thông tin trong lĩnh vực thị giác máy tính. Việc hiểu các thông tin từ ảnh giúp thực hiện được nhiều nhiệm vụ trong các ứng dụng khoa học kĩ thuật như xác định các tế bào ung thư trong y khoa, xác định vị trí sân bay từ dữ liệu điều khiển cảm biến. Và mục tiêu cần thiết phải xác định các phân vùng hay các đối tượng trong ảnh để xử lý hay rút trích thông tin ở mức cao là rất quan trọng trong các quá trình xử lý ảnh và thị giác máy tính. Phân đoạn ảnh là quá trình phân chia ảnh thành các vùng không giao nhau hay các đối tượng như tập các điểm ảnh, hoặc vùng gồm các điểm ảnh có tính chất tương đồng nhau theo một tiêu chuẩn đồng nhất nào đó, ví dụ như màu sắc, cường độ, kết cấu, mức xám ... để từ đó xác định vị trí đối tượng và biên trong ảnh [[1](#Raj12)]. Đây là một trong những nhiệm vụ quan trọng nhất trong phân tích ảnh số tự động vì các kết quả của phân đoạn ảnh sẽ có ảnh hưởng cốt yếu đến tất cả các quá trình tiếp theo trong nhiều ứng dụng của xử lý ảnh và thị giác máy tính, như biểu diễn và mô tả đối tượng trong ảnh, xử lý nhận dạng đối tượng, phân loại ảnh, nén ảnh dựa trên đối tượng hay truy vấn ảnh dựa trên nội dung, …. Trong lĩnh vực thị giác máy tính và xử lý ảnh, phân đoạn ảnh là một trong những công đoạn quan trọng và được quan tâm nghiên cứu rất nhiều do tính cần thiết trong nhiều ứng dụng. Quá trình phân đoạn ảnh số còn có nhiều đóng góp quan trọng trong nhiều lĩnh vực như y tế, viễn thám, dự báo thời tiết, các bài toán nhận dạng, các bài toán giao thông, ....

Việc nghiên cứu phân đoạn ảnh trong nhiều năm nay đã có nhiều thành tựu và có rất nhiều kĩ thuật phân đoạn ảnh được đưa ra. Tuy nhiên không có một phương pháp nào là tốt nhất cho các loại ảnh khác nhau, cũng như không phải tất cả các phương pháp là tốt nhất cho một loại ảnh đặc trưng nào. Cho nên việc lựa chọn kĩ thuật phân đoạn ảnh nào đó còn dựa trên tính chất của ảnh và vần đề cần giải quyết. Có nhiều cách để phân loại các kĩ thuật này, theo [[1](#Raj12),[2](#Man13)] các kĩ thuật có thể phân loại thành:

- Kĩ thuật dựa trên phát hiện cạnh/biên: kĩ thuật này dựa trên việc phát hiện các cạnh hay các điểm ảnh giữa các vùng khác nhau về mức độ biến đổi cường độ nhằm thành các biên giữa các vùng hay đối tượng, ví dụ kĩ thuật sử dụng lược đồ xám của anh - gray histogram, kĩ thuật dựa trên biến đổi gradient, Robert edge detector, the Sobel edge detector and the Canny edge detector, ...

- Kĩ thuật phân ngưỡng (Threshodling methods): kĩ thuật này tách biệt giữa nền ảnh và đối tượng bằng việc đặt một giá trị ngưỡng. Những điểm ảnh thuộc về phần tiền cảnh có cường độ cao hơn hoặc bằng giá trị ngưỡng cho trước, và ngược lại là những điểm ảnh thuộc về nền ảnh, ví dụ các kĩ thuật Mean Technique, P-Tile Technique, ...

- Kĩ thuật dựa trên vùng (Region based methods): phân vùng ảnh thành các vùng tương đồng nhau theo một tiêu chí cho trước, chủ yếu gồm các phương pháp: loang vùng, chia và trộn vùng.

- Kĩ thuật phân đoạn ảnh bằng phương trình vi phân (Partial Differential Equation - PDE): phương pháp chủ yếu dựa trên những thay đổi về cường độ hoặc màu sắc của ảnh dựa trên mô hình PDE và giải phương trình PDE

- Kĩ thuật phân đoạn ảnh bằng phương pháp gom nhóm: các điểm ảnh có tính tương đồng sẽ được gom nhóm thành các cụm theo một tiêu chuẩn tương đồng nào đó dựa trên đặc trưng của ảnh như cường độ, kết cấu, màu sắc, ...., và xét thêm quan hệ giữa các điểm ảnh khi phân nhóm và cập nhật giá trị đại diện cho mỗi nhóm. Các kĩ thuật gom nhóm thường dùng như k-Means, FCM, ...

- Kĩ thuật dựa trên phân hoạch đồ thị: ảnh được xem như đồ thị, và nguyên lý chủ yếu của kĩ thuật này là tìm tiêu chuẩn phân hoạch tốt nhất tạo ra mỗi đồ thị con tương ứng với vùng cần phân đoạn. Các kĩ thuật phổ biến đại diện cho phương pháp này như Min-Max Cut, Normalized Cuts.

Trong nhiều thập kỉ qua, việc nghiên cứu phân đoạn ảnh dựa vào lý thuyết đồ thị đã có nhiều kĩ thuật tiến bộ. Đặc biệt, trong những năm gần đây sự xuất hiện nhiều thuật toán mới đã chứng minh rằng phương pháp này là một hướng nghiên cứu đầy hứa hẹn trong cộng đồng nghiên cứu phân đoạn ảnh. Các phương pháp dựa trên lý thuyết đồ thị đã có nhiều ứng dụng quan trọng trong lĩnh vực thị giác máy tính, đặc biệt là các ứng dụng trong phân đoạn ảnh. Dựa trên các nguyên lí cấu trúc hình thức (gestalt principles) về mức độ tương đồng hoặc lân cận trong việc gom nhóm tri giác, đồ thị được chia thành các vùng tương ứng với một vùng hay đối tượng nào đó trong ảnh. Kết quả phân đoạn của các phương pháp đầu tiên này chỉ sử dụng ngưỡng cố định và tính toán cục bộ, chưa thể hiện được tính chất toàn cục, nên các phân vùng thường chia nhỏ và không như mong muốn. Khoảng những năm 90, phương pháp phân đoạn ảnh bằng đồ thị sử dụng hàm tính toàn cục được đề xuất với kết quả phân đoạn cuối cùng tốt hơn và đáp ứng được yêu cầu của nhiều ứng dụng thị giác. Sau đó, các kĩ thuật tối tưu hóa trên đồ thị cho phân đoạn ảnh ngày càng được quan tâm nghiên cứu. Các phương pháp cơ bản dựa trên các kĩ thuật sau hoặc kết hợp các kĩ thuật với nhau [[3](#BoP13),[4](#PFF04),[5](#SHK97)]:

* Phương pháp tìm cây bao trùm tối tiểu của đồ thị (Minimal Spanning Tree - MST): các thuật toán MST cho phân đoạn ảnh định nghĩa ảnh bằng đồ thị có trọng số theo không gian đặc trưng nào đó, xem việc gom nhóm các điểm ảnh được thực hiện như việc tìm cây bao trùm tối tiểu của đồ thị tương ứng với ảnh đầu vào. Đồ thị được chia thành các đồ thị con bằng cách xóa đi các cạnh sao cho thỏa tổng trọng số nhỏ nhất.
* Phương pháp graph-cut với các hàm giá trị: tương tự với MST cũng định nghĩa dựa trên đồ thị có trọng số và phân vùng đồ thị mang tính toàn cục. Phương pháp graph-cut có thể xem là mẫu chung cho các ứng dụng phân đoạn ảnh bằng kĩ thuật phân vùng đồ thị. Đây là một thuận lợi cho các ứng dụng khác nhau vì có thể định nghĩa các tiêu chuẩn ‘cut’ khác nhau, hay tối ưu hóa các hàm tính toán giá trị toàn cục cho việc phân vùng đồ thị cho ứng dụng riêng biệt nào đó, ví dụ tiêu biểu là thuật toán max-flow/min-cut, mô hình Markov random field. Các phương pháp tiêu biểu cho phương pháp graph-cut như Minimal cut, Normalized cuts, Mean cut, Ratio cut [[6](#Sis01),[7](#Son03)].
* Phương pháp dựa trên tìm đường đi ngắn nhất của đồ thị: biên của đối tượng hay vùng của ảnh được định nghĩa như tập cạnh là đường đi ngắn nhất giữa hai đỉnh của đồ thị bằng các thuật toán tiêu biểu trong lý thuyết đồ thị như Dijsktra. Việc tìm biên của đối tượng hay phân vùng trong ảnh được chuyển thành việc tìm đường đi ngắn nhất giữa cặp đỉnh trong đồ thị. Và trong nhiều ứng dụng đòi hỏi có sự tương tác của người dùng như chọn điểm khởi tạo của biên.

Rõ ràng các phương pháp phân đoạn ảnh sử dụng lý thuyết đồ thị đã mang lại hiệu quả đáng kể trong xử lý ảnh số, hầu hết các kết quả đều thể hiện được các phân vùng theo thị giác con người [[3](#BoP13),[8](#CTZ71)]. Đồ thị được dùng như mô hình toán học mô tả những đặc tính rời rạc của ảnh trong thế giới thực. Hơn nữa, trong lý thuyết đồ thị cũng đã có sẵn nhiều định lý và công trình giải quyết vấn đề gom nhóm, mà nó cũng là vấn đề quan trọng trong phân đoạn ảnh bằng đồ thị. Việc dùng đồ thị để biểu diễn ảnh số cho ta một hướng nghiên cứu hữu hiệu trong phân đoạn ảnh. Tuy nhiên, đa số các thuật toán phân đoạn ảnh dựa trên lý thuyết đồ thị chiếm bộ nhớ và thời gian lớn đối với dữ liệu ảnh đầu vào lớn, điển hình là ảnh y khoa, ảnh thiên văn đòi hỏi phải cho kết quả nhanh và tốt nhất là kết quả tương đối với thời gian thực.

Thuật toán Normalized cuts (Ncuts) mà J.Shi và J.Malik [[9](#Shi00)] đưa ra là thuật toán phân đoạn ảnh kinh điển dựa vào lý thuyết đồ thị khá phổ biến trong cộng đồng xử lý ảnh. J.Shi và J.Malik đã đưa ra một tiêu chuẩn mới cho việc tìm phân vùng tối ưu - tiêu chuẩn *normalized cuts*, không chỉ tập trung trên đặc tính cục bộ trên ảnh mà còn tập trung vào việc phân vùng và gom nhóm dựa trên đặc trưng toàn cục ảnh. Thuật toán Normalized Cuts là thuật toán sử dụng phương pháp phân vùng dựa trên lý thuyết đồ thị. Tập dữ liệu được thể hiện một mối quan hệ tương ứng bằng việc chuyển tập điểm dữ liệu ban đầu thành một đồ thị trọng số vô hướng G = (E,V). Mỗi đỉnh của đồ thị G thể hiện một điểm trong không gian đặc trưng. Mỗi cạnh được hình thành giữa hai đỉnh với trọng số w(i,j) là hàm tính khoảng cách giữa hai đỉnh i và j theo nhiều cách dựa trên đặc trưng của ảnh và ứng dụng.

Normalized Cuts là thuật toán kinh điển khá phổ biến trong lĩnh vực phân đoạn ảnh dựa trên đồ thị. Tuy nhiên thuật toán cũng tồn tại những khuyết điểm của nó.

* Với tập ảnh lớn thuật toán thực thi với tốc độ chậm, mất nhiều thời gian và chiếm nhiều bộ nhớ trong tính toán.
* Vấn đề tìm số phân vùng cần phân vùng có thể chủ yếu còn dựa vào cảm tính trực quan, chưa có cơ sở lý thuyết để áp dụng chung.

Dựa trên những tồn tại trên của thuật toán, chúng tôi đề xuất giải pháp cải tiến thuật toán Normalized Cuts trong phân đoạn ảnh với mong muốn đạt tính hiệu quả về tốc độ hơn cho dữ liệu ảnh có kích cỡ lớn.

* Dữ liệu ảnh vào cần phân đoạn yêu cầu phải xác định số phân vùng k đầu vào cho thuật toán, nhưng với số phân vùng k bao nhiêu là phù hợp với ảnh cần phân đoạn. Nếu với số k phân vùng đầu vào nhiều hơn cần thiết sẽ làm cho kết quả phân vùng quá mịn dẫn đến chi phí tính toán cao nhưng kết quả không tốt. Ngược lại với k phân vùng đầu vào quá ít lại cho kết quả phân vùng thô không như mong muốn. Để giải quyết vấn đề này chúng tôi đề xuất phương pháp tìm số phân vùng (nhóm) ban đầu thay vì chỉ dựa vào trực quan con người theo đặc trưng của mỗi ảnh khác nhau bằng phương pháp histogram. Hơn nữa, số nhóm tìm được này cũng tương đối thích hợp với ảnh thực nhằm giảm chi phí và thời gian cho quá trình phân đoạn sau cùng.
* Như đã trình bày ở trên với dữ liệu ảnh vào có kích thước lớn thì thuật toán Normalized cuts cho kết quả với thời gian thực thi lớn. Trên cơ sở đó, với mục đích cải thiện thời gian xử lý của thuật toán có thể chấp nhận được với sự phát triển của công nghệ như hiện nay, chúng tôi cũng đề xuất phương pháp xử lý song song cho bước tính chiếm nhiều thời gian và bộ nhớ trong thuật toán. Đó là quá trình song song hóa thuật toán tìm trị riêng cho hệ phương trình phi tuyến trên thiết bị đồ họa GPU bằng ngôn ngữ song song CUDA.

# Cơ sở lý thuyết

## Đồ thị

### Khái niệm về đồ thị

Đồ thị là một mô hình cực kì quan trọng trong toán học ứng dụng, được áp dụng rộng rãi và tối ưu trong nhiều ứng dụng kĩ thuật cũng như nghiên cứu khoa học. Một đồ thị G(V, E) bao gồm n đỉnh (node, vertex) và m cạnh (edge, link) nối các đỉnh với nhau [[10](#Oli12)]. Đồ thị có thể được phân loại thành đồ thị có trọng số và đồ thị không trọng số. Với đồ thị có trọng số, một cạnh eij nối hai đỉnh i và j được gán bởi một trọng số wij. Bên cạnh đó, đồ thị còn được phân loại thành đồ thị có hướng và đồ thị vô hướng. Một đồ thị chứa một tập đỉnh với các cạnh nối theo một trật tự có hướng nào đó thì đồ thị đó được gọi là đồ thị có hướng. Trong đề tài của tôi, tôi chỉ tập trung vào đồ thị vô hướng có trọng số. Hình 2.1 là một ví dụ minh họa về đồ thị vô hướng có trọng số.

1

3

6

4

2

7

Đỉnh

2225

7

4

8

2

5

Cạnh

Trọng số

Hình 2.1 Đồ thị vô hướng có trọng số

Có nhiều cách khác nhau để lưu trữ một đồ thị trong máy tính. Đồ thị có thể được biểu diễn bằng ma trận như ma trận liên thuộc, ma trận Laplace và ma trận kề. Để xây dựng ma trận từ một đồ thị thì các đỉnh và các cạnh của đồ thị phải được đánh số. Ma trận liên thuộc **A** là một ma trận gồm 0, 1, và -1 biểu diễn các đỉnh trên cột và các cạnh trên dòng. Trên mỗi dòng, phần tử thứ ij của ma trận liên thuộc sẽ có giá trị là 1 hoặc -1 đại diện cho một cạnh có mối liên hệ giữa đỉnh i và đỉnh j.

Trọng số của cạnh của một đồ thị có thể được lưu trữ trong một ma trận đường chéo **C**. Khi đó, ma trận Laplace **L** có thể được xây dựng bằng phép nhân giữa ma trận liên thuộc **A** và ma trận đường chéo trọng số **C** theo công thức (2.1)

**** (2.1)

Ma trận bậc dường chéo **D** là một ma trận đường chéo của ma trận Laplace **L**. Trong đó những phần tử đường chéo là tổng trọng số của các cạnh nối đến các đỉnh. Ma trận kề **W** bao gồm những phần tử ngoài đường chéo của ma trận Laplace **L**, tương tự ma trận liên thuộc, ma trận kề **W** cũng cho biết thông tin kết nối giữa các đỉnh. Mối liên hệ giữa ma trận bậc **D**, ma trận kề **W**, và ma trận Laplace **L** được biểu diễn theo công thức (2.2)

 (2.2)

Ma trận Laplace **L** là ma trận đơn nửa xác định dương đối xứng. Ma trận này có một trị riêng zero.

### Phân hoạch đồ thị

Phân vùng đồ thị là việc gom nhóm tất cả các đỉnh của đồ thị thành hai hay nhiều vùng dựa trên tiêu chuẩn nhất định nào đó. Các tiêu chuẩn đó có thể là vị trí của các định, dựa trên đặc trưng giá trị của đỉnh (ví dụ như cường độ điểm ảnh trong bài toán phân đoạn ảnh), hoặc dựa trên tính kết nối giữa các đỉnh. Các kĩ thuật graph cut được sử dụng phổ biển để phân vùng đồ thị như Minimum Cut, Normalized Cut và Isoperimetric Partitioning.

Đồ thị được sử dụng để mô hình hóa nhiều bài toán như bài toán mạng giao thông, mạng internet, bài toán lập lịch, bài toán gom nhóm dữ liệu. Kĩ thuật phân vùng đồ thị được áp dụng trong các bài toán gom nhóm dữ liệu, trong nghiên cứu sinh tin học kĩ thuật graph cut sử dụng để gom nhóm dữ liệu microarray. Đồng thời các nhà khoa học máy tính cũng sử dụng phương pháp phân vùng đồ thị trong lĩnh vực thị giác máy tính, mà cụ thể là trong phân đoạn ảnh, giúp máy tính có thể giải quyết bài toán phân đoạn ảnh một phần giống như từ góc nhìn của con người. Hơn nữa, phương pháp phân vùng đồ thị cũng khá quan trọng trong tính toán song song. Dữ liệu hay tập các đỉnh có thể được chia thành các nhóm cân bằng, khi đó các kết nối hay các cạnh giữa các nhóm có thể được cắt giảm tại cùng một thời điểm theo một tiêu chuẩn nào đó. Bằng cách này, chúng ta có thể phân chia công việc như nhau cho nhiều luồng xử lý ở một thời điểm để đạt tính toán nhanh hơn.

### Xây dựng đồ thị trong phân đoạn ảnh

Ảnh ban đầu sẽ được biểu diễn tương ứng bằng một đồ thị vô hướng có trọng số. Tập hợp những điểm ảnh trong không gian đặc trưng bất kì được biển diễn bằng đồ thị trọng số vô hướng G=(V, E). Trong đó, tập đỉnh V của đồ thị là những điểm ảnh hoặc vùng trong không gian Euclide. Mỗi cạnh trong tập cạnh E nối giữa một cặp đỉnh i và j. Mỗi cạnh (vi , vj)  E có trọng số wij thể hiện mối quan hệ hoặc tính chất giữa hai điểm được nối bởi cạnh đó, cũng chính là hàm tính độ tương đồng giữa hai điểm ảnh i và j.

Ảnh được phân chia thành các phần riêng biệt nhau, sao cho mỗi phần A là một đồ thị con liên thông G’ = (V’, E’), với V’V, E’E. Hay nói cách khác, các tập con không rỗng A1, A2, …, Ak là các phân vùng của đồ thị G thỏa AiAj =  và A1…Ak = V. Có nhiều tiêu chuẩn để xác định phân vùng, nhưng nhìn chung các tiêu chuẩn mà các phương pháp đưa ra đều cố gắng sao cho các phần tử trong cùng mỗi phân vùng có tính tương đồng nhau về độ sáng, màu sắc, hoặc kết cấu … và các phần tử trong các phân vùng khác nhau có mức độ không tương đồng cao hơn [[10](#Oli12)].

Trong lý thuyết đồ thị, mức độ không tương đồng giữa hai thành phần hay vùng có thể được tính dưới dạng graph-cut. Mỗi cut là một tập cạnh mà khi xóa đi, đồ thị G sẽ được chia thành hai tập đỉnh phân biệt A và B. Giá trị cut được tính bằng tổng trọng lượng nối giữa hai tập đỉnh đó theo công thức (2.3)

 với A∩B=∅ (2.3)

Trong xử lý ảnh, ảnh nhiễu và tính không rõ ràng của ảnh thực sẽ mang lại sự không chắc chắn hoặc sẽ có nhiều kết quả khác nhau trong phân đoạn ảnh. Như vậy khó mà có giải pháp xác định chính xác phân vùng ảnh đúng như trực quan con người. Các phương pháp dựa trên kĩ thuật tối ưu hóa cố tím cách giải quyết vấn đề cũng như việc tối thiểu hóa các tiêu chí mà mỗi giải pháp đưa ra. Vì vậy, các phương pháp dựa trên tối ưu hóa này mong muốn tìm các giải pháp chính xác hơn hoặc gần đúng với bài toán thị giác. Và trong trường hợp này, việc tìm phân vùng tối ưu của đồ thị được thực hiện như việc tối thiểu hóa giá trị cut trong công thức (2.3).

Wu và Leahy [[11](#ZWu93)] đã đề xuất phương pháp phân cụm dựa trên tiêu chuẩn minimun cut này. Tiêu chuẩn minimum cut là minh họa trực quan cho nguyên lí dựa trên cấu trúc hình thức trên đồ thị. Tuy nhiên, minimum cut cho kết quả phân vùng không tốt trong một số trường hợp. Phương pháp này chia nhỏ đồ thị thành tập nhỏ các đỉnh một cách cục bộ, mà khi xét trên toàn cục thì các tập con này có liên quan nhau và có thể được xét như một nhóm. Phương pháp có xu hướng cho kết quả phân vùng mịn, đặc biệt trong trường hợp dữ liệu có phần tử ngoại lai hoặc rời rạc. Có nhiều nghiên cứu được đề xuất để giải quyết vấn đề graph-cut này với nhiều hàm mục tiêu chuẩn hóa khác nhau liên quan đến lý thuyết gom nhóm đồ thị quang phổ [[11](#ZWu93)]. Normalized Cuts là một trong những thuật toán tiêu biểu trong việc giải quyết vấn đề này.

## Thuật toán Normalized cuts

Thuật toán Normalized Cuts sử dụng lý thuyết đồ thị để tìm nhóm các điểm ảnh trong không gian đặc trưng của ảnh số. Normalized Cuts là thuật toán dùng để phân đoạn một ảnh ra thành những vùng khác nhau có thuộc tính tương tự nhau. Nó còn dùng để gom nhóm các đối tượng trong một đồ thị theo một không gian thuộc tính cho trước nào đó.

Shi và Malik [[9](#Shi00)] đã đề xuất tiêu chuẩn mới – tiêu chuẩn Normalized Cuts (Ncut) để khắc phục trường hợp trên. Thay vì chỉ quan tâm đến giá trị tổng trọng lượng các cạnh giữa hai vùng, tiêu chuẩn normalized cut tính giá trị ‘cut’ theo tỉ số trên tổng trọng lượng kết nối đến tất cả các đỉnh trong đồ thị.

(2.5)

Trong đó, , là tổng trọng lượng của các kết nối từ các đỉnh trong A đến tất cả các đỉnh trong đồ thị. Với tỉ số này, rõ ràng giá trị ‘cut’ sẽ có giá trị phần trăm lớn đối với các ‘cut’ nhỏ và những phân vùng có tập đỉnh nhỏ sẽ không có giá trị Ncut nhỏ. Do đó, tiêu chuẩn Ncut tránh phân vùng mịn như tình trạng minimum cut như trên.

Với hơn hai vùng thì tiêu chuẩn normalized cut được mở rộng theo công thức (2.6)

(2.6)

Giả sử tập đỉnh V là một vùng của đồ thị, cần tách thành hai tập con A, B. Đặt x là vector có N=|V| chiều, với xi=1 nếu đỉnh i ∈ A và ngược lại xi=-1. Đặt  là tổng kết nối từ đỉnh i đến tất cả các đỉnh khác.

Việc cực tiểu hóa tiêu chuẩn Ncut như trên cho đồ thị là bài toán NP-hard. Việc cực tiểu hóa phương trình 2.6 được chuyển thành bài toán giải hệ trị riêng tổng quát. Đây cũng là đề tài nghiên cứu nổi tiếng trong lý thuyết đồ thị quang phổ [[12](#Atu)]. Đặt **W**=(wij) là ma trận đối xứng cỡ N×N và **D** = diag(d1, d2, …, dn) là ma trận đường chéo kích thước N×N. Qua nhiều bước biến đổi toán học, tiêu chuẩn Ncut trở thành vấn đề trong giải bài toán đại số.

 (2.7)

trong đó, y = (1 + x) – b(1 – x) và b = k/ (1 – k), với 

Công thức liên quan đến bài toán giải hệ trị riêng phổ biến trong đại số tuyến tính **A**x = Id**B**x. Với y ∈ Rn, cực tiểu hóa Ncut cũng là giải hệ trị riêng sau (**D**-**W**)y = λ**D**y. Gọi **L** = **D** - **W** là ma trận Laplace của đồ thị G, trong đó **D** và **W** là ma trận đường chéo và ma trận kề tương ứng của đồ thị G. Việc tối ưu hóa công thức 2.7 được nới rộng bằng cách loại các điều kiện rời rạc và cho y nhận các giá trị thực bất kì. Vì **W**, **D** là ma trận rời rạc (sparse matrix) có kích cỡ lớn và có nhiều phần tử 0 nên kĩ thuật Lanczos được áp dụng cho phép tính với những ma trận này.

Theo định lí Rayleigh-Ritz [[9](#Shi00)], vector riêng tương ứng với giá trị riêng nhỏ nhất thứ hai của ma trận Laplace là nghiệm thực cần tìm của phương trình 2.7. Cuối cùng, để phân vùng đồ thị, ta có thể đặt một ngưỡng trên vector riêng này để chia đôi đồ thị và đệ qui theo kiểu two-way-cut để đạt kết quả. Đối với phân vùng theo kiểu k-way-cut, quá trình two-way-cut được thực hiện lặp đi lặp lại trên đồ thị cho tới khi đến một kết quả khả quan. Sau đây là phần trình bày giải thuật gom nhóm theo hai cách: two-way-cut và k-way-cut sử dụng tiêu chuẩn Ncut.

### Giải thuật two-way-cut

Giải thuật sử dụng vector riêng nhỏ thứ hai cho việc phân vùng đồ thị. Vì vector riêng có giá trị liên tục nên ta cần xác định điểm chia (splitting point) cho vector riêng đó. Giá trị 0 hoặc giá trị trung vị (median) có thể được chọn như điểm chia. Tuy nhiên, những giá trị này không chắc chắn vì lỗi xấp xỉ. Một giải pháp khác là phương pháp heuristic tìm kiếm điểm chia. Một số điểm được đánh dấu theo khoảng cách đều nhau trong vector riêng và tính giá trị Ncut cho mỗi điểm. Điểm nào có giá trị Ncut nhỏ nhất sẽ được chọn làm điểm chia. Thuật toán được áp dụng đệ qui trên mỗi phân vùng đồ thị cho đến khi vượt quá một giới hạn Ncut cho trước.

**Thuật toán 2.1:** Thuật toán Ncuts – Giải thuật 2-way cut [[9](#Shi00)]

Thuật toán 2.1: Thuật toán Ncuts – Giải thuật 2-way cut

Bước 1: Ảnh vào I.

* Lập đồ thị vô hướng có trọng số tương ứng cho ảnh G=(V, E), trong đó mỗi đỉnh của G tương ứng với mỗi điểm ảnh của ảnh I.

V = tập đỉnh của G tương ứng với tập điểm ảnh của I. Đặt N là số đỉnh (hay số điểm ảnh), ta có N=|V|.

E = tập cạnh nối giữa hai đỉnh, dựa vào độ tương đồng giữa hai điểm ảnh tương ứng (tương ứng với những đặc trưng của ảnh về độ sáng, màu, hay kết cấu …)

* Xây dựng ma trận tương đồng đối xứng **W** = (wij) cỡ NxN theo công thức (2.8)

(2.8)

Trong đó, X(i) là tọa độ không gian của ảnh gốc I, và F(i) là vector đặc trưng được xác định theo thuật toán 2.2.

* Đặt  là tổng các kết nối từ đỉnh i đến tất cả các đỉnh còn lại. Xây dựng ma trận đường chéo **D** với di là giá trị đường chéo.

Bước 2: Giải hệ trị riêng tổng quát (**D**-**W**)x = λ**D**x và tìm vector riêng có trị riêng nhỏ thứ hai.

Bước 3: Dùng vector riêng này chia đôi đồ thị. Trong trường hợp lý tưởng, vector riêng này chỉ nhận hai giá trị rời rạc, và dựa vào dấu để xác định phân vùng (ví dụ A = {Vi | yi > 0}, và B = {Vi| yi < 0}). Tuy nhiên y nhận những giá trị thực liên tục, vì thế cần chọn điểm chia phù hợp. Có nhiều cách để chọn điểm chia, ví dụ chọn giá trị 0 hoặc giá trị trung vị làm điểm chia. Tuy nhiên như đã nói trên, phương pháp heuristic được áp dụng tối ưu hơn, tìm kiếm điểm chia sao cho giá trị Ncut cực tiểu theo công thức (2.5).

Bước 4: Lặp lại đệ qui chia đôi mỗi phân vùng cho đến khi giá trị Ncut lớn hơn ngưỡng cho trước.

**Thuật toán 2.2**: Tính vector đặc trưng F(i)

Thuật toán 2.2: Tính vector đặc trưng F(i)

- Nếu dữ liệu là tập điểm thì F(i) = 1.

- Nếu dữ liệu được xác định theo giá trị cường độ ảnh cho việc phân đoạn ảnh, ví dụ cho ảnh sáng hay ảnh xám thì *F(i) = I(i).*

- Nếu dữ liệu được xác định cho phân đoạn ảnh màu thì , trong đó h, s, v là giá trị màu HSV.

- Nếu dữ liệu được xác định cho phân đoạn ảnh theo kết cấu thì , trong đó *fi* là hàm lọc DOOG cho phân đoạn ảnh theo kết cấu.

### Giải thuật k-way cut

Tiêu chuẩn cố định của đệ qui two-way-cut đảm bảo được tính dao động của các vector riêng. Tuy nhiên, chỉ vector riêng nhỏ thứ hai được sử dụng và bỏ qua các vector sau đó mà có thể cũng chứa thông tin phân vùng tốt. Thay vì chỉ tìm phân vùng sử dụng đệ qui two-way-cut với vector riêng nhỏ thứ hai như trên, phương pháp k-way-cut sử dụng những vector riêng nhỏ nhất đầu tiên một cách đồng thời cho phân vùng k-way.

**Thuật toán 2.3:** Thuật toán Ncuts cho giải thuật k-way cut [[9](#Shi00)]

Thuật toán 2.3: Thuật toán k-way cut

Bước 1: Ảnh vào I. Lập đồ thị G=(V, E) như ở two-way-cut.

Bước 2: Giải hệ trị riêng (**D**-**W**)x = λ**D**x. Sử dụng n vector riêng (lớn nhất hoặc nhỏ nhất) đầu tiên làm vector chỉ định n chiều cho mỗi điểm ảnh.

Bước 3: Dùng thuật toán gom cụm (K-means hoặc FCM) để nhóm các vùng của ảnh thành k’ nhóm (k’>k, k là số phân vùng cho trước) trên ma trận trị riêng.

Bước 4: Thực hiện phân vùng theo giải thuật 2.4 hoặc 2.5

**Giải thuật 2.4:** Thực hiện trộn 2 nhóm (trong k’ nhóm) cho tới khi còn lại k nhóm (k là số phân vùng cần xác định trước) theo chuẩn k-way Ncut, công thức 2.8

(2.8)

trong đó Ai : tập con thứ i của V. Sau mỗi bước trộn hai nhóm, cập nhật lại ma trận trọng số Wc: Wc(i,j) = assoc(Ai, Aj).

**Giải thuật 2.5:** Với k’ vùng ở bước 3, xây dựng đồ thị thu gọn Gc=(Vc, Ec, Wc). Trong đó, tập đỉnh Vc là tập k’ vùng ở bước 3. Sau đó thực hiện đệ qui chia đôi đồ thị Gc theo tiêu chuẩn 2-way-cut.

Thuật toán Normalized Cuts cho phân đoạn ảnh được phát triển dựa trên các khái niệm và tính chất từ lý thuyết đồ thị quang phổ. Các phép tính toán và biến đổi của phương pháp sử dụng các tính chất đại số tuyến tính của ma trận W và ma trận Laplace **D**-**W** của đồ thị và tạo mối liên hệ với những tính chất khác nhau của đồ thị. Tiêu chuẩn *normalized cut* có mối liên hệ với tính chất của Markov Random Walk, ma trận tương đồng W được chuẩn hóa để định nghĩa từ biến đổi ma trận xác suất P của random walk cho điểm ảnh[meila]. Phương pháp với ý tưởng sử dụng những vector riêng của ma trận Laplace để tìm những phân vùng của đồ thị. Đã có nhiều công trình liên quan dựa trên tính chất phân vùng đồ thị quang phổ tương tự Ncut như Ratio Cut [[7](#Son03)], Average Cut [[6](#Sis01)]. Phương pháp Ncut đầu tiên năm 1997 [[9](#Shi00)] được xem là đại diện cho ứng dụng đầu tiên của phân vùng quang phổ trong lĩnh vực thị giác máy tính cũng như xử lý ảnh. Nhìn chung, các thuật toán phân vùng quang phổ phổ biến nhất được phát triển theo bước chung, và các thuật toán khác nhau phụ thuộc vào cách xây dựng và sử dụng tính chất ma trận Laplace [[12](#Atu)].

**Thuật toán 2.6:** Thuật toán phân vùng quang phổ [[12](#Atu)]

Thuật toán 2.6: Thuật toán phân vùng quang phổ

Input: Ảnh cần phân vùng I hoặc tập dữ liệu (xi), k là số nhóm hay số phân vùng cần cho bài toán.

Output: Các phân vùng của ảnh vào.

Bước 1: Xây dựng đồ thị tương tự G = (V, E), với **W** ∈ Rn x n là ma trận trọng số của đồ thị, và là ma trận kề tương đồng.

Bước 2: Tính ma trận Laplace **L**. (Ncut với ma trận random walk Lrw)

Bước 3: Tính k vector riêng đầu tiên (có các trị riêng nhỏ nhất tương ứng) u1, …, uk của ma trận **L**.

Bước 4: Đặt U ∈ Rn x k là ma trận chứa k vector u1, …, uk theo cột.

Bước 5: Đặt yi (i=1..n) ∈ Rk là vector tương ứng với dòng thứ i của U.

Bước 6: Sử dụng thuật toán gom nhóm K-means để phân nhóm (yi)i=1..n ∈ Rk thành k nhóm C1, …, Ck.

Bước 7: Các nhóm hay phân vùng cuối cùng A1, …, Ak, với Ai = {j / yj ∈ Ci}.

## Thuật toán gom nhóm trong phân đoạn ảnh

Quá trình gom cụm trong phân đoạn ảnh được định nghĩa như quá trình xác định các nhóm có tính tương đồng nhau trong ảnh ban đầu. Kĩ thuật gom cụm được phân vào hai loại: gom cụm có giám sát – đòi hỏi có sự tương tác của con người để quyết định các tiêu chí cho việc gom nhóm; và gom cụm không giám sát – tự quyết định tiêu chí gom nhóm cho chính nó. Trong phân đoạn ảnh, quá trình gom cụm thường là quá trình tìm các nhóm có tính tương đồng nhau dựa trên thuộc tính của nó. Vì vậy, một cụm là một tập các đối tượng hay một nhóm các điểm ảnh có tính tương đồng nhau trong nhóm và tính không tương đồng giữa các cụm với nhau. Tuy nhiên trong phân đoạn ảnh dựa trên phân vùng quang phổ như đã trình bày trên, thuật toán gom cụm được sử dụng như bước cuối để từ đó tạo phân vùng cuối cùng từ ma trận U tạo bởi các vector riêng.

Rõ ràng, về thực nghiệm các thuật toán phân vùng quang phổ cần phải tính k vector riêng đầu tiên của ma trận Laplace. Với kích cỡ dữ liệu ảnh lớn, ma trận Laplace thường có kích thước lớn. Tuy nhiên, ma trận này có thể được biến đổi thành ma trận rời rạc trong quá trình xây dựng đồ thị, cụ thể sử dụng đồ thị k-nearest neighbor hay ε-neighborhood. Cho đến nay, đã có nhiều công trình với các kĩ thuật tìm vector riêng của ma trận (các eigensolver) tính k vector riêng đầu tiên cho ma trận rời rạc hiệu quả. Một điều đáng lưu ý là các vector trong các cơ sở trực giao của không gian riêng được mở rộng bởi các vector xác định cụm  theo dạng , với *ai* là hệ số. Điều này có nghĩa là các vector này là hằng theo từng đoạn (piecewise constant) trên từng nhóm. Vì vậy các vector riêng được tính bởi các eigensolver có thể cho thông tin về các nhóm. Và sau đó thuật toán gom nhóm có thể được sử dụng để tạo lại các nhóm hay phân vùng.

Đề tài chọn áp dụng hai thuật toán gom nhóm phổ biến nhất vào Ncut, nhằm so sánh sự khác biệt của hai thuật toán gom nhóm k-Means và FCM khi áp dụng vào thuật toán phân vùng quang phổ áp dụng trong thuật toán phân đoạn ảnh Ncuts.

### Thuật toán k-Means

k-Means [[13](#Sou13)] là một trong những thuật toán học không giám sát đơn giản nhất. Thuật toán chủ yếu phân nhóm tập dữ liệu d-chiều cho sẵn thành k nhóm (với k là số nguyên dương cho trước cố định). Ý thưởng chính của thuật toán là xác định k mẫu hay tâm cụm cho mỗi nhóm. Mỗi điểm dữ liệu sẽ được gán vào nhóm có tâm cụm tương ứng gần nhất. Giá trị của k tâm cụm này được tính lại sau mỗi bước gom nhóm. Như vậy, k tâm cụm thay đổi vị trí qua từng bước lặp cho tới khi không còn thay đổi. Quá trình gom nhóm chủ yếu là quá trình lặp thực hiện cực tiểu hóa hàm mục tiêu, tính lại k tâm cụm và gán điểm dữ liệu vào nhóm có tâm cụm gần nhất tương ứng. Hàm mục tiêu là hàm tổng sai số bình phương giữa điểm dữ liệu và tâm cụm của nhóm tính theo công thức (2.9)

 (2.9)

Trong đó,  là khoảng cách Euclide giữa mỗi điểm dữ liệu  và tâm cụm . Giá trị của hàm mục tiêu J là để đo mức độ xấp xỉ của n điểm dữ liệu với k nhóm.

**Thuật toán 2.7:** Thuật toán k-means

Thuật toán 2.7: Thuật toán k-means

Bước 1: Chọn k điểm ngẫu nhiên trong không gian dữ liệu làm k tâm cụm khởi tạo.

Bước 2: Mỗi điểm dữ liệu được gán vào một nhóm có tâm cụm tương ứng gần nhất.

Bước 3: Khi tất cả các điểm dữ liệu đã được gán, tính lại vị trí cho k tâm cụm mới cho k nhóm mới.

Bước 4: Lặp lại bước 2 và 3 cho tới khi không có sự thay đổi tâm cụm của các nhóm nữa và mỗi phần tử đều thuộc về một nhóm.

Mặc dù thuật toán được chứng minh là luôn có kết thúc ở bước lặp, nhìn chung k-Means đã không đạt được phân nhóm tối ưu trên toàn cục tương ứng với cực tiểu hóa toàn cục hàm mục tiêu. Mặc dù có những hạn chế, k-Means vẫn thường được chọn cho nhiều ứng dụng vì tính đơn giản và dễ cài đặt. Hơn nữa, k-Means là nền tảng cho các thuật toán cải tiến hơn với các dữ liệu có đặc trưng mờ.

### Thuật toán FCM

Thuật toán gom nhóm Fuzzy c-Means (FCM) được Dunn [[13](#Sou13)] phát triển và sau đó được tổng quát bởi Bezdek. Thuật toán FCM cũng tương tự với k-Means. Tuy nhiên, với k-Means mỗi điểm chỉ thuộc vào một phân vùng duy nhất. Trong khi đó với FCM, một điểm có thể thuộc vào một hay nhiều nhóm tùy thuộc vào mức độ của hàm thuộc của mỗi điểm ứng với tâm của các nhóm (sử dụng logic mờ). Chính vì vậy, FCM có tính linh hoạt hơn với tập dữ liệu có các cụm dữ liệu chồng nhau (tính tương đồng cao với ảnh). Thuật toán chủ yếu dựa trên việc tối ưu hóa hàm mục tiêu theo công thức (2.10)

|  |  |
| --- | --- |
|  | (2.10) |

Trong đó, ma trận thuộc *U=[uij] ∈ Mfcm* là phân vùng mờ của tập dữ liệu Z, uij ∈ [0,1] chỉ độ phụ thuộc của điểm xi vào cluster thứ j, với .

*V = [c1, c2, …, cc]* là vector mẫu hay tâm cụm của các C nhóm, được tính theo chuẩn khoảng cách;  là số mũ xác định độ mờ của việc phân nhóm.

**Thuật toán 2.8:** Thuật toán FCM

Thuật toán 2.8: Thuật toán FCM

Bước 1: Tạo ngẫu nhiên C nhóm với C tâm tương ứng. Khởi tạo ma trận thuộc U(0).

Bước 2: Tính lại tâm cj và ma trận thuộc uij theo công thức (2.11) và (2.12)

 (2.11)

 (2.12)

Bước 3: Nếu  thì dừng, ngược lại quay lại bước 2.

(ε : ngưỡng cho trước thích hợp, ε đủ nhỏ)

Độ phức tạp thuật toán gần như tuyến tính O(k2ndt) [[13](#Sou13)] (d: số chiều của phần tử trong tập dữ liệu). Như vậy, so với thuật toán k-Means thì độ phức tạp của FCM cũng tương đối; và sự phân cụm của FCM cũng tương đối chính xác hơn k-Means với dữ liệu lớn có các cụm chồng nhau.

## Tính toán song song

Hiện nay, mô hình xử lý song song đã và đang có nhiều ứng dụng mạnh mẽ cho các mô hình tính toán phức tạp trong nhiều lĩnh vực khoa học và kĩ thuật như các bài toán xử lý ảnh 2D, 3D, bài toán thăm dò dầu khí, bài toán dự báo thời tiết,…. Xử lý song song giải quyết được nhiều vấn đề mà mô hình xử lý tuần tự gặp phải như vấn đề thời gian thực hiện chương trình, tốc độ xử lý, khả năng lưu trữ của bộ nhớ, xử lý dữ liệu với qui mô lớn …. Trong nhiều ứng dụng thực tế, các mô hình tính toán phức tạp cho các vấn đề lớn hơn, đòi hỏi tính chính xác hơn cần hệ thống tính toán mạnh hơn và không gian bộ nhớ nhiều hơn mà máy tính tuần tự truyền thống có thể cung cấp. Vì vậy, cùng với sự gia tăng về số bộ xử lý trong kiến trúc hệ thống máy tính cũng như khả năng giao tiếp giữa chúng, việc sử dụng tính toán song song là cần thiết áp dụng để thu được hiệu năng tính toán trong nghiên cứu và ứng dụng.

Trong tính toán tuần tự truyền thống với một bộ xử lý, tại mỗi thời điểm chỉ thực hiện một lệnh. Bài toán được tách ra thành một chuỗi các lệnh rời rạc. Các câu lệnh được thực hiện một cách tuần tự và tại mội thời điểm chỉ được thực hiện duy nhất một câu lệnh. Máy tính một bộ xử lý có những giới hạn về hiệu năng tính toán. Thời gian trên bộ nhớ và thiết bị lưu trữ là cản trở chính mà các nhà thiết kế máy tính phải đối mặt. Hạn chế về gian do ràng buộc cấu trúc vật lý và rất khó giảm thiểu. Các phần mềm hay thuật toán được thiết kế theo kiểu tính toán tuần tự truyền thống. Những thuật toán và phần mềm này phù hợp với những máy tính có một bộ xử lý (CPU). Cách tính toán tuần tự trên máy đơn xử lý được minh họa trong hình 2.2.

sn

si

..

s1

s3

s2

CPU

Bài toán cần giải quyết

Các câu lệnh

Hình 2.2. Cách giải quyết vấn đề của máy tính tuần tự

Mặc dù tốc độ xử lý của máy tính tuần tự được cải tiến liên tục trong nhiều năm qua, nhưng với nhu cầu ngày một cao về việc cần đạt hiệu năng tính toán tốt hơn, tính toán song song đã được tiếp cận và nghiên cứu.

Trong tính toán song song, nhiều bộ xử lý cùng kết hợp với nhau cùng giải quyết một bài toán. Bài toán cũng được chia thành nhiều thành phần và có thể được thực hiện đồng thời. Mỗi phần lại được tách thành các câu lệnh rời rạc và được thực hiện đồng thời trên các bộ xử lý khác nhau hay trên hệ thống đa bộ xử lý. Hệ thống song song bao gồm một tập hợp các bộ xử lý. Chúng được kết nối với nhau theo một kiến trúc nào đó cùng với các nghi thức trao đổi thông tin, dữ liệu và các qui tắc hoạt động giữa các bộ xử lý này. Mô hình tính toán song song được minh họa như trong hình 2.3

Toàn bộ vấn đề được chia nhỏ ra để giải quyết song song

Hình 2.3. Cách thực hiện các phần song song trên máy tính song song

s1

s2

s3

sn

.

si

. .

CPU

CPU

CPU

CPU

Hay nói cách khác tính toán song song là quá trình tính toán gồm nhiều tiến trình cùng tham gia tính toán giải quyết một vấn đề, và nói chung phải thực hiện trên các hệ thống đa bộ xử lý. Như vậy để thực hiện tính toán song cần phải có một trong các tài nguyên tối thiểu sau:

* Một máy tính nhưng có nhiều bộ xử lý.
* Các máy tính được kết nối thành mạng.
* Hoặc tổ hợp cả hai loại trên.

Do vậy để tăng tốc độ giải quyết các vấn đề cần đến máy tính, các nhà khoa học thấy rằng cần thiết phải khai thác được khả năng tính toán song song trên thiết bị phần cứng, và cùng với nó là lập trình song song và các thật toán song song. Mục đích tính toán song song là tận dụng các kỹ năng của các hệ đa bộ xử lý để thực hiện các phép toán nhanh hơn, tức là sử dụng nhiều bộ xử lý đồng thời. Việc tính toán song song không những làm tăng tốc độ giải quyết vấn đề mà còn giải quyết được các vấn đề phức tạp hơn, lớn hơn…

Vậy tính toán song song hay xử lý song song là quá trình xử lý thông tin trong đó nhấn mạnh việc nhiều đơn vị dữ liệu được xử lý đồng thời bởi một hay nhiều bộ xử lý để giải quyết một bài toán. Tốc độ của tính toán song song được tính theo tỉ số Speedup = timett / timess. Trong đó, timett là thời gian thực hiện trong tình huống xấu nhất theo thuật toán tuần tự nhanh nhất, timess là thời gian thực hiện trong tình huống xấu nhất theo thuật toán song song đang xét. Hiệu quả (efficient) của thuật toán song song được tính bằng Efficient = Tốc độ / Số bộ xử lý tham gia tính toán. Giá trị (Cost) của một quá trình tính toán trên hệ thống song song được tính như sau: Cost = Độ phức tạp tính toán × Số lượng bộ xử lý tham gia tính toán.

### Kiến trúc tính toán song song

Thuật toán song song liên quan chặt chẽ đến kiến trúc máy tính, phần mềm hệ thống, thuật toán và cả ngôn ngữ lập trình, …. Hay nói cách khác, khi lập trình song song trên cơ sở các thuật toán song song, người lập trình còn phải dựa vào kiến trúc của máy tính song song. Máy tính song song bao gồm một tập hợp các bộ xử lý. Chúng được kết nối với nhau theo một kiến trúc nào đó cùng với các nghi thức trao đổi dữ liệu và các quy tắc hoạt động giữa các bộ xử lý này. Một ưu thế khi sử dụng máy tính song song hiện nay là các hệ điều hành hiện nay đều được hỗ trợ đa nhiệm nên có thể dùng phương pháp lập trình song song trên đó. Vấn đề là làm thế nào để các bộ xử lý độc lập cùng hoạt động, cùng tham gia giải quyết một vấn đề, tức là các tiến trình được thực hiện trên các bộ xử lý một cách đồng bộ, phải trao đổi với nhau, và phải cùng giải quyết một vấn đề cho trước.

Phân loại kiến trúc máy tính song song kinh điển và rộng rãi hiện nay là kiểu phân loại Flynn’s Taxonomy của Flynn (1966). Cách phân loại này dựa vào cách thức mà máy đó tải các dòng chỉ thị, luồng dữ liệu và cấu trúc bộ nhớ. Mỗi cách đó chỉ ra duy nhất một trạng thái đơn hoặc đa chỉ thị hay dữ liệu. Bốn cách phân loại theo Flynn được trình bày tóm tắt như trong bảng sau.

|  |  |
| --- | --- |
| **S I S D**  **Single Instruction,**  **Single Data** | **S I M D**  **Single Instruction,**  **Multiple Data** |
| **M I S D**  **Multiple Instruction,**  **Single Data** | **M I M D**  **Multiple Instruction, Multiple Data** |

Bảng 2.1: Phân loại kiến trúc máy tính song song

SISD là mô hình máy tính đơn dòng lệnh, đơn dòng dữ liệu. Đây là loại máy tính kiểu tuần tự truyền thống và tất cả các bộ chức năng đều được xử lý bởi một bộ xử lý. Máy tính SISD có kiến trúc một CPU cổ điển phổ biến nhất, còn được gọi là máy tính kiểu von Neuuman. Với kiến trúc SISD, tại một thời điểm hay chu kì làm việc thì chỉ một dòng lệnh được thực thi và chỉ một luồng dữ liệu được đọc vào.

SIMD là mô hình máy tính đơn chỉ lệnh, đa dòng dữ liệu. Đây là loại máy tính theo kiểu song song, có một đơn vị điều khiển và có thể điều khiển nhiều đơn vị xử lý. Tại một thời điểm, tất cả các đơn vị xử lý cùng thực thi cùng một luồng lệnh, và mỗi đơn vị xử lý thực hiện cùng phép toán trên một luồng dữ liệu riêng. Đây là mô hình máy tính hiện đại, đặc biệt là trong các máy tính có bộ vi xử lý đồ họa (Graphic Processing Units - GPUs).

MISD là mô hình máy tính đa dòng lệnh, đơn dòng dữ liệu. Một dòng dữ liệu duy nhất được đọc vào từ nhiều đơn vị xử lý. Đây là một kiến trúc song song mà trong đó tại mỗi thời điểm các đơn vị xử lý thực hiện các phép toán khác nhau trên cùng một dữ liệu một cách độc lập.

MIMD là mô hình máy tính đa dòng lệnh, đa dòng dữ liệu. Đây là kiến trúc phức tạp nhưng hỗ trợ xử lý song song cao nhất, phổ biến nhất hiện nay, và đang được ứng dụng rộng rãi trong nhiều lĩnh vực. Mô hình MIMD có đa bộ xử lý, trong đó các bộ xử lý có thể hoạt động một cách độc nhau trên các bộ dữ liệu khác nhau một cách đồng bộ hoặc không đồng bộ, xác định hoặc không xác định.

### Phân loại tính toán song song

Tính toán song song là một hình thức tính toán mà trong đó nhiều phép toán được xử lý một cách đồng thời, dựa trên nguyên lý một vấn đề lớn được chia thành nhiều vấn đề nhỏ hơn, và các vấn đề nhỏ này được giải quyết một cách song song. Tính toán song song được chia ra nhiều loại khác nhau như: Bit-level, Instructional, Data, Task.

Bit-level parallelism: là một dạng tính toán song song dựa trên việc tăng kích thước đơn vị lưu trữ của đơn vị xử lý (processor word size) khi bộ xử lý phải thực hiện một phép toán trên các biến có kích thước lớn hơn chiều dài của đơn vị lưu trữ này. Mô hình được phát triển từ công nghệ VLSI (Very Large Scale Integration). Ví dụ xét trường hợp một bộ xử lý 8-bit cộng hai số nguyên 16-bit sẽ thực hiện việc cộng trong hai bước với từng cụm 8 bit. Bộ xử lý 8-bit đòi hỏi phải có hai lệnh để thực hiện một phép toán. Đầu tiên bộ xử lý sẽ cộng 8-bit thấp từ mỗi số nguyên và sau đó sẽ thêm 8 bit ở bậc cao.

Instruction-level parallelism: là một dạng tính toán song song dựa trên việc xác định bao nhiêu lệnh hay phép toán của một chương trình máy tính có thể được xử lý đồng thời. Các lệnh được chia thành các nhóm hoặc được đặt lại trật tự và sau đó được xử lý mà không thay đổi kết quả cuối cùng. Các phép toán này được thực hiện độc lập hay không là phụ thuộc vào sự ràng buộc của thao tác dữ liệu và phụ thuộc vào kiến trúc xử lý.

Data parallelism: là một hình thức tính toán song song trên nhiều bộ xử lý, trên cơ chế phân bố dữ liệu cho các đơn vị xử lý thực hiện cùng một phép toán trên các nút tính toán song song.

Task parallelism: là một dạng tính toán song song ngược lại với Data parallelism. Nó thực hiện tính toán song song trên cơ chế phân bố các nhiệm vụ hay lệnh trên các nút tính toán song song trên cùng hoặc khác dữ liệu.

### Bộ nhớ trong tính toán song song

1. ***Bộ nhớ chia sẻ***

Kiến trúc bộ nhớ chia sẻ của máy tính song song là khác nhau, nhưng nhìn chung cho phép tất cả các bộ xử lý trong hệ đa bộ xử lý truy cập vào theo kiểu không gian địa chỉ toàn cục. Các bộ xử lý có thể hoạt động một cách độc lập nhưng chia sẻ cùng một tài nguyên bộ nhớ. Khi có sự thay đổi trong một vùng nhớ bởi một bộ xử lý nào đó thì các bộ xử lý còn lại cũng nhìn thấy và ảnh hưởng bởi sự thay đổi đó. Một bộ xử lý này có thể trao đổi thông tin với bộ xử lý khác bằng cách ghi vào bộ nhớ toàn cục và các bộ xử lý khác có thể đọc dữ liệu tại cùng vị trí đó trong bộ nhớ. Chính điều này cho phép trao đổi thông tin giữa các bộ xử lý. Nhưng có một vấn đề là sẽ có đồng thời nhiều bộ xử lý cùng truy cập đến một vùng nhớ trong bộ nhớ toàn cục. Chính vì vậy người lập trình cần phải chú ý đến việc sự đồng bộ khi chương trình truy cập đến bộ nhớ toàn cục.

Kiến trúc bộ nhớ chia sẻ được chia thành hai nhóm chính dựa trên thời gian truy cập vùng nhớ của các bộ xử lý: Bộ nhớ chia sẻ đồng bộ (Uniform Memory Access - UMA) và Bộ nhớ chia sẻ không đồng bộ (Non-uniform Memory Access - NUMA).

Bộ nhớ UMA: Thường được thiết kế cho các kiến trúc máy tính có bộ đa xử lý đối xứng (SMP- Symetric MultiProcessors). Với đa xử lý đối xứng, công việc được phân phối ngang nhau cho các bộ xử lý để không có bộ xử lý nào nhàn rỗi trong khi có bộ xử lý khác làm việc quá sức. Các tài nguyên hệ thống như bộ nhớ được chia sẻ cho tất cả các bộ xử lý trong hệ thống. Tất cả các bộ xử lý đều có thể truy cập đến bộ nhớ toàn cục đồng nhất và thông qua một BUS dùng chung. Kiến trúc này còn được gọi là Cache coheren-UMA (CC-UMA), có nghĩa là khi một bộ xử lý bất kì có sự cập nhật lên một vị trí trong bộ nhớ thì tất cả các bộ xử lý khác đều nhận biết được sự cập nhật đó.

Bộ nhớ NUMA: Mô hình NUMA gồm nhiều mô hình UMA hay SMP nhỏ được nối với nhau thông qua đường truyền vật lý được minh họa như hình. Mỗi một SMP có thể truy cập tới bộ nhớ của các SMP khác. Tuy nhiên với kiến trúc kiểu này thì tất cả các SMP không thể cùng truy cập tới các bộ nhớ cùng lúc và do việc kết nối bằng đường truyền vật lý nên việc truy cập đến bộ nhớ sẽ chậm hơn so với kiến trúc UMA.

Với kiến trúc bộ nhớ chia sẻ chung giúp người lập trình thuận tiện trong việc viết các chương trình song song. Dữ liệu chia sẻ giữa các nhiệm vụ được đảm bảo cả hai tiêu chuẩn là nhanh và đồng bộ. Tuy nhiên với kiến trúc này cũng có một số khó khăn khi mở rộng số lượng các bộ xử lý vì có thể làm tăng lưu lương truy cập giữa bộ xử lý và bộ nhớ. Với hệ thống như vậy đòi hỏi người lập trình có sự xử lý về tính đồng bộ để đảm bảo tính đúng đắn của dữ liệu dùng chung.

1. ***Bộ nhớ phân tán***

Khác với kiến trúc bộ nhớ chia sẻ, kiến trúc bộ nhớ phân tán không có khái niệm bộ nhớ chia sẻ dùng chung toàn cục đến tất cả các bộ xử lý. Mỗi bộ xử lý có một bộ nhớ cục bộ của riêng nó và các bộ xử lý không thể truy xuất đến địa chỉ của vùng nhớ của bộ xử lý khác. Vì mỗi bộ xử lý có bộ nhớ cục bộ riêng nên các bộ xử lý hoạt động độc lập nhau và tất nhiên tác động của một bộ xử lý lên bộ nhớ cục bộ của nó sẽ không ảnh hưởng đến bộ nhớ của các bộ xử lý khác. Khi một bộ xử lý cần truy xuất đến dữ liệu của một bộ xử lý khác sẽ thông qua các nhiệm vụ mà người lập trình đã định nghĩa trước đó. Vì vậy sự đồng bộ dữ liệu là cần thiết đối với người lập trình trong kiến trúc này. Trong kiến trúc song song với bộ nhớ phân tán, các bộ xử lý giao tiếp với nhau thông qua thông điệp được truyền qua mạng liên kết gồm các liên kết truyền thông trực tiếp giữa một số cặp bộ xử lý, như hình.

Rõ ràng tốc độ truy xuất tối ưu đạt được khi các bộ xử lý được nối trực tiếp với nhau. Tuy nhiên điều này thường là không khả thi do số lượng các liên kết là quá lớn giẫn đến việc tăng giá thành của hệ thống. Cách thứ hai được sử dụng là các bộ sử lý liên lạc thông qua một BUS chia sẻ. Điều này dẫn đến việc độ trễ cao khi số lượng bộ xử lý lớn dẫn đến vấn đề tranh chấp BUS. Với kiến trúc bộ nhớ phân tán dễ dàng cho việc mở rộng số lượng các bộ xử lý. Mỗi bộ xử lý có thể truy xuất vào bộ nhớ riêng nhanh chóng mà không có sự can thiệp vật lý ngoài nào và không cần phải giữ sự liên kết giữa các bộ nhớ cache. Tuy nhiên với tổ chức bộ nhớ kiểu này cũng gặp khó trong việc thiết kế cấu trúc dữ liệu dựa trên bộ nhớ toàn cục.

1. ***Bộ nhớ lai – Bộ nhớ kết hợp giữa chia sẻ và phân tán***

Hiện nay, xu hướng của các máy tính song song nhanh nhất trên thế giới dựa trên sự kết hợp của kiến trúc bộ nhớ chia sẻ và bộ nhớ phân tán. Trong kiến trúc này gồm nhiều thành phần là các SMP với bộ nhớ chia sẻ và các bộ xử lý trong các thành phần bộ này chỉ có thể truy cập bộ nhớ của thành phần đó như là bộ nhớ toàn cục của thành phần đó. Và các thành phần này được nối với nhau như trong kiến trúc bộ nhớ phân tán, là một mạng nối nhiều SMP hay nhiều thành phần với kiến trúc bộ nhớ phân tán.

Các SMP chỉ có thể truy xuất đến bộ nhớ toàn cục trong thành phần bộ nhớ chia sẻ của chúng và không thể truy xuất đến các bộ nhớ chia sẻ ở các thành phần SMP khác. Chính vì vậy, thành phần giao tiếp trên mạng kết nối như kiến trúc bộ nhớ phân tán đòi hỏi phải có để đảm bảo chuyển tải dữ liệu giữa các SMP.

### Lập trình song song

#### *Mô hình lập trình song song*

Một trong những vấn đề quan trọng trong lập trình song song là thao tác hay cầu nối giữa người lập trình và mô hình song song của ứng dụng. Không giống như lập trình tuần tự, việc trừu tượng hóa cấp cao bằng các mô hình lập trình song song cho phép tăng hiệu quả của của kiến trúc song song có sẵn. Rõ ràng mô hình lập trình ở mức thấp không còn hiệu quả với một lượng các lệnh thực hiện rất lớn với kiến trúc phần cứng và chương trình ứng dụng hiện nay. Chính vì vậy, mô hình lập trình trừu tượng mức cao có thể quản lý trực tiếp quá trình thực hiện song song của nhiều bộ xử lý cũng như tương tác giữa các bộ xử lý.

Mô hình lập trình song song là một tập hợp các công nghệ phần mềm đảm nhiệm hai vai trò chính là đặc tả các thuật toán song song và là cầu nối giữa ứng dụng và hệ thống song song. Mặc dù có một số mô hình được cho là triển khai hiệu quả hơn hẳn các mô hình khác. Nhưng không có một mô hình nào là tốt nhất, mà nó thường là sự kết hợp giữa kiến trúc phần cứng có sẵn và sự lựa chọn phù hợp. Có nhiều loại mô hình lập trình song song được nghiên cứu và ứng dụng. Sau đây là một số mô hình phổ biến hiện nay.

1. *Mô hình truyền thông điệp (Message passing model)*

Đây là một trong những mô hình phổ biến trong lập trình song song và thường được sử dụng trong các kiến trúc bộ nhớ phân tán. Trong các chương trình với mô hình truyền thông điệp, cũng giống như mô hình Master/Slave (Task/Channel), vấn đề cũng được chia thành các nhiệm vụ. Tất cả các nhiệm vụ chỉ sử dụng dữ liệu trong bộ nhớ cục bộ của chúng trong suốt quá trình tính toán. Nhiều nhiệm vụ có thể được thực hiện trên cùng một máy, và các máy được nối với nhau thông qua một mạng vật lý. Mỗi nhiệm vụ được xác định bởi một tên duy nhất, và các nhiệm vụ trao đổi dữ liệu với nhau thông qua giao thức gửi và nhận thông điệp theo tên nhiệm vụ đã được đặt.

Như vậy với mô hình truyền thông điệp, việc giao nhiệm vụ là động, có thể thực hiện nhiều nhiệm vụ trên một bộ xử lý, hoặc thực thi các chương trình khác nhau thông qua các nhiệm vụ khác nhau. Trên khía cạnh lập trình song song, việc thực thi mô hình truyền thông điệp thường dựa trên các hàm chuẩn hay thủ tục con được viết trong thư viện truyền thông điệp. Người lập trình cần xác định tất cả các bước song song hóa của chương trình và áp dụng vào mô hình truyền thông điệp với thư viện có sẵn. MPI (Message Passing Interface) là thư viện mô hình truyền thông điệp được sử dụng rộng rãi hiện nay.

1. *Mô hình luồng (Thread model)*

Trong mô hình luồng, một tiến trình đơn thường có thể có nhiều luồng thực thi đồng thời. Mỗi luồng có dữ liệu riêng của nó và chia sẻ dữ liệu toàn cục của chương trình chính. Bất kì luồng nào cũng có thể thực hiện bất kì thủ tục con nào với các luồng khác. Mô hình sử dụng các khóa (lock) và các biến điền kiện cho việc thiết lập các giao thức truyền thông và đồng bộ giữa các luồng. Chính vì vậy mô hình luồng đòi hỏi người lập trình phải linh hoạt trong xử lý đồng bộ dữ liệu. Nhìn chung, mô hình luồng phù hợp với các ứng dụng đa dạng dữ liệu. Tuy nhiên, các ứng dụng sử dụng mô hình luồng tương đối khó triển khai bởi việc quản lý sự truyền thông và đồng bộ giữa các luồng đòi hỏi tính chặt chẽ cao, dễ mắc phải lỗi chia sẻ dữ liệu.

1. *Mô hình song song dữ liệu (Data Parallelism Model)*

Mô hình song song dữ liệu thường được áp dụng hiệu quả trong các chương trình song song với tập dữ liệu lớn. Tập dữ liệu thường được tổ chức theo cấu trúc chung, như mảng hoặc dữ liệu khối. Một tập các nhiệm vụ thực hiện song song trên cùng cấu trúc dữ liệu đó. Nhưng mỗi nhiệm vụ thực thi cùng một phép toán hay thao tác chỉ trên một phần nào đó của cấu trúc dữ liệu. Như vậy mỗi phép toán được thực thi trên một phân vùng của cấu trúc dữ liệu được xem như là một nhiệm vụ độc lập. Mỗi phân vùng đó được xem như dữ liệu cục bộ của một nhiệm vụ tính toán song. Chính vì vậy, người lập trình cần xác định dữ liệu được phân bố như thế nào trên các bộ xử lý, hay nói cách khác dữ liệu được phân chia đến các nhiệm vụ như thế nào.

Trong kiến trúc bộ nhớ chia sẻ, tất cả các nhiệm vụ truy xuất vào cấu trúc dữ liệu thông qua bộ nhớ toàn cục. Còn trong kiến trúc bộ nhớ phân tán, cấu trúc dữ liệu được chia ra như các phân vùng trong bộ nhớ cục bộ của mỗi nhiệm vụ.

1. *Mô hình lai (Kết hợp các mô hình lập trình song song)*

Tùy vào đặc thù của ứng dụng, các mô hình lập trình song song có thể được áp dụng kết hợp với nhau.

Mô hình SPMD (Single Program Multiple Data): là mô hình lập trình song song cấp cao được xây dựng dựa trên sự kết hợp từ các mô hình đã được đề cập trước đó như mô hình truyền thông điệp, mô hình song song dữ liệu. Trong mô hình SPMD, một chương trình được thực thi bởi các nhiệm vụ một cách đồng thời có thể trên dữ liệu khác nhau. Tại cùng thời điểm, các nhiệm vụ có thể thực hiện cùng một lệnh hoặc các lệnh khác nhau trong cùng một chương trình. Chương trình SPMD cần phải có tính logic cao khi thiết kế các nhiệm vụ cho những phần của chương trình. Các nhiệm không nhất thiết phải thực hiện hết toàn bộ các phần của chương trình mà có thể chỉ thực hiện song song ở một phần nào đó cần thiết của chương trình.

Mô hình MPMD (Multiple Program Multiple Data): giống với mô hình SPMD, cũng là mô hình kết hợp từ nhiều mô hình lập trình song song. Các ứng dụng MPMD thường có nhiều chương trình giao phó cho một hoặc nhiều nhiệm vụ thực thi đồng thời. Trong các ứng dụng chạy song song, mỗi nhiệm vụ có thể được thực thi cùng hoặc khác chương trình như các nhiệm vụ khác.

#### *Lập trình song song trên cùng một máy tính đa lõi và đa bộ xử lý*

Nhu cầu tính toán trong lĩnh vực khoa học và công nghệ ngày càng cao, dần trở thành một thách thức lớn nếu chỉ sử dụng phương pháp xử lý tuần tự chạy trên một bộ xử lý. Xử lý song song trên các hệ thống máy tính song song, siêu máy tính là chìa khóa để giải quyết bài toán khó này. Khi nói đến tính toán song người ta thường nghĩ đến một hệ thống song song có thể là một máy tính đơn với nhiều bộ xử lý hoặc sử dụng nhiều máy tính đồng thời được kết nối thành mạng, hoặc các siêu máy tính với hiệu suất cao.

Những năm gần đây, khả năng xử lý của các bộ xử lý được cải thiện rất đáng kể, đặc biệt là tốc độ xử lý, nhưng do hạn chế về mặt vật lý, nên tốc độ của các bộ xử lý cũng bị giới hạn ở một mức nào đó. Sự thay đổi cơ bản trong công nghệ phần cứng thúc đẩy các nhà sản xuất quan tâm sản xuất một bộ xử lý tích hợp một số đơn vị xử lý trên một chip. Những con chip này có bộ điều khiển độc lập và có thể truy xuất cùng một bộ nhớ đồng thời. Thông thường, thuật ngữ bộ xử lý một lõi được dùng cho bộ xử lý có đơn vị xử lý đơn và bộ xử lý đa lõi dùng cho bộ xử lý có vài đơn vị xử lý tính toán. Như vậy việc sử dụng những bộ xử lý đa lõi có thể vận hành chiếc máy tính như một hệ thống song song nhỏ. Trong thực tiễn cũng xuất hiện các vấn đề mà các hệ thống đơn bộ xử lý không đáp ứng được yêu cầu về mặt thời gian, do đó cũng phải sử dụng các hệ thống đa bộ xử lý và đòi hỏi các bộ xử lý này phải cùng tính toán. Như vậy, với tính toán tuần tự vì chỉ có một bộ xử lý nên chỉ thực hiện một phép toán tại mỗi thời điểm, hay kiểu SISD. Nhưng trong tính toán song song, vì có nhiều bộ xử lý, các bộ xử lý này cùng kết hợp với nhau để cùng tính toán giải quyết một vấn đề. Tức là có thể có nhiều thao tác, hoặc phép toán được thực hiện đồng thời, điều đó làm giảm đáng kể thời gian tính toán và giảm độ phức tạp của việc giải quyết vấn đề. Các CPU đa lõi và nhiều nhân có thể được tận dụng cho cho mô hình luồng công việc thực thi song song cho nhiều nhiệm vụ. Những nhiệm vụ song song này có những chỉ lệnh khác nhau trên tập dữ liệu khác nhau, hay kiểu mô hình MIMD.

#### *Lập trình song song trên hệ thống cụm nhiều máy*

Tuy nhiên, với sự cải thiện đáng kể trên các hệ thống đa bộ xử lý và sự phát triển của các bộ xử lý đa lõi như vậy, tốc độ tính toán và không gian bộ nhớ tuy được cải tiến nhiều nhưng vẫn chưa đáp ứng được một số yêu cầu của một số bài toán thực tế đặt ra. Một giải pháp nữa để xây dựng một hệ thống máy tính song song - siêu máy tính là kết hợp rất nhiều bộ xử lý trung tâm CPU (nút tính toán) với nhau thông qua một mạng chuyên dụng tốc độ cao (mô hình cluster).

Thay vào đó, hệ thống song song thường được xây dựng như một cụm song song (parallel cluster), với mong muốn đạt hiệu năng cao hơn so với máy đơn. Một hệ thống cụm song song là một nhóm các máy tính được nối với nhau, gọi là các nút (node) làm việc cùng với nhau trong một mạng cục bộ tốc độ cao. Một ưu thế khi sử dụng hệ thống song song hiện nay là các hệ điều hành hiện nay đều được hỗ trợ đa nhiệm nên có thể áp dụng phương pháp lập trình song song trên đó. Vấn đề là làm thế nào để các bộ xử lý độc lập cùng hoạt động, cùng tham gia giải quyết một vấn đề, tức là các tiến trình được thực hiện trên các bộ xử lý một cách đồng bộ, có thể trao đổi với nhau và cùng giải quyết một vấn đề cho trước.

Tuy nhiên cluster vẫn có những nhược điểm cơ bản xét từ góc độ hiệu quả khai thác và sử dụng, như giá thành, tỷ lệ giá thành/ hiệu năng sử dụng, chi phí vận hành (năng lượng, quản trị, bảo trì hệ thống) rất cao.

#### *Lập trình song song trên thiết bị đồ họa*

Thiết bị đồ họa [[14](#NVI11)] (Graphics Processing Unit – GPU, đôi khi còn được gọi là VPU – Virtual Processing Unit) là một bộ xử lý chuyên dụng nhận nhiệm vụ tăng tốc, xử lý đồ họa, hỗ trợ lập trình song song ở mức cao và tính toán với dữ liệu lớn. GPU tích hợp hàng trăm nhân xử lý thực hiện xử lý tính toán trên một hệ thống bộ nhớ chung, và đặc biệt đó là thiết bị đồ họa là có thể lập trình được. Các GPU hiện đại có năng suất cao trong xử lý đồ họa máy tính. Với cấu trúc mang tính xử lý song song mạnh mẽ, GPU cho thấy nó hiệu quả hơn CPU rất nhiều trong nhiều thuật toán phức tạp đòi hỏi hiệu năng cao.

Bộ xử lý đồ họa hiện đại GPU được thiết kế chuyên biệt với hàng trăm hàng ngàn nhân xử lý có thể tận dụng để tối ưu hóa cho mô hình tính toán đơn chỉ lệnh cho đa luồng STMT

|  |
| --- |
|  |
| Hình 2.4. Kiến trúc CPU đa nhân và kiến trúc GPU |

Công nghệ GPU đã tạo được những bức ảnh tuyệt vời về hình dạng phức tạp, các hiệu ứng quang học phức tạp, các chuyển động liên tục liền mạch với các ứng dụng 2D, 3D và tăng hiệu năng nhiều ứng dụng kĩ thuật thông minh khác.

Cấu trúc xử lý đồ họa GPU là một mô hình đồ họa hình ống, trong đó, các đỉnh, điểm ảnh là có thể lập trình được. Mô hình đồ họa hình ống bao gồm các quá trình: các biển đổi đỉnh và ánh sáng, xử lý hình học biến đổi dựng hình với các tính chất nguyên thủy, quá trình quét mành chuyển đổi đa giác thành các điểm ảnh, quá trình xử lý điểm ảnh với các thiết lập kết cấu và bộ đổ bóng điểm ảnh, tạo bộ khung đệm với màu sắc và Z-value. Kiến trúc GPU ngày càng tập trung vào việc thực hiện đổ bóng điểm ảnh. Các bộ đổ bóng điểm ảnh gồm các thread chạy các chương trình khác nhau trên một lõi linh hoạt.

|  |
| --- |
|  |
| Hình 2.5. Kiến trúc bộ nhớ GPU |

Kiến trúc bộ nhớ được chia ra làm nhiều loại vùng nhớ với các chức năng truy xuất và lưu trữ khác nhau được minh họa trong hình. Kiến trúc bộ nhớ GPU gồm bộ nhớ các thanh ghi (Registers), bộ nhớ cục bộ (Local memory), bộ nhớ chia sẻ (Shared memory), bộ nhớ toàn cục (Global memory) và bộ nhớ hằng (Constant memory), vùng nhớ Texture. Xét từ việc truy xuất của các tiểu trình cho tới nhóm gồm nhiều tiểu trình (block, grid), vùng nhớ thanh ghi registers chỉ cho phép sự truy xuất đọc ghi của các tiểu trình, và chỉ riêng nội bộ trên một tiểu trình. Vùng nhớ cục bộ cũng cho phép truy xuất trên nội bộ của mỗi tiểu trình. Vùng nhớ chia sẻ cho phép truy xuất của một nhóm các tiểu trình, gọi là khối tiểu trình (thread block). Như vậy chỉ có các tiểu trình trong cùng một khối mới có thể truy xuất đến cùng một vùng nhớ chia sẻ, đây là vùng nhớ trung gian cấp cao và quan trọng trong phân cấp vùng nhớ của GPU. Với vùng nhớ hằng thì có thể truy xuất đến từ các tiểu trình trong cùng grid gồm một nhóm các khối tiểu trình, vùng nhớ này chỉ cho phép đọc dữ liệu và có thể truyền dữ liệu từ vùng nhớ host (vùng nhớ bên ngoài thiết bị GPU). Tương tự với vùng nhớ hằng, vùng nhớ Texture cũng là vùng nhớ chỉ đọc, chỉ được truy xuất dữ liệu từ các grid tiểu trình và có tốc độ truy xuất nhanh hơn vùng nhớ chia sẻ. Cuối cùng là vùng nhớ toàn cục cho phép sự truy xuất đọc và ghi từ tất cả các tiểu trình trên GPU, vùng nhớ này cũng cho phép truyền dữ liệu giữa vùng nhớ host và thiết bị GPU.

Song song với công nghệ phần cứng chuyên biệt GPU, các ngôn ngữ hỗ trợ lập trình song song hóa cũng được phát triển và hỗ trợ mạnh mẽ trên thiết bị GPU như CUDA, Matlab hỗ trợ lập trình song song trên GPU.

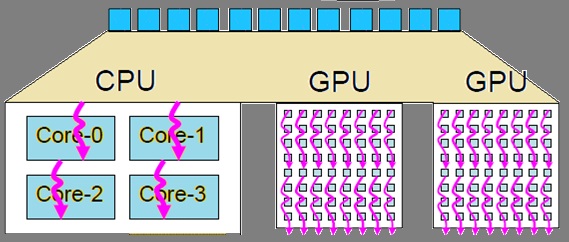
Nhằm hỗ trợ cho việc phát triển các ứng dụng dựa trên GPU rút ngắn thời gian lập trình và đạt hiệu suất cao, NVIDIA công bố cùng với GPU là bộ engine, các thư viện CUDA và công cụ lập trình NEXUS. CUDA là một kiến trúc kết hợp xử lý giữa phần cứng và phần mềm cho tính toán song song. Nó cho phép các NVIDIA GPUs thực hiện được các chương trình viết bằng C, C++, Fortran, OpenCL, DirectCompute, và các ngôn ngữ khác. Bởi vì hầu hết các ngôn ngữ đều thiết kế chương trình theo kiểu truyền thống cho một thread tuần tự, CUDA vẫn giữ mô hình này và mở rộng nó với các khái niệm trừu tượng để thực hiện song song. Điều này cho phép các lập trình viên chỉ tập trung vào các vấn đề quan trọng có thể song song được, và làm thế nào để thiết kế một thuật toán song song hiệu quả. Và điều còn lại là chỉ sử dụng một ngôn ngữ quen thuộc thôi. Sau đó, với thiết kế đó, CUDA cho phép phát triển chương trình song song ở mức cao. Chương trình có thể chạy trên hàng chục ngàn thread và hàng trăm nhân xử lý. Một chương trình CUDA được biên dịch có thể thực thi trên bất kì GPU nào, sử dụng tính song song một cách tự động trên các GPU với nhiều nhân xử lý và nhiều luồng.

|  |
| --- |
|  |
| Hình 2.6. Mô hình bộ nhớ GPU trong mô hình lập trình CUDA |

#### *Lập trình song song với mô hình lai trên CPU – GPU*

Hệ thống lai CPU + GPU [[15](#Agu12)] là một hệ thống kết hợp xử lý mạnh mẽ vì CPU và GPU có những thuộc tính kết hợp sử dụng cả hai loại bộ xử lý bổ sung nhau cho phép thực thi nhiều ứng dụng lớn đạt hiệu năng tối ưu. Chip CPU đa nhân đảm trách việc xử lý tuần tự, trong khi đó chip xử lý GPU với hàng ngàn nhân xử lý được thiết kế cho phần tính toán và xử lý song song. Những đoạn mã tuần tự vẫn được thực thi trên CPU, trong khi những phần mã được thiết kế song song lại được thực thi trên GPU. Dựa trên thiết kế phối hợp mạnh mẽ này, một ứng dụng có thể được lập trình song song một cách dễ dàng, mạnh mẽ và tiết kiệm được nhiều thời gian hơn. Hệ thống lai với những nhân GPU song song kết hợp đồng thời với bộ xử lý đa nhân CPU như vậy sẽ là kiến trúc tính toán đầy hứa hẹn trong tương lai giúp tăng tốc nhiều ứng dụng lớn trong tính toán hiệu năng cao.

Với mô hình kết hợp xử lý như vậy, người lập trình có thể sử dụng đúng nhân xử lý trên đúng những đoạn mã cần thiết. Nhân xử lý CPU sử dụng cho việc tối ưu hóa một luồng đơn với thời gian chờ thấp trên đoạn mã tuần tự, và những nhân GPU sử dụng tối ưu hóa tổng hợp băng thông lớn trên những phần song song của mã chương trình. Cách tiếp cận này cho hiệu năng tính toán cao hơn so với việc chỉ sử dụng trên một nhân CPU hoặc chỉ trên các nhân GPU.



Hình 2.7. Hệ thống lai CPU+GPU

Mô hình lập trình đại diện trên GPU như đã nói trên là SIMT. Để lập trình cho mô hình SIMT người lập trình phải xác định chỉ số thread và khéo léo trong việc truy xuất bộ nhớ. Và với lập trình trên mô hình lai, để các tính toán kiểu SPMD trên CPU và GPU phối hợp tối ưu, thì người lập trình cũng cần tính toán sao cho thực thị trên GPU với thời gian tối ưu nhất và dữ liệu đầu vào phải phù hợp trong cả GPU và bộ nhớ CPU.

# Cải tiến thuật toán Ncuts

Chương này sẽ trình bày các vấn đề mà chúng tôi đề xuất giải quyết trong bài toán phân đoạn ảnh Ncuts nhằm cho hiệu năng tính toán cao hơn và khả thi hơn. Thứ nhất, việc chọn số k phân vùng trong thuật toán phụ thuộc nhiều vào người dùng. Nhận thấy tính tự động trong các ứng dụng phân đoạn ảnh cần có phương pháp dự đoán số phân vùng k, chúng tôi đề xuất dự đoán số k phân vùng dựa vào histogram đặc trưng của ảnh. Thứ hai, trong một bước của thuật toán sử dụng phương pháp gom nhóm k-means để gom nhóm trên tập vector riêng tìm được. Vì tập vector riêng là tập dữ liệu thực sẽ có sai số trong quá trình tính toán gom cụm do cấu trúc máy tính và biểu diễn số của máy tính. Chúng tôi đề xuất dùng FCM với mong muốn có thể làm mờ hóa dữ liệu nên có thể chấp nhận sai số để tiên liệu gom cụm dữ liệu tốt hơn. Cuối cùng, quá trình tìm ma trận tương đồng và trị riêng của bài toán thực hiện tuần tự chiếm thời gian thực thi khá lớn trong toàn bộ thuật toán. Chúng tôi đề xuất áp dụng tính toán song song trên GPU cho bước tính toán này với mong muốn cho hiệu năng tính toán tốt hơn trên dữ liệu ảnh lớn.

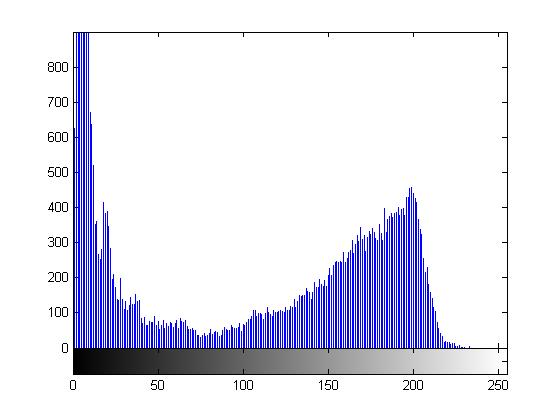
## Xác định số nhóm k cần phân đoạn

Trong hầu hết các bài toán phân đoạn ảnh, việc xác định số nhóm, số thành phần hay số phân vùng cần phân đoạn rất cần thiết và quan trọng hỗ trợ nhiều cho tính ứng dụng của bài toán ứng dụng. Hầu hết số nhóm cần phân đoạn được chọn là biến đầu vào khá quan trọng ở các thuật toán. Thuật toán mà chúng tôi đang xét, thuật toán Ncuts cho phân đoạn ảnh, số nhóm cần phân đoạn là tham số đầu vào cần xác định trước. Trong thuật toán của tác giả, tham số này được xác định dựa vào tính trực quan của ảnh vào, tức là người sử dụng sẽ ước lượng trước số nhóm cần phân đoạn, và cho số nhóm này như dữ liệu vào của bài toán.

Nhận thấy tính tự động trong phân đoạn ảnh của thuật toán là cần thiết trong nhiều bài toán ứng dụng, chúng tôi đã xây dựng thêm bước xác định số nhóm k trong quá trình cài đặt thuật toán. Việc xác định số phân vùng hay số nhóm cho việc phân đoạn ảnh vào dựa trên đặc trưng của từng vùng cần gom nhóm của ảnh. Chính vì vậy, ý tưởng chính của bước này chúng tôi dựa vào lược đồ xám của ảnh histogram xác định số các đỉnh cực trị (cụ thể là các đỉnh cực đại) tương ứng với số nhóm cần phân vùng trong ảnh được minh họa trong ..

σ

δ



Hình 3.1. Lược đồ histogram minh họa cực trị với khoảng cách tương ứng

Việc xác định số các đỉnh cực đại cũng chính là số nhóm cần thiết sao cho thỏa điều kiện công thức 3.1 và 3.2.

 (3.1)

 (3.2)

Trong đó, delta(peak1, peak2) là độ lệch về chiều cao, hay độ chênh lệch về tổng số điểm ảnh trong một thang màu xám của lược đồ histogram giữa hai đỉnh cực đại peak1 và peak2; giá trị dist(peak1, peak2) là khoảng cách giữa hai cực trị peak1 và peak2 trong thang màu xám của lược đồ histogram. Độ lệch delta này phải thỏa điều kiện lớn hơn một giá trị \* và đồng thời độ lệch dist tương ứng phải lớn hơn một giá trị \*. Giá trị \* và \* được ước lượng theo phương pháp học thống kê từ tập ảnh được chọn trong đề tài, với đặc điểm của tập ảnh chọn thực nghiệm được ở chương 4. Cụ thể chúng tôi xác định giá trị \*, \* qua quá trình huấn luyện và kiểm tra trên tập dữ liệu ảnh, dựa trên đặc trưng về mức xám và độ sáng của các vùng ảnh trên tập dữ liệu.

## Gom nhóm bằng FCM

Gom nhóm là bước sau cùng để trích ra phân vùng cuối cùng từ ma trận vector riêng với giá trị thực. Để gom nhóm tác giả [[9](#Shi00)] đề xuất phương pháp k-means ở bước sau cùng trong thuật toán Ncuts. Tuy nhiên vì trong quá trình tính toán trên máy tính có sai số, gom nhóm bằng k-means có thể sẽ có kết quả không tốt. Trong đề tài này, chúng tôi chọn một kỹ thuật gom nhóm FCM thay vì sử dụng thuật toán gom nhóm k-means mà tác giả chọn áp dụng trong bài báo để làm rõ điều nói trên và hoàn toàn có thể sử dụng kỹ thuật gom nhóm khác cho kết quả tối ưu hơn. Phương pháp FCM làm mờ hóa dữ liệu nên có thể chấp nhận sai số tính toán cho quá trình gom cụm tối ưu hơn trên dữ liệu thực, tập vector riêng của bài toán Ncuts. Kết quả biểu diễn ở bước gom nhóm ma trận vector riêng bằng FCM mà chúng tôi thu được sẽ được trình bày ở chương sau.

**Thuật toán 3.1:** Giải thuật gom nhóm cho ma trận vector riêng

Thuật toán 3.1: Giải thuật gom nhóm cho ma trận vector riêng

- Đặt ma trận U ∈ Rnxk là ma trận k vector riêng v1, …, vk theo dạng cột

- Với i = 1, …, n, đặt yi ∈ Rk là vector tương ứng với dòng thứ i của ma trận U.

- Gom nhóm các điểm (yi)i=1,…,n ∈ Rk bằng FCM thành các cụm C1, …, Ck.

- Kết quả output là các cụm A1, …,Ak với Ai = {j/ yj ∈ Ci}

## 

## Tính ma trận tương đồng trên GPU

Ma trận trọng số W của đồ thị G = (V, E) được tính theo công thức (2.8). Với ảnh I có kích thước row × col, đồ thị G được xây dựng với tổng số đỉnh là n=row×col, và tương ứng ma trận trọng số **W** được tính với kích cỡ n×n. Như vậy với ảnh có kích cỡ càng lớn thì kích cỡ ma trận **W** cũng tăng theo đáng kể. Cho nên việc tối ưu thời gian tính **W** sẽ rút ngắn được thời gian trong toàn bộ quá trình phân đoạn ảnh Ncuts là cần thiết.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | wij |  |  |
| Ảnh I |  | Đồ thị G |  | Ma trận affinity W |

Hình 3.2. Hình minh họa đồ thị và ma trận tương đồng của ảnh

Xét theo tính liên thông của các điểm ảnh cho ta đồ thị G tương ứng và hiển nhiên ma trận trọng số **W** là ma trận đối xứng và chứa rất nhiều số 0 được minh họa trong hình 3.2. Để tiết kiệm bộ nhớ, ma trận thưa **W** được lưu dưới dạng COO [[16](#Bel08)]. Lúc này ma trận thưa được lưu thành các mảng với giá trị và những vị trí tương ứng của các phần tử khác 0 của ma trận. Ví dụ dạng COO của ma trận thưa **A** sẽ được lưu thành 3 mảng row, col và val là chỉ số dòng, chỉ số cột và giá trị tương ứng của các phần tử khác 0 của **A**.

|  |  |
| --- | --- |
|  | row = [0 0 0 1 1 2 2 3 3 3]  col = [0 1 3 0 1 2 3 0 2 3]  val = [1 3 5 3 1 1 4 5 4 1] |

Với mỗi điểm ảnh i sẽ có các điểm ảnh liên thông j tương ứng, hay nói cách khác sẽ có wij cho hai điểm ảnh liên thông i và j theo khoảng cách liên thông cho trước r. Như vậy với ảnh vào I(row x col) cần xét (row x col) đỉnh tuần tự nhau sẽ mất nhiều thời gian. Chính vì vậy, ta có thể thực hiện song song mỗi đỉnh i ∈ (row x col) tìm các đỉnh liên thông j và tính trọng số wij. Chúng tôi đề xuất giải thuật 3.2 để tìm ma trận trọng số **W** được song song hóa trên GPU.

**Giải thuật 3.2:** Giải thuật tìm ma trận trọng số W trên GPU

Giải thuật 3.2: Giải thuật tìm ma trận trọng số W trên GPU

* Sao chép ma trận I của ảnh vào từ host (bộ nhớ trên CPU) sang device (bộ nhớ GPU)
* Xác định tổng số phần tử khác 0 của ma trận trọng số **W** tương ứng với ma trận vào I.
* Hàm kernel: với mỗi đỉnh i xác định vị trí đỉnh liên thông j và tính trọng số wij tương ứng.
* Cấp phát bộ nhớ cho ma trận **W** trên GPU.
* Gọi hàm kernel thực thi song song trên GPU với số block và số thread tương ứng với thiết bị hệ thống.
* Lưu kết quả dưới dạng COO trên GPU.

Kết quả thực nghiệm đạt khi tính ma trận trọng số W trên GPU với thời gian khá tốt so với tính trên CPU, kết quả và nhận xét được trình bày ở chương sau.

## Tính trị riêng trên GPU

Thuật toán Ncuts dựa trên phương pháp phân hoạch đồ thị và phân vùng quang phổ như đã trình bày ở chương 2, bài toán chuyển về giải hệ phương trình tuyến tính. Bài toán cần giải quyết vấn đề tìm trị riêng của ma trận trọng số **W**. Dưới góc nhìn của phương pháp phân vùng quang phổ, bài toán đưa về giải quyết vấn đề tìm k vector riêng nhỏ nhất của ma trận Laplace của ma trận trọng số **W**. Vì ma trận trọng số **W** là ma trận thưa đối xứng có kích cỡ tương đối lớn nên thời gian tính vector riêng chiếm khá nhiều thời gian trong toàn bộ quá trình phân đoạn ảnh Ncuts. Đặc biệt với kích thước ảnh càng lớn thì kích cỡ ma trận cũng tăng đáng kể, vì vậy thời gian tìm trị riêng chiếm nhiều thời gian hơn trong trường hợp này. Nhận thấy vấn đề này, với mong muốn rút ngắn thời gian tính trị riêng trên ma trận **W**, chúng tôi đề xuất phương pháp song song hóa tính trị riêng của ma trận **W** trên GPU. Thuật toán chúng tôi chọn nghiên cứu trong đề tài để tìm trị riêng là thuật toán Lanczos. Thuật toán Lanczos là thuật toán hữu hiệu cho việc tìm k trị riêng nhỏ nhất hoặc lớn nhất.

**Thuật toán 3.2:** Thuật toán Lanczos [[17](#Jan02)]

Thuật toán 3.2: Thuật toán Lanczos tính trị riêng và vector riêng

Input: Ma trận đối xứng **A** ∈Rnxn và vector w∈Rn là vector đơn vị chuẩn 2. Ma trận đường chéo đối xứng **T**k có tính chất λ(**T**k) ⊂ λ(**A**).

Output: Trị riêng và vector riêng

Bước 1: Gán giá trị v(1 : n) = 0; β0 = 1; k = 0;

Bước 2: while βk <> 0 do

if k <> 0 then

for i = 1 to n do

t = wi;

wi = vi / βk;

vi = -βk t

end for

end if

v = v + Aw; k = k + 1;

αk = wT v; v = v - αkw; βk= || v ||2

end while

Mỗi bước trong thuật toán Lanczos chủ yếu thực hiện các phép nhân ma trận và vector, các phép gán ở mỗi bước lặp. Chúng tôi cải tiến thuật toán Lanczos được thực hiện song song hóa trên GPU theo giải thuật 3.3 với số thread và block tương ứng vơi thiết bị hệ thống.

**Giải thuật 3.3:** Thuật toán Lanczos tìm trị riêng trên GPU

Giải thuật 3.3: Thuật toán Lanczos tìm trị riêng trên GPU

Input: ma trận **W** trên GPU, số k

Output: ma trận **T** và **V** trên CPU

- Khởi tạo vector v0

- Gọi kernel thực hiện các phép gán

for i = 2 to k

Gọi kernel thực hiện các phép nhân ma trận và vector

Gọi kernel thực hiện các phép nhân vector và vector

Gọi kernel thực hiện các phép gán

end for

- Chép kết quả từ device về host

- Giải phóng bộ nhớ device

# Kết quả và hướng phát triển

## Cài đặt thử nghiệm

### Tập dữ liệu

Tập dữ liệu ảnh[[1]](#footnote-1) (Computer Vision Group – University of Granada) dùng cho cài đặt thử nghiệm đánh giá với nhiều chủ đề và kích cỡ khác nhau chúng tôi chia thành 4 tập dữ liệu. Tập ảnh miscelaneous gồm 15 ảnh vật thể với kích cỡ 128×128 gọi là dữ liệu 1. Tập ảnh miscelaneous gồm 11 ảnh về người và cảnh vật với kích cỡ 256×256 gọi là dữ liệu 2. Tập ảnh miscelaneous gồm 64 ảnh kích cỡ 512×512 với đa dạng chủ đề về người, động vật, vật thể và cảnh vật gọi là tập dữ liệu 3. Tập ảnh miscelaneous gồm 3 ảnh kích cỡ 1024×1024 gọi là tập dữ liệu 4. Tập dữ liệu chủ yếu được định dạng .pgm với ảnh xám và .ppm với ảnh màu.

### Môi trường cài đặt

Cấu hình phần cứng và các phần mềm đề tài áp dụng cài đặt thử nghiệm.

* Cấu hình phần cứng

Bảng 4.1: Cấu hình phần cứng sử dụng trong thực nghiệm

|  |  |
| --- | --- |
| Cấu hình | Chỉ số |
| CPU | Intel Core 2 Quad 2.3GHz |
| RAM | 2 GB |
| GPU | GeForce GT 630 |
| GPU Compute Capability | 2.1 |

* Công cụ phần mềm sử dụng

Bảng 4.2: Một số phần mềm sử dụng

|  |  |
| --- | --- |
| Tên phần mềm | Ghi chú |
| OS | Windows 7 Profesional 32-bit |
| Matlab R2011b |  |
| Microsoft Visual Studio 10.0 |  |
| CUDA Toolkit version 5. | 5NVIDIA Corporation |

Cấu hình các thông số cho môi trường Visual C++ và CUDA được trình bày trong phụ lục A.

### Kết quả cài đặt thực nghiệm

1. ***Xác định số k nhóm phân vùng***

Chúng tôi thử nghiệm xác định số phân vùng bằng histogram được cài đặt trên bộ dữ liệu như mô tả ở mục 4.1.1. Dựa trên đặc trưng của ảnh và dùng lược đồ histogram để xác định số phân vùng của ảnh trên bộ dữ liệu. Bảng 3 thể hiện số phân vùng được xác định bằng phương pháp histogram và số phân vùng được xác định trực quan, trích thể hiện 10 ảnh trong bộ dữ liệu. Dựa vào phương pháp học thống kê trên 4 tập dữ liệu, chúng tôi có thông số σ\* = 100 và δ\* = 50 như đã nói đến ở mục 3.1. Từ bảng cho ta thấy số sai khác với hai cách xác định bằng trực quan và dùng histogram là không đáng kể. Chính vì vậy việc dự đoán số phân vùng là khả thi trong các ứng dụng xử lý ảnh động.

Bảng 4.3: Số phân vùng trong ảnh xác định trực quan và dùng lược đồ histogram

|  |  |  |  |
| --- | --- | --- | --- |
| Tập ảnh | Số ảnh đúng | Số ảnh sai | Số phân vùng sai khác trung bình |
| Dữ liệu 1 | 11 | 4 | 1.5 |
| Dữ liệu 2 | 9 | 2 | 1 |
| Dữ liệu 3 | 57 | 7 | 1.5 |
| Dữ liệu 4 | 3 | 0 | 0 |

1. ***Gom nhóm trên tập vector riêng bằng k-means và FCM***

Sau khi tìm được tập vector riêng U(n x k), thuật toán Ncuts sử dụng phương pháp gom nhóm trên tập vector riêng thành k nhóm, tương ứng với k phân vùng của tập điểm ảnh.

Hình 4.1 và hình 4.2 minh họa dùng k-means và FCM ở bước gom nhóm tập vector riêng Unxk trong thuật toán phân đoạn ảnh Ncuts thành k nhóm. Các nhóm phân vùng không có sai khác đáng kể ở hai thuật toán trên tập dữ liệu.

|  |  |
| --- | --- |
|  |  |
|  | |
| Hình 4.1. Gom nhóm bằng k-means trên tập vector riêng Unxk thành k phân vùng | |

|  |  |
| --- | --- |
|  |  |
|  | |
| Hình 4.2. Gom nhóm bằng FCM trên tập vector riêng Unxk thành k phân vùng | |

1. ***Quá trình song song hóa***

* *Song song hóa quá trình tính ma trận tương đồng* ***W***

Trong phần này, chúng tôi thực thi phần tính ma trận tương đồng trên CPU (chương trình thuật toán Ncuts [[9](#Shi00)]) và thực hiện song song trên GPU. Chương trình thực thi trên tập dữ liệu ảnh với kích thước ảnh tăng dần và đo thời gian thực hiện trên CPU và song song trên GPU. Kết quả đo thời gian ở bảng 4 và đồ thị hình 4.3 cho thấy với kích thước ảnh tăng dần thì thời gian thực hiện trên GPU cũng tăng theo tuyến tính. Trong khi đó thời gian tính toán song song trên GPU tăng không đáng kể với số pixel tăng dần và thời gian thực hiện trên GPU nhỏ hơn thực thi trên CPU đáng kể. Từ bảng 4 ta thấy biểu diễn tính toán ma trận **W** trên CPU dừng lại ở ảnh với kích thước 200×200 trên phần cứng hệ thống như trình bày ở mục 4.6. Trong khi đó, với dạng biểu diễn COO cho ma trận thưa W trên GPU có thể biểu diễn tính toán được với ảnh có kích thước lớn hơn mà trên Matlab CPU biểu diễn tính toán.

Bảng 4.4: Thời gian tìm ma trận tương đồng W trên CPU và song song trên GPU

|  |  |  |
| --- | --- | --- |
| Size | Time CPU | Time GPU |
| 30x30 | 0.025933 | 0.017453 |
| 64x64 | 0.13251 | 0.021704 |
| 128x128 | 0.57453 | 0.25554 |
| 140x140 | 0.66462 | 0.028229 |
| 200x200 | 1.4051 | 0.037773 |
| 256x256 | NA | 0.044946 |
| 512x512 | NA | 0.12403 |
| 1024x1024 | NA | 0.42414 |

(Giá trị NA – tính toán trên CPU không thực thi được với ảnh có kích thước 256×256 trở lên)

|  |
| --- |
| W1.jpg |
| Hình 4.3. Thời gian tính ma trận tương đồng W trên CPU và song song trên GPU |

* *Song song hóa thuật toán Lanczos tìm trị riêng*

Trong phần này, chúng tôi do thời gian thực hiện phần tìm trị riêng của ma trận **W** của tác giả [[9](#Shi00)] chủ yếu dùng hàm eigs() của Matlab và thực hiện tính toán song song thuật toán Lanczos trên GPU. Chúng thôi cài đặt thực thi với ảnh có kích cỡ tăng dần như bảng 5 và cố định với số phân vùng là 3 phân vùng, tương ứng với việc tìm 3 trị riêng và vector riêng của ma trận **W**. Bảng 5 cho thấy thời gian tìm trị riêng song song GPU ít hơn đáng kể so với thời gian thực hiện trên CPU. Và khi kích cỡ ảnh tăng dần thì thơi gian tính toán song song trên GPU tăng không nhiều so với thực hiện trên CPU.

Bảng 4.5: Thời gian tìm trị riêng trên CPU và song song trên GPU (với 3 segment)

|  |  |  |
| --- | --- | --- |
| Size | Time CPU | Time GPU |
| 30x30 | 0.36698 | 0.891987 |
| 64x64 | 0.55629 | 0.957186 |
| 128x128 | 1.9973 | 0.73514 |
| 140x140 | 2.5642 | 0.952911 |
| 200x200 | 6.614 | 1.007127 |
| 256x256 | x | 1.074854 |
| 512x512 | x | 1.48407 |
| 1024x1024 | x | 3.18036 |

|  |
| --- |
| eig_size1.jpg |
| Hình 4.4. Thời gian tìm trị riêng trên CPU và GPU theo kích cỡ ảnh (với 3 segment) |

Chúng tôi cũng cài đặt thực thi đo thời gian thực thi hàm tìm trị riêng eigs() trên CPU và song song Lanczos trên GPU khi số phân vùng tăng dần trên cùng kích cỡ ảnh 128×128. Bảng 6 và đồ thị hình 4.4 cho thấy thời gian tìm trị riêng trên CPU chiếm nhiều hơn song song trên GPU tương ứng, và khi số trị riêng cần tìm nhiều hơn thì thời gian thực thi song song trên GPU cũng tăng ít so với thực thi trên CPU.

Bảng 4.6: Thời gian tìm trị riêng trên CPU và GPU theo số phân vùng (với ảnh có kích cỡ 128x128)

|  |  |  |
| --- | --- | --- |
| Số segment | Time CPU | Time GPU |
| 2 | 2.5111 | 0.76126 |
| 3 | 2.0667 | 0.77996 |
| 4 | 1.9737 | 0.74381 |
| 5 | 2.3868 | 0.79546 |
| 6 | 2.0641 | 0.78656 |
| 7 | 2.1699 | 0.80556 |
| 8 | 2.3376 | 0.83346 |
| 9 | 2.3145 | 0.84626 |
| 10 | 2.7137 | 0.86296 |

|  |
| --- |
| eig_kSegment1.jpg |
| Hình 4.5. Thời gian tìm trị riêng trên CPU và GPU theo số phân vùng  (ảnh 128x128) |

## Kết luận và hướng phát triển

Trong luận văn chúng tôi đã nghiên cứu tổng quan về phân đoạn ảnh dựa trên đồ thị và phân vùng quang phổ bằng phương pháp giải hệ phương trình tìm trị riêng, là đặc trưng chủ yếu của thuật toán Normalized Cuts mà đề tài nghiên cứu. Đồng thời chúng tôi cũng tìm hiểu các phương pháp tính toán song song nhằm áp dụng cải thiện thời gian thực thi của thuật toán Ncuts. Như đã trình bày ở chương trước, với tập dữ liệu có kích cỡ ảnh tăng dần thì thời gian thực thi ngày càng tăng, chủ yếu ở bước tính ma trận trọng số của đồ thị và tính trị riêng của hệ phương trình tuyến tính. Chúng tôi cũng tìm hiểu thuật toán và đưa ra các ý tưởng cải thiện hiệu năng của thuật toán hơn so với yêu cầu ứng dụng hiện nay và sao cho tương ứng với công nghệ hiện nay. Kết quả cài đặt tính toán song song đạt được tương tối tốt hơn so với tính toán tuần tự. Tuy nhiên cũng có một số vấn đề cần tiếp tục giải quyết để có kết quả tính toán tốt hơn.

* **Một số kết quả đạt được**
* Dự đoán số phân vùng dựa vào lược đồ xám của ảnh vào làm thông số đầu vào cho thuật toán Ncuts trong phân đoạn ảnh. Kết quả dự đoán bằng histogram trên đặc trưng của tập ảnh sai khác không đáng kể so với việc dự đoán trực quan và có thể chấp nhận được trong xử lý ảnh tự động.
* Trình bày và cài đặt song song tính ma trận tương đồng W với kích cỡ ảnh vào lớn và song song thuật toán Lanczos tìm trị riêng trên GPU. Kết quả thời gian đo được khá tốt so với tính toán tuần tự trên CPU.
* **Một số vấn đề cần tiếp tục giải quyết và hướng nghiên cứu tiếp theo**
* Tuy việc cài đặt thực thi song song trên GPU cho hiệu năng khá tốt nhưng với dữ liệu số thực lưu trên GPU với kiến trúc CUDA ở dạng dấu chấm động IEEE 754 [[18](#Nat11)] nên có nhiều sai số trong quá trình tính toán. Trong hướng nghiên cứu tiếp theo, chúng tôi sẽ tìm hiểu và cài đặt lại các hàm tính toán trên kiến trúc CUDA trên GPU với dữ liệu số thực lưu dạng dấu chấm dộng sao cho làm tròn với sai số ít nhất hoặc định nghĩa lại dạng kiểu dữ liệu sao cho phù hợp và cho ít sai số hơn. Ví dụ hàm nhân hai số lưu dạng dấu chấm động áp dụng trong các phép nhân ma trận và vector, nhân vector với vector, hàm mũ, hàm căn thức, …
* Trong thuật toán k-way Ncuts, đồ thị ban đầu được phân hoạch thành k’ (k’ < k) đồ thị con tương ứng với k’ phân vùng. Sau đó tiếp tục thực hiện phân vùng trên mỗi k’ phân vùng con này. Nghiên cứu phương pháp thực hiện song song trên mỗi k’ phân vùng dựa trên tiêu chuẩn k-way Ncuts để được k phân vùng cuối cùng.

Tài liệu tham khảo

x

|  |  |
| --- | --- |
| [1] | Priyanka, Swapna Devi Rajesh Dass, "Image Segmentation Techniques," *International Journal on Electronics & Communication Technology*, vol. 3, no. 1, Jan-March 2012. |
| [2] | Abhishek Misal Manjusha Singh, "A survey paper on various visual image segmentation techniques," *International Journal of Computer Science and Management Research*, vol. 2, no. 1, Jan. 2013. |
| [3] | Lei Zhang, David Zhang Bo Peng, "A survey of graph theoretical approaches to image segmentation," *Pattern Recognition*, vol. 46, no. 3, pp. 1020-1038, March 2013. |
| [4] | D.P. Huttenlocher P.F. Felzenszwalb, "Efficient graph based image segmetnation," *International Journal of Computer Vision*, vol. 59, no. 2, pp. 167-181, Sept 2004. |
| [5] | A.G. Constantinides S.H. Kwok, "A Fast Recursive Shortest Spanning Tree for Image Segmentation and Edge Detection," *IEEE TRANS. on Image processing*, vol. 6, no. 2, pp. 328-332, 1997. |
| [6] | S. Wang and J.M. Siskind, "Image Segmentation with Minimum Mean Cut," in *IEEE International Conference on Computer Vision, ICCV 2001*, vol. 1, 2001, pp. 517-524. |
| [7] | Song Wang and J.M. Siskind, "Image Segmentation with Ratio Cut," *Pattern Analysis and Machine Intelligence, IEEE Trans.*, vol. 25, no. 6, pp. 675-690, 2003. |
| [8] | C.T. Zahn, "Graph-theoretic methods for detecting and describing gestalt clusters," *IEEE Transactions on Computing*, vol. 20, pp. 68-86, 1971. |
| [9] | J. Shi and J. Malik, "Normalized Cuts and Image Segmentation," *Pattern Analysis and Machine Intelligence, IEEE Trans.*, vol. 22, no. 8, pp. 888-905, 2000. |
| [10] | Olivier Lezoray and Leo Grady, *Image Processing and Analysis with Graphs- Theory and Practice*.: CRC Press, Jul 3, 2012. |
| [11] | Z. and Wu , R. Leahy, "An optimal graph theoretic approach to data clustering Theory and its application to image segmentation," *Pattern Analysis and Machine Intelligence, IEEE Trans*, vol. 15, no. 11, pp. 1101-1113, Nov 1993. |
| [12] | Ulrike von Luxburg, "A tutorial on Spectral Clustering," 2007. |
| [13] | Soumi Ghosh and Sanjay Kumar Dubey, "Comparative analysis of K-means and Fuzzy C-means algorithms," *International Journal of Advanced Computer Science and Applications*, vol. 4, no. 4, 2013. |
| [14] | NVIDIA Group, "NVIDIA CUDA C Programming Guide version 4.1," NVIDIA Corporation, 2011. |
| [15] | Vázquez F, Garzón EM, Fernández JJ. Agulleiro JI, "Hybrid computing:CPUþGPU co-processinganditsapplicationto," *Ultramicroscopy at SciVerse ScienceDirect*, no. 115, pp. 109-114, Feb 2012. |
| [16] | Nathan Bell and Michael Garland, "Efficient Sparse Matrix-Vector Multiplication on CUDA," NVIDIA Technical Report, 2008. |
| [17] | Jane K. Cullum and Ralph A. Willoughby, *Lanczos Algorithms for Large Symmetric Eigenvalue Computations*, 2nd ed. USA: Society for Industrial and Applied Mathematics Philadelphia, 2002. |
| [18] | Nathan Whitehead and Alex Fit-florea, "Precision and Performance: Floating Point and IEEE 754 Compliance for NVIDIA GPUs," NVidia Technical White Paper, 2011. |

x

1. <http://decsai.ugr.es/cvg/dbimagenes/> [↑](#footnote-ref-1)