ỨNG DỤNG LÝ THUYẾT ĐỒ THỊ ĐỂ TÌM KIẾM TỰ ĐỘNG CÁC VÒNG KHÉP TRONG LƯỚI TRẮC ĐỊA

TS. NGUYỄN NGỌC LÂU
Trường Đại học Bách khoa TP Hồ Chí Minh

Tóm tắt:

Chúng tôi nghiên cứu bài toán tìm vòng khép trong lý thuyết đồ thị và áp dụng vào việc tìm kiếm tự động các vòng khép trong lưới Trắc địa. Công việc này phục vụ cho công tác đánh giá chất lượng trị đo và kiểm tra sai số thô. Chương trình đã lập ra được áp dụng một cách hiệu quả vào phần mềm GNSS Pro xử lý dữ liệu GNSS cạnh ngắn.

1. Giới thiệu

Kiểm tra chất lượng các trị đo, phát hiện và loại những trị đo không đạt yêu cầu trong một mạng lưới trắc địa rất quan trọng. Đây là một yêu cầu bắt buộc trước khi tiến hành bình sai mạng lưới trắc địa. Những trị đo không đạt yêu cầu không được phép tham gia vào việc bình sai và phải được đo lại.

Các trị đo trong lưới trắc địa là chênh cao, góc, độ dài hay phổ biến nhất hiện nay là thành phần baseline GNSS (Global Navigation Satellite Systems). Trong vài nghiên cứu trước đây [1, 2, 3], chúng tôi đã chỉ ra các baseline dù thỏa mãn tất cả các tiêu chuẩn đề ra nhưng vẫn có thể chứa sai số hệ thống. Sai số hệ thống này chỉ có thể phát hiện được khi kiểm tra sai số từ hai vòng khép trở lên. Vì vậy việc xác định các vòng khép và tính sai số khép trong lưới trắc địa là một công cụ hiệu quả nhằm đánh giá chất lượng trị đo và phát hiện để loại bỏ những trị đo có chứa sai số thô.

Khi số trị đo thừa trong lưới càng nhiều thì số vòng khép (độc lập và phụ thuộc) sẽ tăng lên làm gia tăng gánh nặng tính toán. Do đó việc xác định vòng khép cần được thực hiện một cách tự động. Trong bài báo này, chúng tôi nghiên cứu bài toán tìm vòng khép trong lý thuyết đồ thị và áp dụng vào việc xác định vòng khép và tính sai số khép trong mạng lưới GNSS.

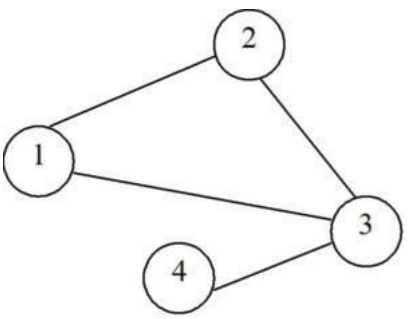
2. Bài toán tìm vòng khép trong lý thuyết đồ thị

Để tìm kiếm tự động và nhanh chóng các vòng khép trong mạng lưới trắc địa, chúng tôi áp dụng một bài toán tương tự đã có trong lý thuyết đồ thị. Bài toán phát biểu như sau:

Cho một đồ thị vô hướng G = (V, E) trong V là tập đỉnh |V| = n và E là tập cạnh |E| = m. Tìm tất cả các vòng khép trong đồ thị G.

Để giải bài toán trên người ta thường áp dụng thuật toán Depth-First Search (DFS). Thuật toán DFS cho phép xây dựng một cây có hướng từ gốc (nút đầu tiên của V). Nếu tồn tại một đường dẫn có hướng trong cây từ v đến w, thì v là "tiền bối" của w và w là "hậu duệ" của v. Để dễ hiểu hơn, chúng tôi sẽ minh họa thuật toán trên một ví dụ sau trong tài liệu [4] *(Xem hình 1)*

Đồ thị đã cho ở hình 1 có V = {1,2,3,4} và E = {(1,2),(1,3),(2,3),(3,4)}. Ta xây dựng ma



Hình 1: Ví dụ về đồ thị vô hướng

Ngày nhận bài: 23/02/2016, ngày chuyển phản biện: 24/02/2016, ngày chấp nhận phản biện 03/3/2016, ngày chấp nhận đăng: 04/3/2016

trận nút kề nút như sau:

$$\begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

Và cấu trúc nút kề cạnh là:

1: 2,3

2: 1,3

3: 1,2,4

4: 3

Nút 4 được gọi là "lá" vì nó chỉ có một cạnh liên kết duy nhất

- DFS bắt đầu bằng cách coi nút 1 là nút hiện tại

- DFS lặp (i là nút hiện tại)

+ Nếu 1 hay nhiều nút có liên kết với nút i chưa "viếng thăm" từ i, gọi j là nút đầu tiên chưa "viếng thăm". Nếu nút j đã được "viếng thăm" bởi DFS (từ nút nào đó khác i), đánh dấu cạnh (i, j) là cạnh "sau", ngược lại đánh dấu cạnh (i, j) là cạnh "cây" và đặt nút i là "cha" của j. Đánh dấu nút j đã được "viếng thăm" trong hồ sơ của nút i. Đặt nút j làm nút hiện tại.

+ Ngược lại (tất cả các nút trong hồ sơ của nút i đã được "viếng thăm"), đặt nút k làm nút hiện tại, trong đó k là "cha" của i.

- Thuật toán sẽ dừng lại khi nút 1 là nút hiện tại (tức là thuật toán trở về nút 1). Nếu tất cả các nút đã được "viếng thăm" thì đồ thị được liên kết. Khi đó mỗi cạnh "sau" (i, j) sẽ xác định một vòng khép. Một vòng khép sẽ bao gồm cạnh "sau" và các cạnh "cây" tạo thành đường dẫn từ j đến i.

Đối với đồ thị đã cho ở hình 1, thuật toán DFS thực hiện như sau:

Nút hiện tại	Cạnh	
1	(1,2)	cây, 1 là cha của 2
2	(2,3)	cây, 2 là cha của 3
3	(3,4)	cây, 3 là cha của 4
4	không có	
3	(3,1)	sau
3	không có	
2	không có	
1	kết thúc	

3. Áp dụng vào việc tìm vòng khép trong lưới trắc địa

Việc tìm ra tất cả các vòng khép trong lưới (kể cả phụ thuộc) là không cần thiết. Trong khi công việc này lại tốn nhiều thời gian và bộ nhớ do phải tìm kiếm và lưu trữ số lượng vòng khép quá lớn trong một mạng lưới. Một số phần mềm chọn cách giới hạn số cạnh trong vòng khép cần tìm. Ví dụ phần mềm TBC (Trimble Business Center) [5] chỉ cho phép tìm sai số khép trong các tam giác. Tuy nhiên điều này sẽ hạn chế khả năng phát hiện sai số thô khi trong lưới có vòng khép độc lập có 4 cạnh trở lên. Phần mềm LGO (Leica Geo Office) [6] cho phép tìm các vòng khép có số cạnh tối đa là 6.

Để cải thiện thuật toán DFS đã nêu ở mục 2, chúng tôi đưa thêm vào thuật toán tham số số cạnh tối đa của vòng khép (max_edgs). Chương trình tìm vòng khép trong lưới trắc địa được lập ra theo sơ đồ khối ở hình 2 *(xem hình 2)*

Theo sơ đồ khối ở hình 2, một lưới trắc địa nhập vào sẽ được dùng để tạo đồ thị G = (V, E). Thuật toán sẽ lặp trên từng cạnh của tập E. Đối với mỗi cạnh, việc tìm vòng khép sẽ bắt đầu lần lượt ở hai điểm của cạnh đó bằng chương trình con Tim_vong_khep. Chương trình này áp dụng thuật toán DFS để dò tìm vòng khép chứa trong tập hợp **path**. Mỗi khi tìm được 1 vòng khép chứa trong **path** có số cạnh nhỏ hơn max_edgs, chương trình sẽ kiểm tra xem vòng khép này đã có trong tập hợp **cycles** chưa? nếu đã có nó bỏ qua. Ngược lại nó sẽ cập nhật vào tập hợp **cycles**. Khi thuật

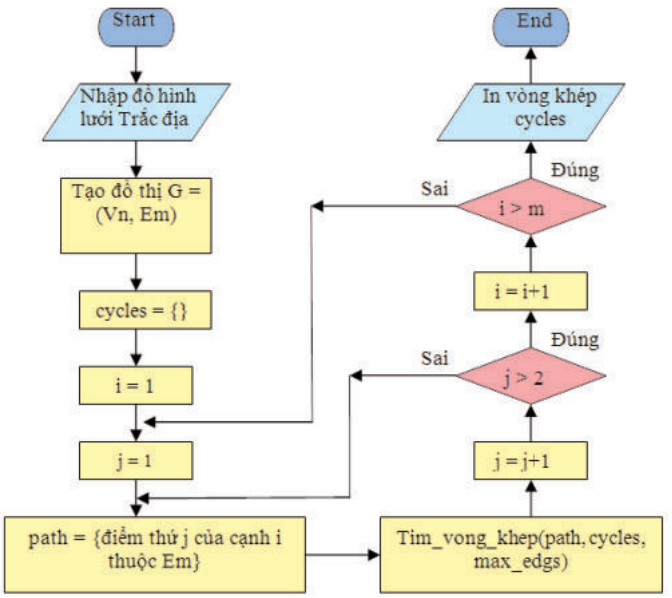
toán kết thúc, toàn bộ vòng khép cần tìm sẽ chứa trong tập hợp **cycles**.

Dựa vào thuật toán nêu trên, chúng tôi viết chương trình all_cycles.exe bằng ngôn ngữ lập trình C++. Ngoài việc tìm ra tất cả các vòng khép có số cạnh nhỏ hơn hoặc bằng max_edgs, chương trình còn tính sai số khép cho các baseline GNSS. Chúng tôi sẽ dùng chương trình này để chạy thử nghiệm trên dữ liệu thực ở mục 4.

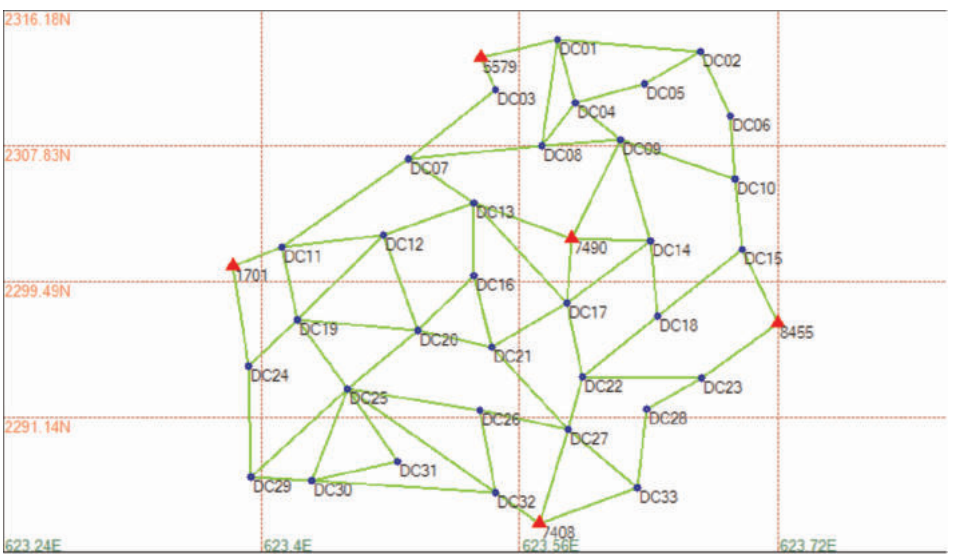
4. Thử nghiệm trên dữ liệu thực

Để thử nghiệm chương trình, chúng tôi dùng một mạng lưới GNSS gồm 38 điểm và 44 cạnh. Trong lưới có nhiều loại vòng khép khác nhau.

Khi cài đặt max_edgs = 3, chương trình tìm được tất cả 14 vòng khép tam giác,



Hình 2: Sơ đồ khối chương trình tìm vòng khép trong lưới trắc địa



Hình 3: Lưới GNSS dùng trong thử nghiệm

đúng như đã đánh số *(xem hình 3)*, và tính các sai số khép theo thành phần X, Y, Z và tổng hợp.

Khi cài đặt max_edges = 4, chương trình tìm được tất cả 32 vòng khép có số cạnh nhỏ hơn hoặc bằng 4. Ngoài 14 vòng khép tam giác đã liệt kê *(xem bảng 1)*, còn có thêm 18 vòng khép tứ giác *(xem bảng 2)*. Trong đó có nhiều vòng khép phụ thuộc. Nếu công việc này làm bằng tay thì rất khó

Bảng 1: Các vòng khép tam giác

STT	Vòng khép	ΔX	ΔY	ΔZ	ΔS	ΔS/S (ppm)
1	DC31-DC25-DC30	+0.0191	-0.0729	+0.0187	0.0776	4.6
2	DC25-DC30-DC29	-0.0386	-0.0127	-0.0142	0.0430	2.4
3	DC25-DC30-DC32	+0.0179	-0.0358	-0.0011	0.0400	1.4
4	DC11-DC12-DC19	-0.0360	-0.0843	+0.0738	0.1177	6.5
5	DC19-DC20-DC25	+0.0202	+0.0183	+0.0143	0.0308	1.7
6	DC19-DC20-DC12	-0.1163	+0.0874	+0.0008	0.1455	6.9
7	DC25-DC32-DC26	-0.0042	-0.0211	+0.0578	0.0617	2.5
8	7408-DC27-DC33	-0.0040	+0.0249	-0.0188	0.0315	1.7
9	DC16-DC20-DC21	+0.0731	-0.1003	-0.0554	0.1359	9.7
10	DC04-DC01-DC08	+0.0389	+0.0023	-0.0841	0.0927	6.6
11	DC04-DC09-DC08	+0.0462	+0.0467	-0.0119	0.0668	5.7
12	7490-DC09-DC14	+0.0065	-0.0080	+0.0171	0.0200	1.1
13	DC13-7490-DC17	-0.0307	+0.1158	+0.0976	0.1545	8.2
14	7490-DC14-DC17	+0.0278	-0.0215	+0.0067	0.0358	2.4

Bảng 2: Các vòng khép tứ giác

STT	Vòng khép	ΔX	ΔY	ΔZ	ΔS	ΔS/S (ppm)
1	DC31-DC25-DC29-DC30	+0.0577	-0.0602	+0.0329	0.0896	4.0
2	DC31-DC25-DC32-DC30	+0.0012	-0.0371	+0.0198	0.0421	1.3
3	DC25-DC30-DC32-DC26	+0.0137	-0.0569	+0.0567	0.0815	2.6
4	DC25-DC29-DC30-DC32	+0.0565	-0.0231	+0.0131	0.0624	1.8
5	1701-DC24-DC19-DC11	-0.0086	-0.0065	-0.0367	0.0383	2.1
6	DC11-DC12-DC20-DC19	+0.0803	-0.1717	+0.0730	0.2031	8.3
7	DC11-DC12-DC13-DC07	-0.0048	-0.0013	+0.0012	0.0051	0.2
8	DC19-DC24-DC29-DC25	-0.0267	+0.0224	-0.0581	0.0678	2.8
9	DC19-DC12-DC20-DC25	+0.1365	-0.0691	+0.0135	0.1536	6.3
10	DC32-DC26-DC27-7408	-0.0134	-0.0157	-0.0377	0.0430	2.1
11	DC12-DC13-DC16-DC20	-0.0120	+0.0479	-0.0618	0.0791	3.7
12	DC16-DC21-DC17-DC13	+0.0299	+0.0338	+0.0284	0.0533	2.3
13	DC21-DC27-DC22-DC17	-0.0127	-0.0099	-0.0314	0.0353	1.7
14	DC04-DC01-DC08-DC09	-0.0073	-0.0444	-0.0722	0.0851	4.5
15	DC04-DC05-DC02-DC01	-0.0300	+0.0218	-0.0480	0.0607	2.9
16	7490-DC09-DC14-DC17	+0.0343	-0.0295	+0.0238	0.0511	2.1
17	DC13-7490-DC14-DC17	-0.0029	+0.0943	+0.1043	0.1406	5.4
18	DC14-DC17-DC22-DC18	-0.0145	-0.0086	-0.0422	0.0454	2.1

có thể liệt kê và tính toán sai số khép một cách đầy đủ.

Lần lượt cho max_edgs = 5, 6, 7 và 8, chương trình tìm được 55, 93, 159, và 271 vòng khép một cách tương ứng. Ta thấy số vòng khép tăng rất nhanh theo số cạnh tối đa của vòng khép. Vì vậy khi max_edgs càng lớn thì chương trình sẽ chạy chậm hơn và chiếm dụng bộ nhớ máy tính nhiều hơn. Hiện tại chúng tôi cài đặt max_edgs = 8 là phù hợp với cấu hình máy tính.

5. Kết luận

Để phục vụ cho việc đánh giá chất lượng trị đo và kiểm tra sai số thô trong các mạng lưới trắc địa, việc tìm kiếm các vòng khép để tính sai số khép là rất cần thiết. Công việc này nếu làm bằng tay sẽ dễ thiếu sót và mất nhiều thời gian, đặc biệt đối với những mạng lưới lớn.

Chúng tôi đã nghiên cứu áp dụng bài toán tìm vòng khép trong lý thuyết đồ thị vào việc tìm vòng khép trong các mạng lưới Trắc địa một cách tự động. Ngoài việc áp dụng thuật toán DFS cho phép tìm kiếm nhanh các vòng khép, chúng tôi còn bổ sung thêm tham số số cạnh tối đa của vòng khép (max_edgs) để hạn chế bớt các vòng khép lớn phụ thuộc không cần thiết. Điều này giúp chương trình chạy nhanh hơn và không chiếm dụng nhiều bộ nhớ của máy tính.

Chúng tôi đã áp dụng một cách hiệu quả chương trình C++ all_cycles.exe vào phần mềm **GNSS Pro**. Đây là phần mềm xử lý GNSS cạnh ngắn nằm trong đề tài nghiên cứu khoa học do Viện Khoa học Đo đạc và Bản đồ làm chủ trì từ 2013 đến 2015 [2].❍

Tài liệu tham khảo

[1]. Nguyễn Ngọc Lâu, Nguyễn Thị Thanh Hương và Dương Tuấn Việt, (2014), "Các nguyên tắc thiết kế phần mềm GNSS PRO xử lý dữ liệu GNSS", Tuyển tập báo cáo HNKH & CN "Trắc địa và Bản đồ vì hội nhập quốc tế", Hà Nội 7-2014, pp. 37-45.

[2]. Nguyễn Ngọc Lâu, Nguyễn Thị Thanh Hương và Hà Minh Hòa, (2015), "Ứng dụng phần mềm GNSS-Pro xử lý các mạng lưới GNSS cạnh ngắn tại Việt Nam", Hội nghị Khoa học và Công nghệ lần thứ 14 tại Đại học Bách khoa TPHCM 10/2015.

[3]. Lau Ngoc Nguyen, Viet Tuan Duong and Richard Coleman, (2015), "Validation of GNSS processing results from some commercial software packages under un-advantageous conditions, submitted for Journal of Earth Sciences and Environment, Vietnam.

[4]. Jonathan F Bard, "Cycles in an undirected graph", https://www.me.utexas.edu/~bard

[5]. Trimble, (2006), "Trimble Business Center software technical notes".

[6]. Leica, (2010), "Leica Geo Office software Datasheet".❍

Summary

Application of graph theory for automatically searching polygons in geodetic networks

Dr. Nguyen Ngoc Lau, Hochiminh City University of Technology

We research on problem of finding cycles in an undirected graph and apply to automatically searching polygons in a geodetic network. This work serves for quality estimation of measurements and detection of gross errors. The result source code has been effectively implemented to GNSS Pro software package.❍