The Journal of The Institute of Internet, Broadcasting and Communication (IIBC)
Vol. 22, No. 6, pp.91-97, Dec. 31, 2022. pISSN 2289-0238, eISSN 2289-0246

https://doi.org/10.7236/JIIBC.2022.22.6.91
JIIBC 2022-6-14

방향과 무 방향 일반 그래프의 최대 사이클 검출 알고리즘

Algorithm for Maximum Cycle Detection of Directed and Undirected General Graphs

이상운*

Sang-Un Lee*

요 약 사이클 검출 문제에 대해, 단일 출발(SS)을 갖는 단일 연결 리스트(SLL)에 한해 $O(n)$ 복잡도의 거북이와 토끼 경주법(HTA)이 제안되었으며, 다중 출발지-다중 종착지, 다중 분기(MSMDMB)를 갖는 일반 그래프에 대해서는 빠른 방법이 알려져 있지 않고 있다. 본 논문에서는 MSMDMB를 갖는 주어진 무 방향과 방향 그래프의 최대 사이클을 선형시간 복잡도로 검출할 수 있는 방법을 제안하였다. 제안된 방법은 주어진 원 그래프 G에는 사이클 형성 조건을 충족시키지 못하는 다수의 정점(또는 노드)가 존재한다는 사실에 기반하여 이들 정점(또는 노드)들을 제거한 축소된 그래프 G'를 얻었다. 이 축소된 그래프에 대해 선형시간 복잡도인 선형탐색으로 사이클 집합 C와 사이클 길이 λ를 찾았다. 제안된 알고리즘을 실험 데이터에 적용한 결과 모든 데이터들에 대해 최대 사이클을 찾을 수 있음을 보였다.

Abstract There is hare and tortoise racing algorithm(HTA) for single-source(SS) singly linked list(SLL) with $O(n)$ time complexity. But the fast method is unknown for general graph with multi-source, multi-destination, and multi-branch(MSMDMB). This paper suggests linear time cycle detection algorithm for given undirected and digraph with MSMDMB. The proposed method reduced the given graph G contained with unnecessary vertices(or nodes) to cycle into reduced graph G' with only necessary vertices(or nodes) to cycle based on the condition of cycle formation. For the reduced graph G', we can be find the cycle set C and cycle length λ using linear search within linear time. As a result of experiment data, the proposed algorithm can be obtained the cycle for whole data.

Key Words : Cycle, Cycle length, Hare and Tortoise, Stack, Reduction

Ⅰ. 서 론

주어진 $G=(V,E), n=|V|, m=|E|$의 무방향 그래프(undirected graph) 또는 $G=(N,A), n=|N|, m=|A|$의 방향 그래프(digraph) 에서 사이클이 존재하는지, 만약 존재한다면 사이클 길이 λ와 사이클 집합 C를 찾는 문제를 사이클 검출 문제(cycle detection problem, CDP)라 한다.[1]

CDP는 의사난수 생성기(pseudorandom number generator), 이산대수(discrete logarithm), 암호(cryptographic), 컴퓨터 프로그램의 유한 루프, 셀 구조 자동화(cellular automation)에서의 주기적 형성

*정회원, 강릉원주대학교 과학기술대학 멀티미디어공학과
접수일자 2022년 3월 9일, 수정완료 2022년 11월 9일
게재확정일자 2022년 12월 9일

Received: 9 March, 2022 / Revised: 9 November, 2022 / Accepted: 9 December, 2022
*Corresponding Author: sulee@gwnu.ac.kr
Dept. of Multimedia Eng., Gangneung-Wonju National University, Korea

(periodic configuration), 연결리스트의 형상분석 등에 응용되고 있다.[2]

사이클을 검출하는 대표적인 알고리즘으로는 Floyd[3]의 $O(n)$ 복잡도인 2개 포인터를 사용하는 거북이와 토끼 알고리즘(tortoise and hare algorithm, THA), Brent[4]의 거북이 순간이동 알고리즘(teleporting turtle algorithm, TTA), Nivasch[2]의 스택 알고리즘(stack algorithm, SA) 등이 제안되었으며, 이외에도 인접리스트(adjacency list)를 사용하는 수행 복잡도 $O(m+n)$의 깊이우선 탐색 법(depth-first search, DFS)[5]과 위상정렬(topological order algorithm, TOA)[6]도 있다.

이외에도 방향 그래프에 대한 사이클 검출 방법으로, Rungta et al.[7]의 연결 리스트 이용 방법, Pandy와 Chawla[8]의 역방향 초기하 그래프(backward directed hypergraph, BDH)에 대한 $O(n^2)$ 복잡도 방법, Wu et al.[9]의 방향 그래프에 대한 모든 수용 가능한 사이클 검출 방법 등이 있다. 또한, 대용량의 희소 그래프(sparse graph)d 대해 Rocha와 Thatte[10]은 진 약수 함수(proper divisor function)를 적용하는 방법을 제안하였다.

가장 단순한 알고리즘인 THA는 동일 출발지점에서 시작하여 거북이는 $i,(i=1,2,\cdots,n)$보폭으로 토끼는 $2i$ 보폭으로 경주하면서 서로 만나는 지점이 존재하면 사이클이 존재한다고 한다. THA는 단일 연결 리스트(singly linked list, SLL)에 특화된 알고리즘이다. TTA는 THA와 유사하지만 2^i와 2^{i+1}의 보폭으로 경주하는 방식으로 THA보다 복잡하지만 THA에 비해 36% 정도 빠른 방법으로 알려져 있다. 무방향이나 방향 그래프에 대해서는 SA, DFS나 TOA를 적용할 수 있다. DFS나 TOA의 경우 방향그래프에 대해서는 $O(m+n)$복잡도로 위상정렬 시켜 사이클을 탐색할 수 있다.

그러나 이들 알고리즘 모두 사이클 유무만을 판정하는데 초점을 맞추고 있으며, 우리가 찾고자 하는 사이클 길이 λ와 사이클 집합 C를 찾으려면 토끼와 거북이가 만난 지점에서 THA를 적용하여 토끼는 멈추어 있고, 거북이만 사이클을 따라 한 바퀴 돌면서 걸음 수 λ를 계산하고, 각 보폭에 대한 노드를 저장해야 한다.

만약, 다중 출발-다중 목적지(multi-source multi-destination, MSMC)을 갖는 일반 그래프의 경우 THA를 다중 출발지 횟수로 반복 수행해야만 하는 번거로움이 있다.

본 논문에서는 단일 출발(single-source, SS)을 갖는 SLL 뿐 아니라 MSMC의 무방향과 방향 일반 그래프에 대해 $O(m+n)$수행 복잡도를 갖는 사이클 검출 알고리즘을 제안한다. 2장에서는 CDP의 개념과 관련된 알고리즘들을 고찰해 본다. 3장에서는 CDP에 대해 $O(m+n)$ 복잡도의 사이클 알고리즘(cycle detection algorithm, MCDA)을 제안한다. 4장에서는 다양한 실험 데이터들에 대해 제안된 알고리즘을 적용하여 알고리즘 적합성 여부를 평가해 본다.

II. 관련 연구와 문제점

임의의 함수 f는 유한집합 S와 사상되며 초기치 $x_0 \in S$를 가지며 반복되는 함수 값의 순서는 식 (1)로 표현된다.[2]

$$x_0, x_1 = f(x_0), x_2 = f(x_1), \cdots, x_i = f(x_{i-1}), \cdots \quad (1)$$

Floyd[3]의 THA에서는 토끼와 거북이가 x_0에서 출발하며, 토끼는 x_{2i}, 거북이는 x_i의 보폭으로 움직인다. 여기서 $i=1,2,\cdots,n$이다. 만약, $x_i = x_{2i}$이면 사이클이 존재한다고 판정한다. 반면에, Brent[4]의 TTA에서는 거북이는 x_{2^i}, 토끼는 $x_{2^{i+1}}$에 위치시키고 내부 반복의 각각에서 거북이는 2^n, 토끼는 2^n+i에 존재한다. 일단 토끼가 2^{n+1}에 도달하면 거북이를 이 위치로 순간이동 시킨다.

Nivasch[2]의 SA는 (x_i,i) 쌍의 스택을 준비하고, 각 스텝 j에서 $x_i > x_j$이면 스택에서 모든 (x_i,i)을 pop하면서 $x_i = x_j$를 발견하면 사이클이 존재하는 경우로 사이클 길이 $\lambda = j-i$를 구한다. 그렇지 않으면 스택의 top에 (x_i,i)를 push한다.

이들 방법의 특징은 λ는 구할 수 있지만 사이클을 형성하는 집합 C의 노드들을 알지 못한다는 점이다. 또한 다중 출발지-다중 종착지, 다중 분기(multi-branch, MB)인 MSMDMB의 일반 그래프에 적용하는데도 한계점이 있다.

그림 1의 G_1 단일 연결 리스트(SLL) 사례로 사이클을 찾아보자. G_1은 Codingfreak[11]에서 인용되었다.

$x_i, i=1,2,\cdots,n$에 대해 μ를 값 x_i의 일련의 순서 내에서 값 x_μ가 다시 나타나는 값의 인덱스로 정의한다. 즉, μ는 사이클로 진입하는 지점을 의미하며, 사이클의 길이를 λ로 정의한다.

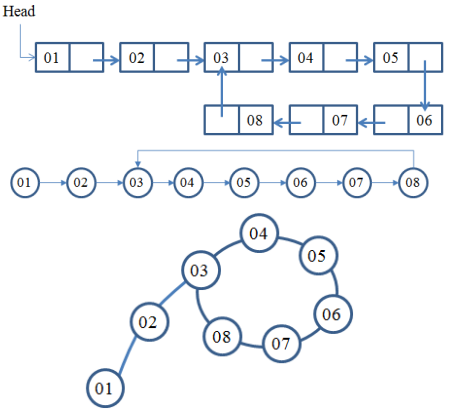


그림 1. G_1 단일 연결 리스트의 사이클 검출
Fig. 1. Cycle detection for G_1 singly linked list

SLL에 대한 사이클은 이산대수 $a^b(\mathrm{mod}\, n)$에 대한 사이클을 찾는 Pollard's rho(ρ) 알고리즘에서 아이디어를 얻었으며, 사이클을 찾는 방법은 Floyd's의 THA를 일반적으로 적용하여 $O(n)$의 선형시간 복잡도로 사이클 존재 유무만을 판정한다. 그림 1의 예제에 대해 THA를 수행한 결과는 표 1에 제시되어 있다.

표 1. 단일 연결 리스트에 관한 THMCDA
Table 1. THMCDA for singly linked list

	출발점	1st Round						2nd Round					
		1	2	3	4	5	6	7	8	9	10	11	12
토끼	01	03	05	07	03	05	07	07	07	07	07	07	07
거북이	01	02	03	04	05	06	07	08	03	04	05	06	07
판정	-	-	-	-	-	-	cycle	λ					

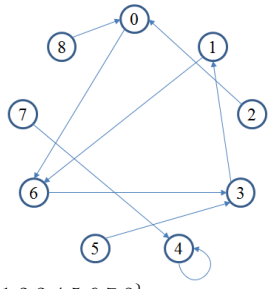
THA는 첫 번째 라운드에서는 사이클 존재 유무만을 판정하며, λ와 사이클 집합 C를 구하기 위해서는 두 번째 라운드까지 수행해야만 한다. 이 방법은 단일 출발지점인 경우에 한정되며, 분기점이 없어야만 한다. 즉 SLL은 Single-source, without branch인 경우에 한정되며 방향 그래프로 생각할 수 있다.

방향 그래프 $G=(N,A)$는 노드(node) 집합 $n_i \in N$과 두 노드 n_i, n_j간 방향성을 가진 간선 순서쌍인 호(arc) 집합 A, $a=w(n_i,n_j)\in A$로 구성되어 있으며, $m=|N|$, $n=|A|$이다.

그림 2의 G_2 방향 그래프 사례를 대상으로 사이클을 검출해 보자. G_2는 Wikipedia[12]에서 인용되었다.

여기서는 출발지점(유입차수=0)은 {7,8,2,5}로 4개로 다중 출발이며, 사이클 C={1,6,3}이다. 따라서 이 사이클을 찾으려면 노드 7에서 시작하면 실패한다. 결국, THA로 정확한 해를 구하기 위해서는 {7,8,2,5}의 모든 출발지에 대해 THA를 적용해야만 한다. 따라서 단일 출발지에 비해 수행횟수는 출발 노드 수 배수로 증가한다.

x	$f(x)$
0	6
1	6
2	0
3	1
4	4
5	3
6	3
7	4
8	0



$S=\{0,1,2,3,4,5,6,7,8\}$
cycle = {1,3,6}

그림 2. G_2 방향 그래프의 사이클 검출
Fig. 2. Cycle detection for G_2 Digraph

따라서 3장에서는 다중 출발지-다중 종착지, 다중 분기를 갖는 무 방향과 방향 그래프 모두에서 정확한 사이클 집합 C를 구할 수 있는 $O(m+n)$수행 복잡도 알고리즘을 제안한다.

Ⅲ. 최대 사이클 검출 알고리즘

n개의 정점(노드)들 사이에 사이클(순환)이 형성되려면 그림 3과 같이 모든 노드(정점)의 차수는 2이며, 무방향 그래프인 경우는 $\forall_i, d(u_i)=2$, 방향 그래프인 경우는 유입과 유출 차수가 각각 1인 $\forall_i, d^-(u_i)=1$ and $d^+(u_i)=1$로 연결되어야 만 한다. 이러한 사이클이 갖는 기본 속성을 반영하여 본 장에서는 방향과 무방향 그래프에 대해 가능한 최대 사이클을 찾는 알고리즘을 제안한다.

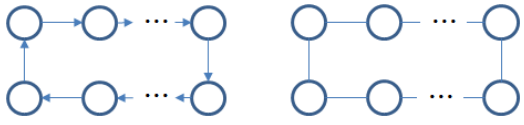


그림 3. 방향과 무방향 그래프의 사이클
Fig. 3. Cycle for digraph and undirected graph

제안된 방법은 주어진 그래프 G에서 사이클이 존재할 가능성이 있는 정점(또는 노드)들만을 추출한 축소된 그래프 G'를 대상으로 한다. 즉, 주어진 무방향 그래프 $G=(V,E)$에 대해서는 사이클이 존재하지 않는 $d_G(u)=1$인 정점들과 $u=f(u)$를 삭제하며, 방향 그래프 $G=(N,A)$에 대해서는 $d^+(u)=0$와 $d^-(u)=0$인 노드들($\forall\ d(u)=d^+$ or $\forall\ d(u)=d^-$)을 삭제한다. 이 방법이 기존의 알고리즘들과 가장 큰 차별성이다.

제안된 알고리즘은 주어진 그래프에서 가능한 최대의 사이클만을 찾는 방법으로 최대(maximum) 사이클 검출 알고리즘(MCDA)이라 하며, 다음과 같이 수행된다.

[방향 그래프]

$G=(N,A)$

$d^-(u)$: u 노드의 유입 차수

$d^+(u)$: u 노드의 유출 차수

- $x=f(x),(u,v),u=v$인 자신에서 유출하여 자신으로 유입되는 궤환 호를 갖는 노드 u와 u에 부속된 호 삭제.
- $d^+(u)=0$ or $d^-(u)=0$인 노드 u와 u에 부속된 호 삭제
- 최 외곽 노드들을 연결한 가장 긴 사이클 형성, 사이클 내부의 3 노드 간 내부 호 삭제

[무방향 그래프]

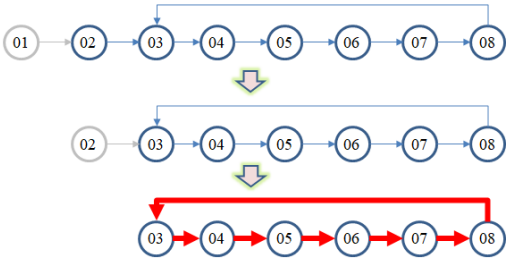
$G=(V,E)$

노드 i의 차수 $d(u)$계산.

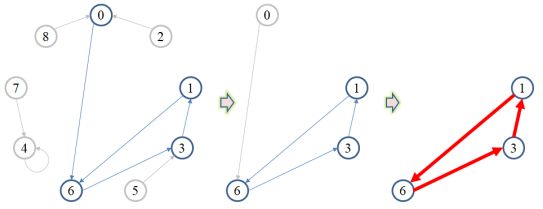
$d(u)=1$인 노드 u와 u에 부속된 간선 삭제.

- 최 외곽 노드들을 연결한 가장 긴 사이클 형성
- 내부의 미 연결된 노드들로 간선이 존재하는 인접 노드 $\{u,v\}$에 대해 $\{u,v\}$ 간선 삭제, 내부 노드들로 사이클 확장.

그림 1의 G_1 SLL 예제와 그림 2의 G_2 방향 그래프에 대해 MCDA를 적용한 결과는 그림 4에 제시되어 있다.



(a) MCDA for G_1 graph



(b) MCDA for G_2 graph

그림 4. SLL과 방향 그래프에 대한 MCDA
Fig. 4. MCDA for SLL and Digraph

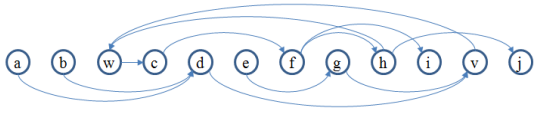
(a)에서는 $d^+(u)=0$과 $d^-(u)=0$인 노드 01, 02를 차례로 삭제하면 축소된 방향 그래프 G'={03,04,05,06,07,08}에 대한 사이클 C={03,04,05,06,07,08}을 얻는다.

(b)에서는 $(u,v),u=v$인 '4' 노드가 먼저 삭제되고, 다음으로 $d^-(u)=0$인 {2,5,7,8} 노드가 삭제되었으며, 남은 축소된 방향 그래프 G'에서 $d^-(u)=0$인 '0' 노드가 삭제되어 C={1,6,3}을 얻었다.

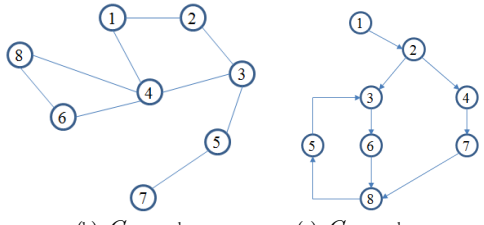
제안된 알고리즘은 1차적으로 각 노드의 차수를 계산하면서 불필요한 정점 또는 노드를 삭제하여 축소된 그래프 G'를 대상으로 최대 사이클을 찾는 방법으로 수행 복잡도는 $O(m+n)$이다. 이는 DFS와 TOA와 동일한 수행 복잡도를 갖는다. 다만 SLL에 대해서는 THA의 $O(n)$에 비해 보다 복잡함을 알 수 있다. 그러나 THA를 2 라운드 수행하는 방법 보다는 수행횟수를 크게 단축시키면서 사이클 집합 C를 찾을 수 있는 장점이 있다.

Ⅳ. 적용 및 결과 분석

본 장에서는 그림 5의 8개 데이터에 대해 제안된 MCDA를 적용하여 본다.



(a) G_3 graph



(b) G_4 graph (c) G_5 graph

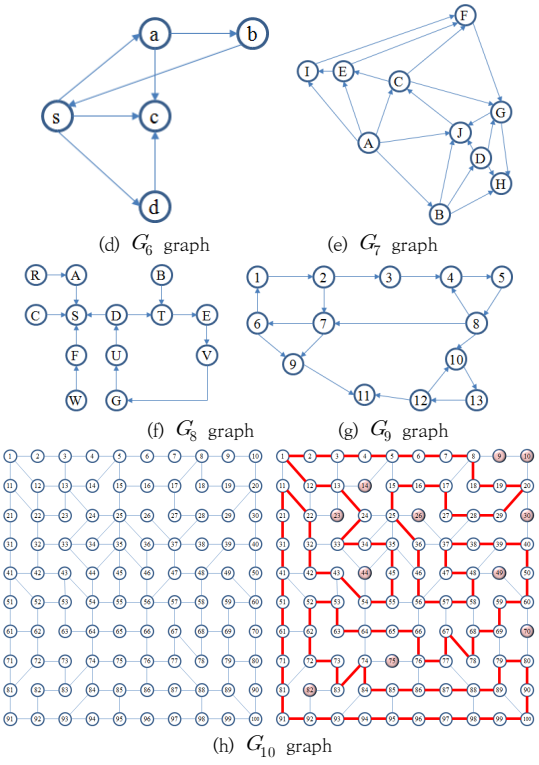
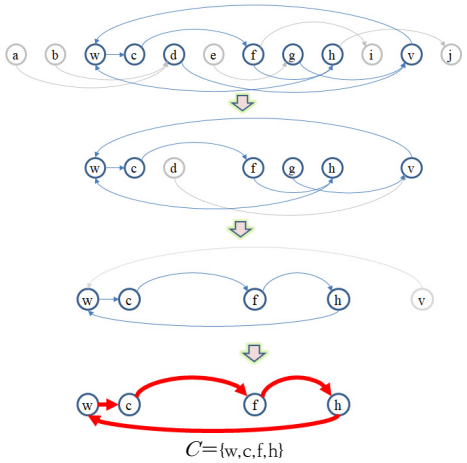
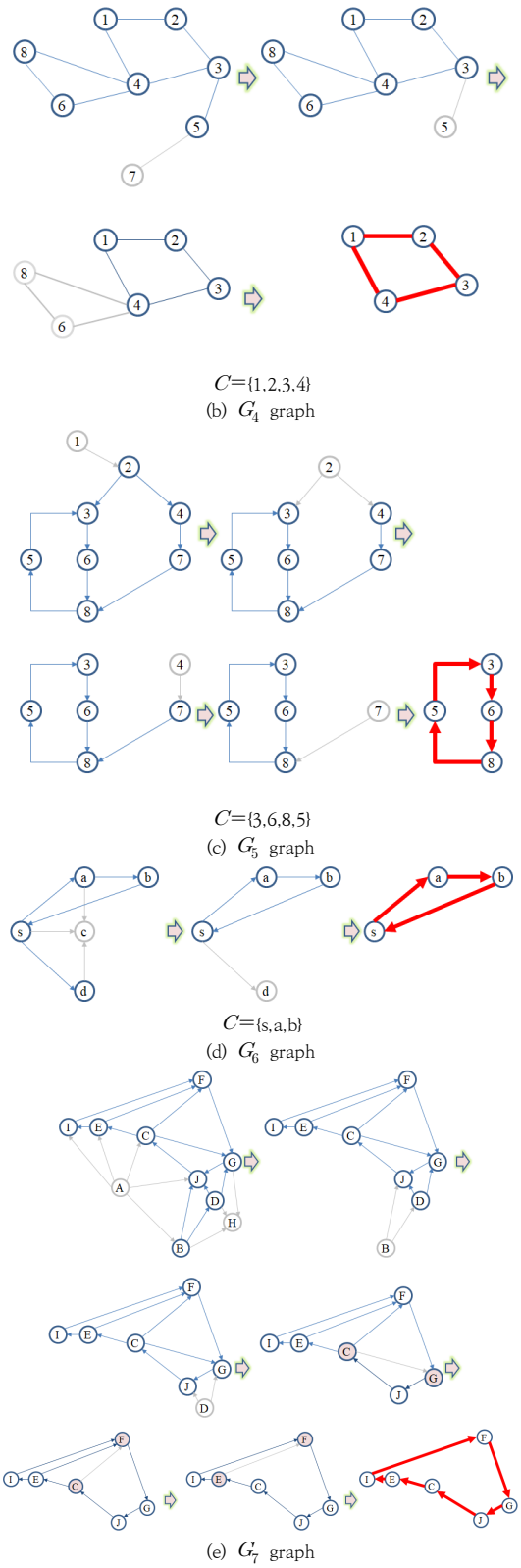


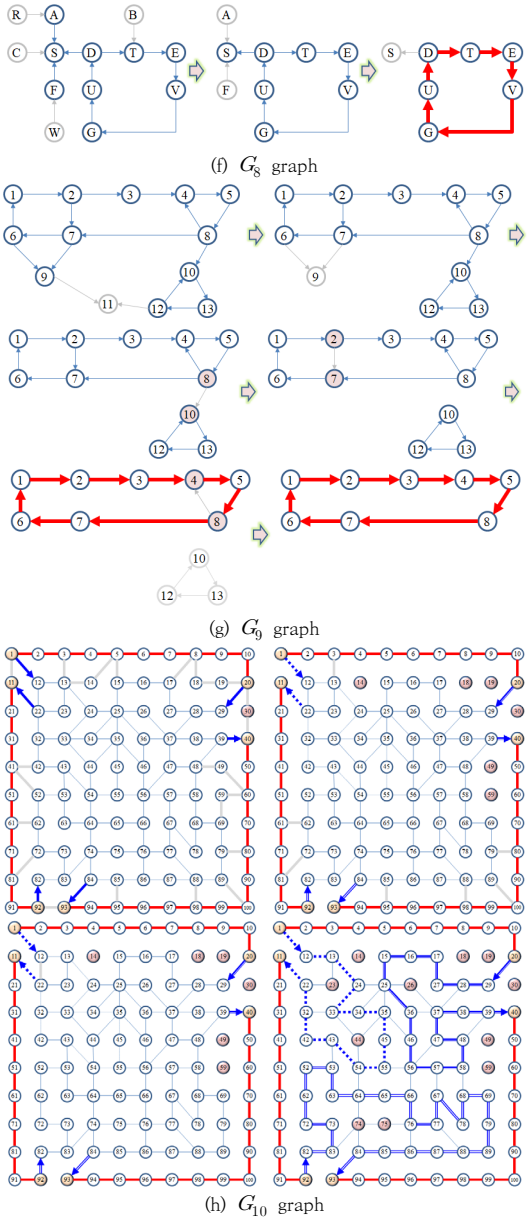
그림 5. 실험 데이터
Fig. 5. Experimental data

G_3은 Haeupler et al.[13]에서, G_4는 Safar et al.[14]에서, G_5는 Boukerche와 Tropper[15]에서, G_6은 Panigrahi[16]에서, G_7, G_8, G_9, G_{10}은 인터넷의 사이클 검출 이미지로 부터 인용되었다.

그림 5의 실험 데이터에 대해 제안된 MCDA를 적용한 결과는 그림 6에 제시되어 있다.







(f) G_8 graph

(g) G_9 graph

(h) G_{10} graph

그림 6. 실험 데이터에 대한 MCDA
Fig. 6. MCDA for experimental data

그림 5의 실험 데이터에 대해 제안된 MCDA를 적용하여 그림 6으로 얻은 최대 사이클 집합 C는 표 2에 요약되어 있다. 여기서는 제안된 MCDA가 최대 사이클을 찾는지 여부를 검증할 수 있도록 원 그래프에서 내포하고 있는 모든 사이클도 제시하였다. 표에서 { }는 집합을, ( )는 호를, →는 순서를 의미한다. 표에서는 G_{10}은 복잡성으로 인해 제외하였다. G_{10}은 100개 노드 중 89개 노드로 연결된 사이클을 얻는데 성공하였다.

표 2. MCDA의 성능
Table 2. Performance of MCDA

그래프 G			MCDA for G'			
문제	내포된 사이클	n	노드와 호(간선) 삭제		n'	C
			삭제 순서	횟수		
G_1	{03,04,05,06,07,08}	8	01 → 02	2	6	{03,04,05,06,07,08}
G_2	{1,6,3}	8	2,8,7,4,5 → 0	2	3	{1,6,3}
G_3	{w,c,f,h}	12	a,b,e,i,j → d → g → v	4	4	{w,c,f,h}
G_4	{4,6,8} {1,2,3,4}	8	7 → 5 → 6,8	3	4	{1,2,3,4}
G_5	{3,6,8,5}	8	1 → 2 → 4 → 7	4	4	{3,6,8,5}
G_6	{s,a,b}	5	c → d	2	3	{s,a,b}
G_7	{G,J,C}, {F,G,J,C}, {F,G,J,C,E}, {F,G,J,C,E}	10	A,H → B → D → (C,G)→(C,F)→ (E,F)	6	5	{F,G,J,C,E}
G_8	{D,T,E,V,G,U}	13	R,C,W,B → F → S	3	6	{D,T,E,V,G,U}
G_9	{1,2,3,4,5,8,7,6}	13	11 → 9 → (8,10) → (2,7) → (4,8)	5	8	{1,2,3,4,5,8,7,6}

제안된 알고리즘은 사이클 형성 조건을 충족시키지 못하는 불필요한 정점(또는 노드)와 간선(또는 호)를 먼저 삭제하여 축소된 그래프 G'의 정점(또는 노드) 개수 n'는 정확히 $|C|$개가 됨을 알 수 있다. 또한, G를 G'로 축소시키는데도 수행횟수는 적게는 2회에서 최대 6회로 원 그래프 G의 정점(또는 노드) 개수 n에 비해 평균 37% 정도만을 수행하면 됨을 알 수 있다.

V. 결 론

본 논문에서는 방향과 무방향 그래프에 대해 최대 사이클을 검출할 수 있는 수행 복잡도 $O(m+n)$의 선형시간 알고리즘을 제안하였다.

기존에는 단일 출발(SS)을 갖는 SLL에 한해 $O(n)$ 복잡도의 THA가 제안되었다. 또한, 다중 출발지-다중 종착지, 다중 분기(MSMD)를 갖는 일반 그래프에 대해서는 빠른 방법이 알려져 있지 않고 있다.

이러한 이유로 인해, 본 논문에서는 MSMD의 일반 그래프에 적용할 수 있는 방법을 제안하였다. 주어진 그래프 G는 사이클 형성 조건을 충족시키지 못하는 다수의

정점(또는 노드)를 포함하고 있다는 사실에 착안하여, 제안된 알고리즘은 1차적으로 사이클 미형성 정점(또는 노드)들을 제거한 축소된 그래프 G'를 $O(m+n)$수행 복잡도로 얻었다. 이 축소된 그래프에 대해 제안된 알고리즘은 선형시간 복잡도로 임의의 정점(또는 노드)부터 순차적 탐색 기법을 적용하여도 $O(n)$ 수행 복잡도로 쉽게 찾을 수 있는 장점을 갖고 있다.

제안된 알고리즘은 CDP를 다루는 의사난수 생성기, 이산대수, 암호, 컴퓨터 프로그램의 유한 루프, 셀 구조 자동화에서의 주기적 형성, 연결리스트의 형상분석 등 분야에서 유용하게 활용될 수 있을 것이다.

References

[1] R. Sedgewick, T. G. Szymanski, and A. C. C. Yao, "The Complexity of Finding Cycles in Periodic Functions," SIAM Journal on Computing, Vol. 11, No. 2, pp. 376-390, May 1982, https://doi.org/10.1137 /0211030

[2] G. Nivasch, "Cycle Detection using a Stack", Information Processing Letters, Vol. 90, No. 3, pp. 135-140, May 2004, https://doi.org/10.1016/j.ipl.2004.01.016

[3] R. W. Floyd, "Non-Deterministic Algorithms", Journal of ACM, Vol. 14, No. 4, pp. 636-644, Oct. 1967, https://doi.org/10.1145/321420.321422

[4] R. P. Brent, "An Improved Monte Carlo Factorization Algorithm", BIT Numerical Mathematics, Vol. 20, No. 2, pp. 176-184, Jun. 1980.

[5] S. S. Skiena, "The Algorithm Design Manual" , 2ed., Springer-Verlag, 2012.

[6] R. Liu, "A Low Complexity Topological Sorting Algorithm for Directed Acyclic Graph", International Journal of Machine Learning and Computing, Vol. 4, No. 2, pp. 194-197, Apr. 2014, https://doi.org/10.7763/IJMLC.2014.V4.411

[7] S. Rungta, S. Srivastava, U. S. Yadav, and R. Rastogi, "A Methodology to Find the Cycle in a Directed Graph Using Linked List", International Journal of Information Technology, Vol. 6, No. 2, pp. 743-749, Dec. 2014.

[8] G. Pandy and S. Chawla, "A Simple and Efficient Algorithm for Hypercycle Detection", School of Information Technologies, University of Sydney, Technical Report No. 538, pp. 1-9, Aug. 2003.

[9] L. Wu, K. Su, S. Cai, X. Zhang, C. Zhang, and S. Wang, "An I/O Efficient Approach for Detecting All Accepting Cycles", IEEE Transactions on Software Engineering, Vol. 41, No. 8, pp. 730-744, Aug. 2015, https://doi. org/10.1109/TSE.2015.2411284

[10] R. C. Rocha and B. D. Thatte, "Distributed Cycle Detection in Large-Scale Sparse Graphs", Conference on XLVII SBPO - Simpósio Brasileiro de Pesquisa Operacional, At Pernambuco, Brazil, pp. 1-11, Nov. 2015.

[11] Codingfreak, "Datastructures: Detecting a Loop in Single Linked List-Tortoise & Hare", http://codingfreak.blogspot.com/p/data-structures.html, Sep. 2012.

[12] Wikipedia, "Cycle Detection," https://en.wikipedia.org/wiki/Cycle_detection, Wikimedia Inc., Mar. 2016.

[13] B. Haeupler, T. Kavitha, R. Mathew, S. Sen, and R. E. Tarjan, "Incremental Cycle Detection, Topological Ordering, and Strong Component Maintenance", ACM Transactions on Algorithms, Vol. 8, No. 1, pp. 3:1-3:33, Jan. 2012, https://doi. org/10.1145/2071379.2071382

[14] M. Safar, F. Mahdi, and K. Mahdi, "An Algorithm for Detecting Cycles in Undirected Graphs using CUDA Technology", International Journal of New Computer Architectures and Their Applications, Vol. 2, No. 1, pp. 193-212, Jan. 2012.

[15] A. Boukerche and C. Tropper, "A Distributed Graph Algorithm for the Detection of Local Cycles an Knots", IEEE Transactions on Parallel and Distributed Systems, Vol. 9, No. 8, pp. 748-757, Aug. 1998, https://doi.org/10.1109/71. 706047

[16] D. Panigrahi, "COMPSCI 330: Design and Analysis of Algorithms," pp. 1-5, Duke University Department of Computer Science, pp. 1-5, Sep. 2014.

저 자 소 개

이 상 운(정회원)

- 1987년 : 한국항공대학교 항공전자공학과 (학사)
- 1997년 : 경상대학교 컴퓨터과학과 (석사)
- 2001년 : 경상대학교 컴퓨터과학과 (박사)
- 2003년 : 강원도립대학 컴퓨터응용과 전임강사
- 2004년 ~ 2007.2 : 국립 원주대학 여성교양과 조교수
- 2007.3 ~ 2015.3 : 강릉원주대학교 멀티미디어공학과 부교수
- 2015.4 ~ 현재 : 강릉원주대학교 멀티미디어공학과 정교수
- 관심분야 :소프트웨어 프로젝트 관리, 개발 방법론, 분석과 설계 방법론, 시험 및 품질보증, 소프트웨어 신뢰성, 인공지능과 빅데이터분석, 최적화 알고리즘
- e-mail : sulee@gwnu.ac.kr