퍼지 및 지능시스템학회 논문지 2005, Vol. 15, No. 4, pp. 425-430

분산된 대사경로네트워크에 대한 경로검색을 위한 분산알고리즘

A Distributed Path-Finding Algorithm for Distributed Metabolic Pathways

이선아[1], 이건명[1*], 이승주[2]

Sun A Lee, Keon Myung Lee, Seung Joo Lee

[1]충북대학교 전기전자컴퓨터공학부

[2]청주대학교 생명유전통계학과 통계학전공

[1]School of Electrical and Computer Engineering, Chungbuk National University

[2]Department of Statistics, Cheongju University

요 약

많은 문제가 그래프로 모델링될 수 있고, 그래프 이론에 기반한 방법에 의해서 해결될 수 있다. 이 논문에서는 분산되고 중첩된 대사경로 네트워크들에 대해서 경로를 찾는 방법에 대해서 다룬다. 제안한 방법은 분산된 그래프를 통합하지 않은 채, 다중 에이전트의 협동작업을 통해서 경로를 찾는 방법이다. 각 그래프에는 해당 그래프를 책임지고 있는 에이전트가 하나씩 있어서, 해당 그래프에서 시작되는 경로검색을 주도하고, 다른 에이전트로부터 경로에 대한 정보 요청에 응답하도록 한다. 제안한 방법에서는 우선 전체 분산된 그래프에 대해서 뷰그래프라고 하는 추상화된 그래프를 형성하고, 이를 이용하여 경로를 찾기 위해 에이전트간에 어떤 방법으로 협력을 할지 알 수 있게 한다. 각 에이전트는 해당 그래프에 대한 최단경로 정보를 관리하고 있다. 어떤 에이전트가 해당 그래프의 어떤 노드에서 시작하는 경로를 찾으라는 요구를 받게 되면, 다른 에이전트로부터 정보를 받아서 목적지까지는 가는 경로를 찾게 된다.

Abstract

Many problems can be formulated in terms of graphs and thus solved by graph-theoretic algorithms. This paper is concerned with finding paths between nodes over the distributed and overlapped graphs. The proposed method allows multiple agents to cooperate to find paths without merging the distributed graphs. For each graph there is a designated agent which is charged of providing path-finding service for her graph and initiating the path-finding tasks of which path starts from the graph. The proposed method earlier on constructs an abstract graph so-called viewgraph for the distributed overlapped graphs and thus enables to extract the information about how to guide the path finding over the graphs. The viewgraph is shared by all agents which determine how to coordinate other agents for the purpose of finding paths. Each agent maintains the shortest path information among the nodes which are placed in different overlapped subgraphs of her graph. Once an agent is asked to get a path from a node on her graph to another node on another's graph, she directs other agents to provide the necessary information for finding paths.

Key words : metabolic pathway, distributed algorithm, multiagent, path finding

1. 서 론

컴퓨터의 성능 향상, 고성능 분석도구 및 분석기법의 개발에 따라 생물학 데이터가 급속히 증가하고 있다. 핵산서열, 단백질서열, 대사경로, 화합물 정보, 단백질 서열 데이터 등 다양한 생물학 데이터가 데이터베이스화되어 제공되고 있다. 핵산 서열 데이터베이스는 GenBank, EMBL, DDBJ 등이 대표적이고, 아미노산 서열 데이터베이스로는 PIR, PRF, SWISS-PROT 등이 있고, 3차원 구조 데이터베이스에는 PDB, CSD 등이 있고, 단백질 족 및 서열 모티프(motif) 데이터베이스로는 PROSITE, Blocks, PRINTS, Pfam, ProDom 등이고, 대사 경로 데이터베이스로는 KEGG, WIT, EcoCyc, UM-BBD, MetaCyc, aMaze, CSNDB, PathDB, SHARKdb 등이 있고, 게놈의 다양성에 대한 데이터베이스로는 NCBI Taxonomy, OMIM 등이 있다. 이들 다양한 데이터베이스는 서로 중복되는 것도 있고, 특정 데이터베이스에만 존재하는 데이터들도 있다. 경우에 따라서는 서로 다른 형태로 데이터를 저장하고, 제공한다. 이들 데이터베이스에 대해서 효과적으로 원하는 정보를 추출하고, 이에 포함된 데이터에 대해서 효과적으로 분석하기 위한 연구가 바이오인포

* 교신저자

접수일자 : 2005년 6월 10일

완료일자 : 2005년 7월 15일

감사의 글 : 이 논문은 2005학년도 충북대학교 학술연구지원사업의 연구비지원에 의하여 연구되었음.

매틱스 분야에서 많이 활발히 연구되고 있다. 생물학자들은 관심대상에 대한 정보를 찾기 위해서는 해당 정보를 포함하고 있음직한 모든 데이터베이스를 직접 검색해서 원하는 정보를 검색하는 것이 대부분이다. 이러한 번거로운 해결하는 단순한 접근방법은 관련 데이터베이스를 통합하여, 하나의 데이터베이스로 만드는 것이지만, 이러한 일은 데이터의 형식의 불일치 등으로 인해서 쉬운 작업이 아니다. 이 논문에서는 대사경로 데이터베이스을 대상으로 한다. 대사경로 데이터베이스에 대한 주요 연산으로 노드(단백질 또는 대사물) 간의 경로를 검색하는 것이 있다. 대사경로 데이터는 화합물 그래프, 반응 그래프, bipartite 그래프, 하이퍼그래프 등 여러 가지로 모델링 되어 표현되기 때문에, 이질적인 데이터베이스들을 하나로 통합하는 것이 쉽지 않다. 따라서 대사경로 데이터베이스들을 통합하지 않고, 경로를 검색할 수 있도록 하는 것이 생물학자 입장에서는 매우 편리하다. 이를 이 논문에서는 다중에이전트 구조를 이용하여, 대사경로 데이터베이스를 통합하지 않은채, 경로검색을 할 수 있는 다중 에이전트 기반 분산경로검색 알고리즘을 제안한다.

이 논문은 다음과 같이 구성된다. 2절에서는 기존의 대사경로 검색에 대한 관련 연구에 대해서 살펴본다. 3절에서는 제안한 대중 에이전트 기반의 분산경로검색 알고리즘에 대해서 소개하고, 4절에서는 적용가능한 예로서 서로 다른 종간 비교를 위해 경로탐색 알고리즘을 적용하는 것을 보인다. 끝으로 5절에서 결론을 맺는다.

2. 관련 연구

대사경로 데이터베이스에서 경로검색은 주요한 연산으로서, 대부분의 대사경로 데이터베이스 시스템은 경로검색 서비스를 제공하고 있다. 경로검색에도 여러 가지가 있는데, 특정 대사물간의 최단경로를 찾는 것, 특정 대사물로부터 일정 거리 이내에 존재하는 대사물 및 경로를 찾는 것, 대사물간의 모든 경로를 찾는 것, 특정 대사물을 경유하는 대사물간의 경로를 찾는 것 등이 있다. 이러한 경로검색을 위해서 여러 가지 방법이 시도되어 왔다. BioMiner[6]의 PathFinder는 대사경로를 그래프로 표현하고 검색 범위를 줄이는 기법을 도입하여 효율적으로 검색하는 방법을 사용한다. 대표적인 대사경로 데이터베이스인 KEGG 등은 대부분 탐색경로를 최대 k로 제한하지만, PathFinder는 $k/2$로 줄이며 중복된 반응에 대해 집합화하고 검색할 때에 중복된 반응을 검색에 이용하고, 각 path에 가중치를 적용하여 검색범위를 줄인다. Küffner et al.[7]이 제안한 방법은 대사경로를 Petri net으로 표현하는데, Petri net을 이용하면 효소를 노드로 나타내어 보다 많은 정보를 표현할 수 있게된다. 이 방법에서는 발현 데이터를 분석하여 증명된 것에 가중치를 부여한다. 이들 방법은 대사경로 네트워크에 대해 그래프로 표현하고 검색범위를 한정하는 공통점을 가진다. 또한 중복되는 부분을 집합화하거나 네트워크 구조를 다르게 표현하는 방법을 사용함으로써 보다 좋은 검색 결과를 얻을 수 있고 있다.

경로검색에 대한 접근방법으로 그래프 마이닝 기법을 이용하여, 이 정보를 경로검색에 활용하는 기법들도 연구되고 있다[3,5]. 그래프 마이닝이란 검색 대상이 일반 데이터가 아닌 그래프이며 해당 그래프와 인접한 데이터들의 특징을 분석하여 빈번하게 사용되는 데이터를 찾는 방법이다. 그래프에서 자주 사용하는 부분을 찾아 이 부분을 처리하도록 함으로써 검색에 도움을 줄 수 있다. 원래의 그래프 마이닝은 그래프의 동형(isomorphic) 부분을 찾는데 사용되던 방법이지만, 빈번하게 사용되는 부분을 찾는 데에는 유용하다. 기존의 데이터 마이닝 방법을 응용한 방법으로 대사물을 노드로 표현하던 예전과 달리 효소를 노드로 표현하고, 다음 노드 탐색을 위해 역추적(backtracking) 방법을 기반으로 한 깊이 우선 방법과 근접한 노드 정보를 이용하는 방법을 사용한다[4]. 이 방법의 경우 데이터 마이닝과 개념상 유사하며 임계값을 달리하여 여러 단계로 자주 사용되는 그래프를 찾을 수 있다는 장점이 있다. 이러한 접근방법으로는 Inokuchi et al.[5]이 제안한 a priori 기반의 알고리즘(AGM)방법과 gSpan[3]이 있다.

분산된 네트워크에 대한 경로검색 연구는 GPS 기반의 네비게이션 시스템 등에서 최단경로 검색 기법 들이 개발되어 사용되고 있으나, 사용되는 기법들은 유클리디안 거리의 성질을 만족하는 그래프에 대해서만 적용될 수 있는 것으로 휴릭스틱한 방법들이다. 이 논문에서는 네트워크가 일부 중첩되면서 분산되어 있는 네트워크들에 대해서 경로를 찾는데, 다중 에이전트를 통해서 경로를 찾는 방법을 제안한다. 에이전트는 위임된 역할을 자율적으로 수행하는 컴퓨팅 요소로서, 이러한 에이전트를 여러개 사용하여 문제를 해결하는 기법을 다중 에이전트 기반의 접근방법이라 한다.

3. 제안한 분산 알고리즘

이 절에서는 제안한 다중에이전트 기반의 중첩된 분산 네트워크에 대해서 경로를 검색하는 방법을 소개한다. 제안한 방법에 대한 기술을 효과적으로 하기 위해서 다음과 같은 표기법을 사용한다.

$G = \{ G_1, G_2, \cdots, G_n \}$: 그래프 G_i의 집합

$G_i = (N_i, E_i)$: 노드집합 N_i와 에지집합 E_i를 갖는 그래프 G_i

$N_i = \{n_{i1}, n_{i2}, \cdots, n_{in_i}\}$: n_i개의 노드를 갖는 노드집합 N_i

$E_i = \{ e_{ij} | e_{ij} = (n_{ik}, n_{il}), n_{ik}, n_{ik} \in N_i \}$

$GO_{ij} = (NO_{ij}, EO_{ij})$: 그래프 G_i와 G_j 간의 중첩된 서브그래프, $NO_{ij} = N_i \cap N_j$, $EO_{ij} = E_i \cap E_j$

A_i: G_i를 담당하는 에이전트

제안한 방법에서는 분산된 네트워크에 대한 전체적인 형태를 추상화하기 위해서는 다음과 같이 뷰그래프(viewgraph)라고 하는 그래프를 정의하여 사용한다.

정의. 그래프 집합 G에 대한 뷰그래프 $VG(G)$는 노드집합 $VG(G)$와 에지집합 $VE(G)$가 G로부터 다음과 같이 정의되는 것이다.

$VN(G) = \{ Vn_1, Vn_2, \cdots, Vn_n \}$, Vn_i는 G_i에 대응

$VE(G) = \{ Ve_{ij} | Ve_{ij} = (Vn_i, Vn_j), \text{and} NO_{ij} \neq \phi \}$

n_s : 경로 탐색이 시작되는 시작 노드

n_d : 경로 탐색에서 목적지 노드

$G_{(n_i)}$: 노드 n_i를 포함하는 그래프

$G_{(n_i)} = \text{argmin}_k \{ G_k | n_i \in N_i \}$

$A_{(n_i)}$: 그래프 $G_{(n_i)}$에 대응하는 뷰그래프 노드
$GO_{(n_i)j}$: $G_{(n_i)}$와 G_j간의 중첩된 서브그래프
$Adj(Vn_i)$: Vn_i에 대한 인접한 뷰그래프 노드

3.1 뷰그래프를 이용한 분산네트워크 추상화

제안한 방법에서는 다중에이전트 기반으로 에이전트간의 협동작업에 의해서 분산된 네트워크(그래프)들에 대해서, 네트워크를 통합하지 않고 경로를 검색하게 한다. 이를 위해서 전체 네트워크에 대한 전체 구조를 추상화하여, 이로부터 에이전트간 협동을 위한 정보교환 절차를 결정하기 위해 사용할 정보를 효과적으로 나타내는 뷰그래프를 생성하여 에이전트들이 공유하게 된다. 다음 프로시져 Construct_ViewGraph는 그래프 집합 G를 입력으로 받아서, 출력으로 뷰그래프 VG를 생성하는 역할을 한다.

procedure *Contruct_ViewGraph*(G, VG)
input : 그래프 집합 G
output : G에 대응하는 뷰그래프 VG
begin
For each graph $G_i \in G$, create a node Vn_i.
For i = 1 to n
For $j = i$ to n
If $G_i \cap G_j \neq \phi$, then
Create an edge Ve_{ij} between Vn_i and Vn_j
Build $VG(V) = (VN, VG)$ where $VN = \{Vn_i\}$ and $VE = \{Ve_{ij}\}$.
end.

3.2 제안한 분산 경로검색 알고리즘

경로를 찾기 위해 상호협동작업을 시작할 때, 지정된 하나의 에이전트가 다른 에이전트들로부터 해당 에이전트가 담당하는 네트워크의 노드 집합을 수집한 다음, 프로시져 Construct_ViewGraph를 이용하여 뷰그래프를 작성한다. 작성된 뷰그래프는 모든 다른 에이전트에 전달하여, 모든 에이전트가 뷰그래프를 공유하게 한다.

3.2.1 에이전트의 전처리

효과적으로 경로를 찾기 위해, 각 에이전트는 우선 자신이 담당하는 그래프의 노드 간에 최단 경로를 미리 찾아둔다. 각 그래프 G_i에 대해, 에이전트 A_i는 노드 집합 N_i를 다른 그래프에도 노드가 포함되는 중첩된 서브그래프 G_{ij}와, 단지 G_i에만 노드가 포함되는 N_{io}로 분할한다. 다음 프로시져 Find_Shortest_Paths_in_a_Graph는 각 에이전트 A_i가 노드를 찾을 때 사용하는 것이다.

procedure Find_Shortest_Paths_in_a_Graph(G_i)
input : G_i 에이전트 A_i가 담당하는 그래프
begin
For k = 1 to the number ns_i of overlapped subgraph
For $l = k$ to ns_i
Find the shortest paths for each pair of nodes of which one node belongs to G_{ik} and of which the other node belongs to G_{il}.
Find the shortest paths for each pair of nodes of which one node belongs to G_{i0} and of which the other node belongs to $N_i - G_{i0}$.
end.

3.2.2 뷰그래프에서 경로 찾기

에이전트가 자신의 그래프의 노드에서 시작하는 경로를 찾으라는 요구를 받으면, 먼저 경로를 찾기 위해서 다른 에이전트들과 어떻게 협조를 해야 할지 계획을 세워야 한다. 다음 프로시져 Find_ViewPath는 에이전트 A_i가 에이전트 조정을 위한 계획을 수립하도록 한다. 에이전트 A_i가 경로탐색 요구를 받으면, 경로의 시작노드는 A_i의 그래프 G_i에 있다고 전제한다.

procedure Find_ViewPath(n_s, n_d, $VP(n_s, n_d)$)
input: start node n_s, destination node n_d
output : a set of viewpaths $VP(n_s, n_d)$
begin
Search the graph $G_{(n_s)}$ which contains n_d.
Find all paths $VP(n_s, n_d)$ from Vn_i to $Vn_{(n_d)}$ in the viewgraph $VG(G)$ using the procedure Find_All_Paths_between_Nodes($Vn_i, Vn_{(n_d)}$).
end.

뷰그래프에서 시작 노드 $Vn_{(n_s)}$에서 목적지 노드 $Vn_{(n_d)}$까지의 경로를 나타내는 경로 트리를 $PT(Vn_{(n_s)}, Vn_{(n_d)})$라고 하자. 경로 트리에서 각 노드는 뷰그래프의 노드이름을 레이블로 갖고, 루트 노드에서 단말 노드까지의 경로상에서 레이블의 서열은 뷰그래프 상에서 경로를 나타낸다. 뷰그래프에서, $Vn_{(n_s)}$에서 $Vn_{(n_d)}$로의 경로는 사이클을 포함할 수 있지만, 한 경로상에서 동일한 사이클은 한번만 허용한다. 다음 프로시져 Find_All_Paths_between_ViewNodes는 $Vn_{(n_s)}$에서 $Vn_{(n_d)}$까지의 모든 경로를 찾아서, 이를 경로 트리 형태로 반환하는 역할을 한다.

procedure Find_All_Paths_between_ViewNodes ($Vn_{(n_s)}, Vn_{(n_d)}$, $PT(Vn_{(n_s)}, Vn_{(n_d)})$)
input : 시작 노드 $Vn_{(n_s)}$, 목적지 노드 $Vn_{(n_d)}$
output : 뷰그래프 $VG(G)$에서 $Vn_{(n_s)}$에서 $Vn_{(n_d)}$로의 모든 경로를 표현하는 경로 트리 $PT(Vn_{(n_s)}, Vn_{(n_d)})$
begin
Create an empty queue Q.
Create the root node pn_0 of the path tree

corresponding to $Vn_{(n_s)}$.

Enqueue the pointer p_0 to pn_0 to Q.

While Q is not empty

Dequeue a pointer p_i from Q and let Vn_j be the node label corresponding to p_i.

For each $Vn_k \in Adj(Vn_j)$

Find all the ancestor nodes $AN(Vn_k)$ with the node label Vn_j.

If there are no children for all $p_l \in AN(Vn_k)$ whose node label is Vn_kon the path from Vn_j to the root

Create a new node p_n with the label Vn_k.

Append p_n as a child to the node corresponding to the pointerp_i.

Add the pointer for p_n to Q.

End If

End For

End While

For each leaf node

If the leaf node is not labeled with $Vn_{(n_d)}$

Remove the subtree of which root node is the closest node from the leaf node and has sibling node(s).

If there are no leaf nodes of $Vn_{(n_d)}$

Return the message 'No Path between n_s and n_d'.

Return the constructed tree rooted at p_0 as the path tree $PT(Vn_{(n_s)}, Vn_{(n_d)})$.

end.

3.2.3 중첩된 분산 그래프에서 경로 찾기

뷰그래프 상에서 $Vn_{(n_s)}$에서 $Vn_{(n_d)}$까지 모든 경로를 찾은 다음, 그래프 $G_{(n_s)}$에 대한 에이전트 $A_{(n_s)}$는 다른 그래프와 중첩되는 $G_{(n_s)}$의 다른 노드로의 경로를 찾는다. 한편, $A_{(n_s)}$는 에이전트 $A_{(n_d)}$에게 $A_{(n_s)}$의 노드 n_s로부터 $G_{(n_d)}$에 속하는 노드 n_d로의 경로탐색에 대한 요청이 있다는 것을 알려준다. 에이전트 $A_{(n_d)}$는 목적지 노드 n_d로부터 다른 그래프와 중첩되는 부분에 위치하는 노드들로의 경로를 계산한다. 각 에이전트는 자신이 담당하는 그래프에 대해서 서로 다른 중첩된 서브그래프에 포함되는 노드간의 경로를 미리 계산하여 경로정보를 유지관리한다. 프로시저 Find_All_Paths_between_ViewNodes를 이용하여 구성한 경로 트리 $PT(Vn_{(n_s)}, Vn_{(n_d)})$는 원래 중첩된 분산 그래프 집합에서 최단 경로를 포함하는 후보 뷰경로를 찾는데, 워크플로우 역할을 하게 된다. 다음 프로시저 Find_Candidate_Shortest_ViewPath는 그래프 집합 G에서 n_s에서 n_d로 가는 최단 경로를 포함하는 뷰그래프 $VG(G)$ 상의 경로를 찾는다.

procedure Find_Candidate_Shortest_ViewPath

input : the path tree $PT(Vn_{(n_s)}, VN_{(n_d)})$

output : a viewpath from $Vn_{(n_s)}$ to $Vn_{(n_d)}$

global variable :

minPL, 지금까지 찾은 최단경로 길이

bestPH, 지금까지 찾은 최적 뷰경로의 단말 뷰노드

begin

Compute the shortest path length from n_s to all other nodes on the graph with n_s.

Make a list $DL = \{(n_a, d_{sa})\}$ of pairs of node name $n_a \in N_{(n_s)}$ and the distance d_{sa} from n_s to n_a.

Set $minPL = \infty$.

For each child node C_i of the root node C_0 on the path tree PT.

Let A_k be the agent which takes care of the graph G_k corresponding to C_i.

Send the agent A_k the message *Compute_ViewPath_Length*(DL, C_0).

While there is response from other agents about *minPL* and *minPH*.

Update the current *minPL* and *minPH* with the received one.

Construct the viewpath from $Vn_{(n_s)}$ to *minPH* by traversing back from *minPH* to the root and Return it.

end.

메시지 *Compute_ViewPath_Length*(DL, C_0)를 받으면, A_k는 프로시저 Compute_ViewPath_Length(DL, C_0)를 이용하여 다음 작업을 수행한다. 먼저 n_s에서부터 자신이 담당하는 그래프에 있는 각 노드 $n_b \in G_k$로 가는 경로길이를 계산한다. DL은 n_s로부터 C_i에서 도달할 수 있는 각 노드 n_a로의 거리 d_{sa}에 대한 정보를 저장한 리스트이다. 각 에이전트 A_k는 자신에 속하는 노드 n_a와 n_b 간의 거리정보 d'_{ab}를 가지고 있다.

procedure Compute_ViewPath_Length(DL, C_i)

begin

Let $GO_{(i)k} = (NO_{(i)k}, EO_{(i)k})$ be the overlapped subgraph between the graph G_k and the graph corresponding to C_i.

Create a new list DL'.

For each $n_b \in G_k$

Compute $d'_{sb} = min_{a_n \in NO_{(i)k}}\{d_{sa} + d''_{ab}\}$

If $d'_{sb} < minPL$

$DL' \leftarrow DL' \cup \{(n_b, d''_{sb})\}$.

If $n_b = n_d$

$minPL = d''_{sb}$

Broadcast all agents the value $minPL$.

If DL' is not empty

For each child node C_j of C_i on the path tree PT

Let A_l be the agent which takes care of graph G_l corresponding to C_j.

Send A_l the message Compute_ViewPath_Length(DL, C_j).

end.

4. 종별 대사경로 비교를 위한 대사경로 탐색 적용

제안한 방법은 KEGG[1,2]의 대사경로 데이터베이스를 대상으로 종별 대사경로를 탐색하고 종별 대사경로를 서로 비교하는데 이용할 수 있다. KEGG에서 제공하는 대사경로 데이터베이스는 종에 따라 서로 다른 데이터를 제공한다. 뿐만 아니라 저장된 모든 종의 대사경로 데이터를 적절히 조합하여 하나의 커다란 대사경로 네트워크를 그래프로 제공하여 한눈에 전체를 볼 수 있도록 한다. 종마다 가지고 있는 대사경로는 서로 다르다. 서로 다른 그래프를 하나로 표현하기 위해서 KEGG에서는 제공하는 그림에는 표현되지 않았지만 실제 종마다 다르게 나타나는 반응들이 있다. 이 적용 예에서는 이렇게 가려진 정보들을 보완하고자 EcoCyc과 같이 특정 종에 대해 보다 자세한 대사경로 데이터를 제공하는 다른 데이터베이스들을 통합한 것을 검색 대상으로 한다. KEGG는 같은 이름의 대사경로에 대해 종마다 경로가 다르며 큰 그래프 상에 나타낼 때에는 특정 종의 대사경로의 노드를 흰색과 다른 색으로 나타낸다. 그림을 통해 한 눈에 종별로 어떤 과정을 거치는지를 알 수 있다. 하지만 KEGG에 대해서만 알 수 있다는 단점과 알고자 하는 경로에 대해 대사경로별, 혹은 종에 한정하여 보여주기 때문에 제약이 있다.

이 적용 예는 검색 대상을 종이 가지는 모든 대사경로에 대해 경로를 탐색하고 종별 경로를 제시함으로써 나중에 종에 따른 차이점이나 특이성을 보다 효율적으로 비교할 수 있도록 도와준다. 또한 종별로 중복되는 부분을 하나의 집합화하여 경로 탐색이나 서열을 비교할 때에 더 잘 할 수 있도록 한다.

KEGG에서 제공하는 대사경로 데이터는 종에 따라 분류되어 있다. KEGG 이외의 다른 데이터베이스 또한 일치하는 종에 저장한다. 시작 노드 n_s와 목적지 노드 n_d에 대한 경로를 찾아야 한다. 먼저 종에 해당하는 대사경로들에 대해 중복되는 부분을 이용해 대사경로 네트워크를 그래프로 표현한다. 이들을 이용해 각각의 대사경로는 뷰노드로, 서로 중복되는 부분이 있다면 이들을 연결하는뷰에지로 하는 뷰그래프를 만든다. 예를 들어 그림 1과 같이 나타낼 수 있다. 그림 1에서 각각은 대사경로를 나타낸다. a는 Glycolysis metabolism, b는 Pyruvate metabolism, c는 Citrate metabolism, d는 Alanine and asparate metabolism을 나타낸다. 각 노드를 연결하고 있는 에지는 두 대사경로가 중복되는 정보를 가지고 있음을 나타내며 연결이 없는 것은 직접적인 영향이 없음을 나타낸다. 이 정보는 KEGG에서 제공하고 있는 대사경로 정보를 이용하여 Glycolysis metabolism과 관련된 대사경로 중 임의의 4가지 대사경로를 선택하여 도식화한 것이다.

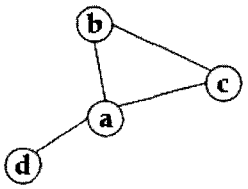

그림 1. 뷰그래프

시작노드 n_s가 어느 대사경로에 존재하는지 알아야 한다. 시작 노드를 기준으로 해당 대사경로에 존재하는 노드들에 대해 깊이 우선 탐색방법을 적용한다. 해당 뷰노드 즉, 대사경로에 대해 검색을 한 결과 목적지 노드n_d 를 찾지 못하면 해당 뷰노드와 연결된 다른 뷰노드에 대해 해당 에이전트에 요구를 하여 검색한다. 연결되지 않은 뷰노드는 검색에서 제외한다. 이때 무한정 경로 탐색이 아니라 임의의 최대 k까지 탐색이 가능하도록 하는 것이 반응시간 측면에서 바람직하다. 만일 해당 뷰노드 안에 n_d 노드가 존재하더라도 경로 탐색은 더 해야 한다. 다른 그래프에 있는 노드를 이용하였을 때에 다른 경로가 발생할 수 있기 때문이다.

시작 노드 n_s와 목적지 노드 n_d가 어느 대사경로에 존재하는지 알 경우에 문제는 더욱 간단하다. n_s를 포함한 대사경로 M1과 n_d를 포함한 대사경로 M2가 있다는 것이 알려져 있다면, 뷰그래프 상에서 M1부터 M2까지의 경로를 제안한 다중 에이전트 기반 알고리즘을 이용하여 찾는다. 각 대사경로들 사이에 중복되는 부분에 대한 정보를 가지고 있기 때문에 이들을 기점으로 M1에서부터 중복되는 부분까지 그 부분부터 다음 뷰노드와 중복된 부분까지의 경로를 찾는 방식으로 검색을 해 나간다. 만일 그림 1에서 하이퍼 노드 c에 있는 a'에서 d에 있는 a''까지 가능한 모든 경로를 찾으려면 먼저 하이퍼 그래프 상에서 c에서 d까지의 경로를 구한다. 경로 트리에서 볼때, c-b-a-d, c-a-d로 a-d는 중복된 경로이으로 한번만 탐색하도록 다중 에이전트를 조정한다. 뷰그래프에서 가능한 모든 경로를 찾은 다음에는 찾아진 뷰경로에 존재하는 노드에 존재하는 하위 요소들에 대해 경로탐색을 한다. a'에서 b와 a가 c와 중복되는 노드들까지의 경로를 찾는다. 또한 n_d가 존재하는 d와 a의 중복지점들까지의 경로를 구한다. b-a까지의 경로를 더 구하여 부분 경로가 모두 구해지면 이들을 합하여 전체 경로를 구한다. 뷰그래프를 통해 시작 노드가 존재하는 하이퍼 노드와 목적지 노드가 존재하는 하이퍼 노드 사이의 모든 경로를 구한다. 이 경로를 기반으로 연결지점의 하위 노드들을 찾고 이것을 기준으로 부분적으로 계산할 수 있다. 종별 대사경로를 탐색하여 종에 따라 중복되는 경로와 서로 다른 경로를 파악할 수 있다. 종마다 중복되는 경로를 하나로 나타냄으로써 비교 범위를 줄일 수 있다. 전체 그래프를 비교하지 않고 중복되지 않은 부분을 통해 차이점을 알 수 있다. 식물, 사람, 효모 등은 서로 공통된 부분을 가지고 있지만 차이나는 부분이 존재하여 이 부분이 해당 종의 특징을 나타낸다고 할 수 있다.

5. 결 론

다양한 생물정보학 데이터베이스 구축에 따라 이를 효과적으로 활용할 수 있도록 하기 위한 도구 개발이 활발히 연구되고 있다. 이 연구에서는 분산된 대사경로 데이터베이스

들에 대해서 이들을 통합하지 않은 상태로 다중 에이전트를 이용하여, 대사물간의 경로를 검색하는 분산 알고리즘을 제안하였다. 제안한 방법은 각 대사경로 네트워크에 대응하는 전담 에이전트를 두고, 이들 에이전트들이 서로 정보교환하는 방법으로 경로를 찾도록 한 것이다. 이를 위해서는 전체 네트워크에 대해서 추상화된 뷰그래프를 형성하고 이를 이용하여 에이전트간 정보교환 순서를 결정할 수 있도록 하고 있다. 제안한 다중에이전트 기반 알고리즘은, 다수의 대규모 데이터베이스를 통합하지 않고, 이질적으로 표현된 경로 데이터베이스에 대한 경로검색을 할 수 있게 하기 때문에 매우 유용하다. 또한 적용가능한 예로, 다른 종의 대사경로를 비교하는데 제안한 방법을 적용하는 것을 소개하였다.

참 고 문 헌

[1] O. Hiroyuki Ogata, G. Susumu, S. Kazushige, F. Wataru, B. Hidemasa, "KEGG: Kyoto Encyclopedia of Genes and Genomes", Nucleic Acids Research, Vol.27, No.1, 1999.

[2] S. Goto, H. Bono, H. Ogata, W. Fujibuchi, K. Sato, M. Kanehisa, "Organizing and computing metabolic pathway data in terms of binary relations", Pacific Symp. Biocomput., 175-186, 1997.

[3] X. Yan, J. Han, "gSpan: Graph-based substructure pattern mining", IEEE International Conference on Data Mining(ICDM'02), Maebashi City, Japan, December, pp.721-724, 2002.

[4] M. Koyutürk, A. Grama, W. Szpankowski, "An efficient algorithm for detecting frequent subgraphs in biological networks", Bioinformatics, Vol.20, Suppl. 1, pp.i200-i207, 2004.

[5] A.Inokuchi, T.Washio, H.Motoda, "An apriori-based algorithm for mining frequent substructures from graph data", PKDD'00, pp.13-23, 2000.

[6] M.Širava, T.Schäfer, M.Eiglsperger, M.Kaufmann, O.Kohlbacher, E.Bornberg-Bauer and H.P.Lenhof, "BioMiner-modeling, analyzing, and visualizing biochemical pathways and networks", Bioinformatics, Vol.18, Suppl.2, pp.S219-S230, 2002.

[7] Küffner, R. et al., "Pathway analysis in metabolic databases via differential metabolic display(DMD)", Bioinformatics, Vol.16, pp.825-836, 2000.

[8] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, Introduction to Algorithms, McGraw-Hill, 2001.

[9] J. A. Azevedo, M.E.O. Santos Cost, J. J. E. R. Silvestre Madeira, E. Q. V. Martins. "An Algorithm for the ranking of shortest paths", European J. Operational Research, Vol.69, pp.97-106, 1993.

[13] D. Eppstein. Finding the k shortest paths.SIAM J. Computing, Vol.28, No.2, pp.652-673, 1988.

[14] V. Jimenez, "A. Marzal. Computing the k Shortest Paths: a New Algorithm and an Experimental Comparison", Lecture Notes in Computer Science series. Springer-Verlag, Vol.1668. pp.15-29, 1999.

[15] S. Kanchi, D. Vineyard. "An optimal distributed algorithm for all-pairs shortest-paths", Proc. ICT&P 2004(Bulgaria), 2004.

저 자 소 개

이선아(Sun A Lee)
충북대 컴퓨터과학과 졸업
충북대 전자계산학과 석사
충북대 전자계산학과 박사과정
관심분야: 인공지능, 바이오인포매틱스

이건명(Keon Myung Lee)
제 14권 6호 참조

이승주(Seung Joo Lee)
청주대 응용통계학과 졸업
동국대 통계학과 이학석사
동국대 통계학과 이학박사
현 청주대학교 응용통계학과 부교수
관심분야 : 베이지안 접근론, 바이오인포매틱스