공학석사 학위논문

분자 컴퓨팅을 이용한 원숭이와 바나나 문제 해결

Solving the Monkey and Banana Problem
Using Molecular Computing

2003년 8월

서울대학교 대학원
협동과정 인지과학전공

박 의 준

분자 컴퓨팅을 이용한 원숭이와 바나나 문제 해결

Solving the Monkey and Banana Problem
Using Molecular Computing

지도교수 장 병 탁

이 논문을 공학석사 학위논문으로 제출함

2003년 4월

서울대학교 대학원
협동과정 인지과학전공
박 의 준

박의준의 공학석사 학위논문을 인준함
2003년 6월

위 원 장	김영정 인
부위원장	장병탁 인
위 원	신효필 인

국 문 초 록

원숭이와 바나나 문제는 인공지능과 관련된 여러 문헌에서 문제 해결(problem solving) 과정을 설명하는 예제로 자주 등장한다. 이 문제에 대한 전통적인 접근 방식은 추론을 수행함에 있어 절차적(procedural) 관점의 도입을 필요로 하며, 이는 복잡한 문제 해결에 제약 조건으로 작용한다. 그러나 대규모 병렬 처리가 가능한 DNA 컴퓨팅 기법을 이용하면, 넓이 우선 탐색(breadth-first search, BFS)에 근거하여 이 문제를 효과적으로 해결할 수 있다. 본 논문에서는 DNA 분자들을 사용하여 원숭이와 바나나 문제를 표현하는 방법을 제시한 후, 컴퓨터 시뮬레이션 및 실험을 통해 최소 4 종류 이상의 다양한 해들(solutions)이 생성됨을 실제로 확인한다. 전통적인 방식이 단 하나의 최적해밖에 제공해 주지 못한다는 사실과 비교해 볼 때, 이것은 문제 풀이 과정에서 나타나는 어떤 인지적인 특성을 포착하고자 한다는 측면에서 의미 있는 결과이다.

주제어 : 원숭이와 바나나 문제, 논리적 추론, DNA 컴퓨팅

학 번 : 2001-23113

목 차

1. 서론

1.1. 연구 배경

원숭이와 바나나 문제(Monkey and Banana Problem)는 상식적인 추리를 수행하는 자동 문제 풀이기(automatic problem solver)의 동작을 설명하기 위해 인공지능(AI)과 관련된 여러 문헌에서 자주 언급되는 문제이다. 그런데 고전적 인공지능의 입장에서는 근본적으로 모든 문제에 대해 순차적(sequential) 혹은 절차적(procedural)으로 접근할 수밖에 없다. 요컨대, 이것은 문제 해결 과정을 논리적 추론으로 설명하려는 규범 모형의 전형적인 예이다.

그런데 지금까지의 심리학 연구 결과들은 이러한 규범 모형이 인간의 실제 추리를 적절하게 기술하고 있는지에 대해 회의적이다. 사람들은 논리학, 수학, 그리고 통계학의 형식 규칙에 따라 추리하도록 훈련되어질 수는 있으나, 이것이 그들이 일상 생활에서 추리하는 방식은 아니다. 오히려 사람들은 특정 상황의 구체적 맥락 속에서 허용 도식(permission schema) 혹은 심적 모형(mental model) 등을 이용하여 일상적인 추리 과정을 진행시켜 나가는 것 같다 [Cheng & Holyoak, 1985][Johnson-Laird & Steedman, 1978][Cosmides, 1989]. 결국 규범 모형은 그 이름이 말해 주듯, 인간이 무엇을 하는지를 기술하기보다는 인간이 실제로 무엇을 하는지를 비교할 수 있는 참조의 틀을 제공하는 역할을 하고 있다고 보인다.

한편, DNA 컴퓨팅은 DNA 분자들을 사용해서 정보 처리를 수행하려는 시도이다 [장병탁, 2002]. 이것의 역사는 1994년 에이들만(L. M. Adleman)이 DNA를 이용하여 해밀토니안 경로 문제(Hamilton-

ian Path Problem, HPP)를 해결한 것 [Adleman, 1994]에서부터 사실상 시작되며, 이후 지금까지 계산학적으로 난제인 NP-완전(NP-complete) 문제들의 풀이에 관해 특히 많은 연구들이 있어 왔다 [Setubal & Meidanis, 1997]. 왜냐하면 DNA 컴퓨팅은 그것의 가장 큰 특징인 대규모 병렬 처리(massive parallel processing)로써 주어진 NP-완전 문제의 상태 공간(state space) 내 모든 상태들을 탐색하는 것을 가능하게 해 주는 장점을 갖고 있기 때문이다.

이에 필자는 이러한 DNA 컴퓨팅을 이용한 새로운 접근 방식을 원래의 규범 모형에 대한 대안적 경쟁 모형으로 제시하고자 한다. 본 논문에서는 먼저 원숭이와 바나나 문제에 대한 고전적 인공지능의 해결 방식을 살펴보고, 이에 수반되는 절차적 관점의 도입에 대해 토론한다. 그리고 이 문제에 대해 DNA 컴퓨팅을 이용한 해법을 제시함으로써, 절차적 관점에서 벗어난 추론 과정의 구현이라는 측면에서 필자의 방식이 인공지능을 추구하는 원래의 목적에 좀 더 충실한 시도일 수 있다는 가능성을 보이고자 한다. 또한 이 새로운 방식은 사람의 행동과 유사하게, 최적해 외에도 다양한 해법들을 생성해 낼 수 있다는 사실을 이끌어 냄으로써, 일반적인 문제 풀이 과정에 존재하는 어떤 인지적인 특성을 포착하고자 한다. 이것은 물론 원숭이와 바나나 문제의 목표 자체를 기존의 시각과는 다른 관점에서 설정하고, 여기에 병렬 처리 기반의 접근 방식을 적용한 데에 결정적으로 의존한다.

1.2. 연구의 범위 및 접근 방법

DNA 컴퓨팅을 이용하여 논리학 추론을 수행하려는 시도는 본 연구 이전에도 발표된 바 있다. 예컨대 여기에는 명제 논리 [Lee et al., 2003][Wasiewicz et al., 2000] 및 삼단 논증 [박의준 외, 2002], 그리고 혼 절(Horn-clause) 논리 정리(theorem) 증명 [Mihalache, 1997][Kobayashi, 1999][Uejima et al., 2001] 등이 있다. 그런데 이들 중 혼 절 논리 정리 증명 연구의 경우는 어느 것도 완벽하다고 보기 어렵다. [Mihalache, 1997]의 문제점은 일정 수 이상의 변수가 포함된 식, 그리고 어느 정도 이상 복잡한 식은 계산할 수 없다는 것이고, [Kobayashi, 1999]의 단점은 원치 않는 에러가 발생할 가능성이 높다는 것이며, [Uejima et al., 2001]의 난점은 지나치게 많은 실험 단계를 요구한다는 것이다. 따라서 이들 모두는 DNA 분자들을 이용한 실제 실험으로 구현되지 못한 상태라는 점에서 공통적이다.

이와는 달리 본 논문에서는 혼 절 논리의 정리 증명을 시뮬레이션 외에 실제 실험을 통해서도 수행하고자 한다. 현재 DNA 컴퓨팅 분야의 기술 수준으로는 한 번의 실험에서 구체적인 하나의 문제만을 해결할 수 있으므로, 필자는 제 1.1절에서 설명된 바대로 규범 모형을 설명하는 특정 사례인 원숭이와 바나나 문제만을 오직 본 연구에서 다룬다. 따라서 여기에 암묵적으로 내재되어 있는 가정은 의미 있을 정도로 상당한 수의 혼 절 논리의 증명 문제들에 대해 본 논문의 접근 방식이 적어도 원리적으로는 또한 적용될 수 있다는 것이다. 왜냐하면 만약 그렇지 못할 경우, 본 논문의 결과는 지나치게 사소한 것이 되기 때문이다. 그러나 이 가정에 대한 정당화는 그 자체로 하나의 큰 문제이므로, 여기서는 다루지 않기로 한다.

1.3. 논문의 구성

본 논문의 나머지 부분은 다음과 같다. 제 2장에서는 원숭이와 바나나 문제가 심리학의 관점과 인공지능의 관점에서 각각 소개된다. 제 3장에서는 이 문제에 대한 전통적인 접근 방식을 소개한 후 이 방식의 문제점을 지적함으로써 본 논문에서 사용된 DNA 컴퓨팅 기반 접근 방식의 필요성을 도출해낸다. 제 4장에서는 원숭이와 바나나 문제를 두 종류의 컴퓨터 시뮬레이션, 그리고 DNA 분자들을 사용한 실제 실험을 통해 해결하고, 이어 이들 결과가 갖는 의미, 제기될 수 있는 의문 및 연구 과제 등을 종합적으로 논의한다. 마지막으로 제 5장에서는 지금까지의 연구 결과를 요약·정리하고 DNA 컴퓨팅 분야에 대한 향후 전망을 조망하는 것으로 본 논문을 마무리한다.

2. 원숭이와 바나나 문제

심리학사를 통해 볼 때, 원숭이와 바나나 문제(Monkey and Banana Problem, 이하 MBP)는 1920년대 후반 독일의 행태주의 심리학자 쾰러(W. Köhler)의 연구에서 최초로 제시되었다고 알려져 있다 [Köhler, 1956]. 이것으로부터 우리는 일반적인 문제 해결 과정의 의미를 조망할 수 있다.

쾰러는 제 1 차 세계 대전 중 카나리아 제도의 한 섬에서 포획된 침팬지 무리를 발견하고, 이들의 문제 해결 행동에 관심을 가졌다. 이들 중 술탄(Sultan)이란 이름의 침팬지에게 준 문제는 두 개의 막대를 이용하여 우리 밖에 있는 바나나를 취하는 것이었다. 그런데 불행히 그 어느 것도 바나나에 미치지 않았다. 실망하여 우리 안에 가만히 앉아 있던 술탄은 갑자기 막대가 있는 곳으로 가더니 한 막대의 끝을 다른 쪽 막대의 끝에 끼워 먹이에 미칠 정도로 충분히 늘이고, 이것으로 바나나를 끌어당길 수 있었다.

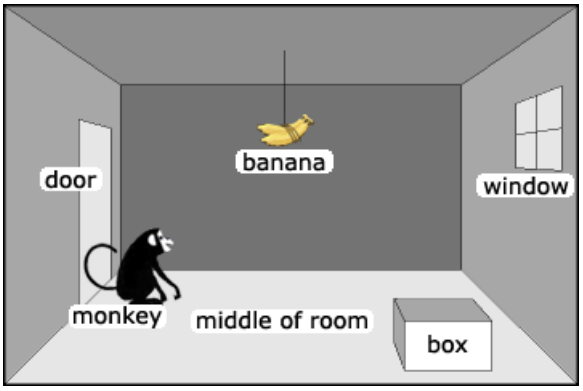
이 일화는 여러 가지 관점—예컨대 창의성의 창발—에서 해석 가능하다. 그러나 우리는 이로부터 일반적인 문제 해결 과정의 핵심적인 특징을 다음과 같이 세 가지로 요약할 수 있다:

(1) 목표 지향성
(2) 원래 목표를 하위 목표로 분할
(3) 연산자(operator) 적용 통한 문제 상태의 전이

요컨대 전체 문제 해결 과정은 분할된 하위 목표들을 각각에 대응하는 연산자들을 사용하여 해결하는 과정들의 연속이다.

한편 MBP는 사람의 사고를 모의 실험하려는 일반적인 문제 풀이기(General Problem Solver)의 동작을 설명하기 위해 인공지능과 관련된 여러 문헌에서 또한 자주 사용되는 예제이기도 하다. 이것은 제 2.1 절에서와는 다소 다르게 다음과 같이 주어지기도 한다 [이광형, 1996]:

> 어떤 방에 원숭이, 상자, 그리고 바나나가 있다. 바나나를 얻기 위해 원숭이는 어떤 동작들을 수행하여야 하는가? 즉 원숭이는 상자 있는 곳으로 가서(walk), 상자를 바나나 있는 곳까지 밀고(push), 상자 위로 올라가(climb), 바나나를 잡아야(grasp) 하는 것이다.



(그림 1) 원숭이와 바나나 문제

이 문제가 가정하고 있는 세계의 각각의 가능한 상태는 다음과 같이 벡터로 표현된다:

(w, x, y, z)

(단, 여기서 w는 원숭이의 위치(2차원), x는 원숭이가 상자 위에 있는가 여부(0 또는 1), y는 상자의 위치(2차원), z는 원숭이가 바나

나를 가졌는가 여부(0 또는 1)이다.)

이제 이 벡터에 대해 다음과 같은 4개의 연산을 정의함으로써 원숭이의 행동을 나타낼 수 있게 된다:

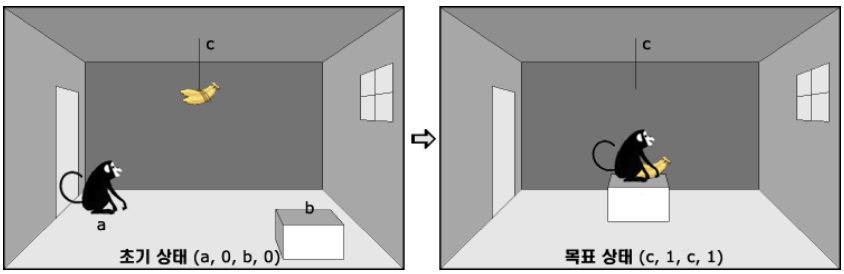
① walk(u): $(w, 0, y, z) \rightarrow (u, 0, y, z)$

② push(v): $(w, 0, w, z) \rightarrow (v, 0, v, z)$

③ climb: $(w, 0, w, z) \rightarrow (w, 1, w, z)$

④ grasp: $(c, 1, c, 0) \rightarrow (c, 1, c, 1)$

따라서 초기 상태와 목표 상태는 (그림 2)와 같다. 여기서 원숭이, 상자, 바나나의 초기 위치 조건은 각각 a와 b와 c이다.



(그림 2) 초기 상태와 목표 상태

3. 고전적 인공지능에서 본 MBP

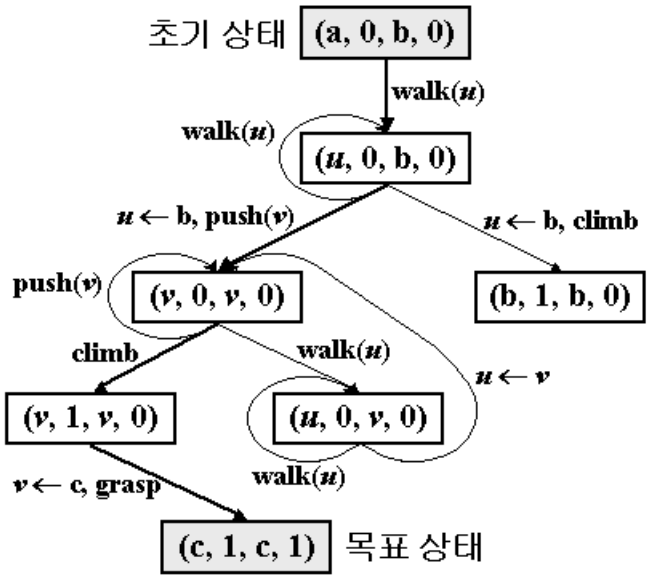
3.1. 일반적인 고전적 인공지능에서의 MBP 표현

MBP에 있어서 가장 중요하게 고려해야 할 사항은 원숭이나 상자의 위치를 나타내는 무한한 값(2차원 벡터)들을 어떤 식으로 해결할 것인가 하는 점이다. 이에 대해서는 두 가지 방법을 생각해 볼 수 있다:

방법 (1) 방을 일정한 간격으로 등분함으로써 위치들의 집합을 유한하게 만든다. 그 결과 우리는 유한한 상태 공간을 얻게 된다. 그러나 적당히 실용적인 간격을 취한다 해도 상태 공간은 지극히 커질 수 있다.

방법 (2) 상태를 표현하기 위해 틀 변수(schema variable)를 사용한다. 다시 말해, 위치를 나타내는 상수(constant)는 변수(variable)로 대치될 수 있고, 변수는 또한 다른 변수로 대치되거나 혹은 상수로 예화(instantiation)될 수 있다.

고전적 인공지능에서는 어떤 문제에 대해서도 항상 순차적으로 접근하므로, 방법 (1)의 적용은 현실적으로 힘들다. 왜냐하면 상태 공간의 크기가 커짐에 따라 목표 상태에 이르는 풀이 경로(solution path)의 길이는 급격하게 증가하게 되기 때문이다. 따라서 틀 변수의 사용만 지원이 된다면, 방법 (2)를 적용시키는 편이 보다 바람직하다. 방법 (2)의 적용 하에서 상태 공간의 탐색을 통한 풀이 경로는 다음 (그림 3)에서의 굵은 화살표로 표시된다:



(그림 3) 원숭이와 바나나 문제의 그래프

3.2. Prolog 구현을 통한 MBP 표현 및 해결

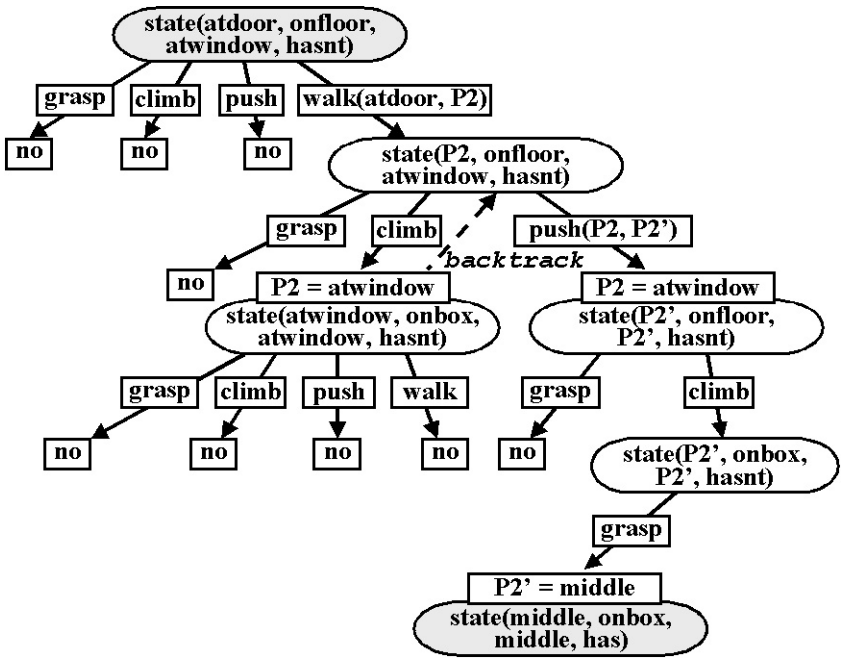
Prolog는 LISP와 더불어 고전적 인공지능 연구자들이 즐겨 이용하는 대표적인 언어이다. 이것은 주어진 문제에 대해 선언적(declarative) 접근, 즉 목표 지향적(goal-oriented) 프로그래밍을 가능하게 하므로, 원숭이와 바나나 문제를 해결하는 데 매우 적합한 도구이다. 나아가 Prolog의 추론 기제는 기본적으로 패턴 매칭(pattern matching)과 자동 백트래킹(backtracking)에 기반하고 있으므로, 구현된 Prolog 프로그램은 제 3.1절 방법 (2)에서의 틀 변수의 사용을 만족스럽게 지원한다. Prolog 프로그램의 소스 코드는 다음과 같다 [Bratko, 1990]:

```
% move( State1, Move, State2): Move in State1 results in State2;
% a state is represented by a term:
% state( MonkeyHorizontal, MonkeyVertical, BoxPosition, HasBanana)

move( state( middle, onbox, middle, hasnot),    % Before move
      grasp,                                    % Grasp banana
      state( middle, onbox, middle, has) ).     % After move

move( state( P, onfloor, P, H),
      climb,                                    % Climb box
      state( P, onbox, P, H) ).

move( state( P1, onfloor, P1, H),
      push( P1, P2),                            % Push box from P1 to P2
      state( P2, onfloor, P2, H) ).

move( state( P1, onfloor, B, H),
      walk( P1, P2),                            % Walk from P1 to P2
      state( P2, onfloor, B, H) ).

% canget( State): monkey can get banana in State

canget( state( _, _, _, has) ).                 % can 1: Monkey already has it

canget( State1) :-                              % can 2: Do some work to get it
  move( State1, Move, State2),                  % Do something
  canget( State2).                              % Get it now
```

(그림 4) 원숭이와 바나나 문제 프로그램

여기서는 (그림 2)에서와 같이 목표 상태를 state(middle, onbox, middle, has)로 가정하고 있다. 이 프로그램은 "?- canget(state(atdoor, onfloor, atwindow, hasnot))."라는 질의에 대해 "Yes"라고

대답한다. 즉, 이것은 초기 상태 state(atdoor, onfloor, atwindow, hasnot)로부터 목표 상태 state(middle, onbox, middle, has)에 이르는 경로가 존재함을 추론해 낸 것이다. 이 때, 추론해 낸 경로는 최단 경로이며, 사실상 이 프로그램은 최단 경로밖에 찾을 수 없다. 이 과정을 그림으로 표현하면 (그림 5)와 같다:



(그림 5) Prolog로 구현된 경우의 추론 과정([Williams]에서 발췌)

3.3. 고전적 인공지능의 접근 방식이 갖는 문제점

앞서 제 3.2절에서는 고전적 인공지능의 관점에 충실하게 상태 공간의 순차적인 탐색을 통해 목표 상태에 도달할 수 있었다. 그런데 실제로 Prolog 프로그램이 성공적으로 수행되는 데에는 그것이 갖는 선언적 측면 이외에 프로그램을 구성하고 있는 절(clause)들의 순서가 결정적인 역할을 한다. 예컨대 'walk' 가 포함된 절이 가장 처음에 나오도록 앞의 프로그램을 수정하면, 이 수정된 프로그램은 무한 루프에 빠지게 됨을 쉽게 알 수 있다. 즉, 이 경우 불쌍한 원숭이는 방 안을 무한정 이리저리 헤매게 되는 것이다.

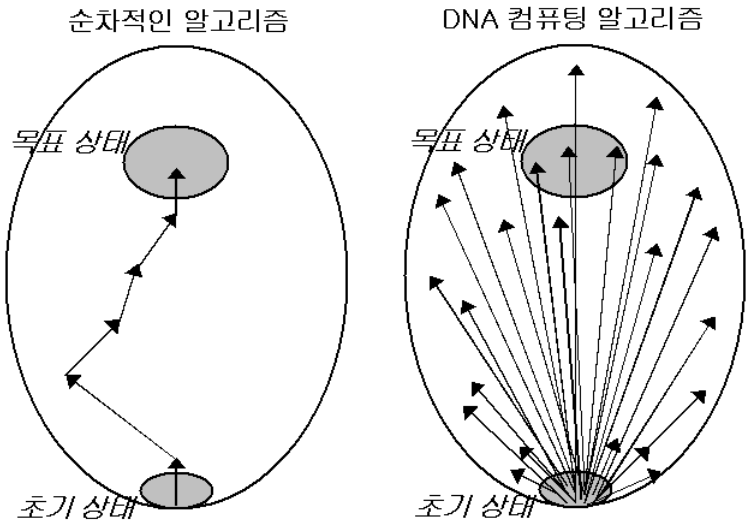
선언적 관점 외에 절차적 관점을 필요로 한다는 것은 우리가 인공지능을 추구하는 본연의 목적을 충족시키지 못함을 의미한다. 왜냐하면 선언적 언어인 Prolog를 사용하여 문제를 해결하려는 시도가 갖는 근본적인 의의는 사람은 문제를 해결하기 위해 필요한 지식들만을 제공하고, 실제 추론은 인공적인 추론 기관이 담당하게끔 하는 데 있기 때문이다. 다시 말해, 절차적 관점의 도입은 추론 과정에 인간의 사고 작용을 필연적으로 개입시키게 되는데, 이것은 바로 고전적 인공지능의 의미를 퇴색시키는 일인 것이다. 그리고 적어도 원숭이와 바나나 문제의 경우, 이것은 순차적 접근이라는 틀을 유지하면서 제 3.1절의 방법 (2)를 적용시키려는 고전적 인공지능 일반의 한계이다. 결코 제 3.2절에서 소개된 Prolog 프로그램만의 문제가 아닌 것이다. 물론 고전적 패러다임 아래에서도 절들의 순서에 구애받지 않고 무한 루프를 배제시키는, 일반적이고 신뢰할만한 여러 방법들이 있다 [Bratko, 1990]. 그러나 각종 테크닉에 의존하는 이런 방법들이 아닌, 보다 근본적인 해결책은 과연 존재하지 않는 것일까?

본 논문의 문제 의식은 바로 이 점에 있다. 에이들만이 [Adleman, 1994]에서 DNA 컴퓨팅을 이용한 해결책을 제시했던 것은 근본적으로 해밀토니안 경로 문제의 경우, 방문해야 할 점(point)들의 개수가 증가함에 따라 가능한 경로의 수가 폭발적으로 증가한다는 사실에 기인한다. 반면 원숭이와 바나나 문제는 이런 계산학적인 관점과는 다소 다른 시각에서 DNA 컴퓨팅의 새로운 의미를 드러나게 해 준다. 그것은 인공지능을 구현하기 위한 바람직한 방법론에 관한 문제 제기이다.

3.4. DNA 컴퓨팅을 이용한 접근 방식의 필요성

고전적 인공지능에서 제 3.1절의 방법 (1)이 고려되지 않은 것은 계산학적인 이유에서이다. 실용적 관점에서는 무한 집합을 다룰 수 없다는 것은 심각한 문제가 아니다. 중요한 것은 입도(granularity)이므로, 필요한 만큼 얼마든지 큰 유한 집합으로 대신하면 되기 때문이다. 이 맥락에서 정말로 심각한 문제는, 유한하지만 매우 큰 상태 공간을 탐색하는 데 걸리는 비용—시간과 메모리이다.

DNA 컴퓨팅 기법은 이 문제를 해결할 수 있는 가능성을 제공한다. DNA 컴퓨팅은 기본적으로 대규모 병렬 처리이므로, 원숭이와 바나나 문제의 상태 공간의 크기와 무관하게 상태 공간 내 상태들을 동시에 탐색할 수 있다. 이러한 점은 다음의 (그림 6)으로 쉽게 설명될 수 있다 [Zhang & Shin, 1998].



(그림 6) 순차적인 알고리즘과 DNA 컴퓨팅 알고리즘의 비교

그러나 DNA 컴퓨팅 기법으로 제 3.1절의 방법 (2)을 사용하는 것, 즉 상수나 변수에 대한 대치와 예화를 구현하는 것은 현재의 기술 수준으로는 곤란하다. 방법 (2)를 사용하기 위해서는 무엇보다도 DNA 시퀀스에 대한 임의의 돌연변이(mutation) 연산을 필수적으로 요구하는데, 이는 지금으로서는 부분적으로 가능하기 때문이다. 따라서 본 논문에서는 DNA 컴퓨팅 기법에 기반한 방법 (1)을 사용하여 원숭이와 바나나 문제를 해결하고자 한다. 이 방법은 Prolog를 사용한 방식과는 달리, 가능한 유한 개의 모든 상태들과 그에 따른 연산들을 DNA로 미리 코딩(coding)해 놓는 대신에 추론 과정에는 전혀 관여하지 아니한다. 물론 실제적인 비용의 측면에서 볼 때, 현재로서는 Prolog를 사용한 방식보다 바람직하지 않다는 점은 명백하다. 그러나 오히려 인공지능 본연의 의미를 추구하는 데에는 더 적합한 접근 방식이라고 생각한다.

4. DNA 컴퓨팅 기법을 이용한 MBP 표현 및 해결

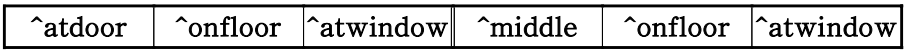
4.1. DNA 컴퓨팅을 위한 MBP 표현 방법

본 논문에서는 유한한 상태 공간을 유지하기 위해, 원숭이의 위치와 상자의 위치가 취할 수 있는 값을 '창 옆(atwindow)', '방문 옆(atdoor)', '방 가운데(middle)'의 3가지로 제한하기로 한다. 앞 절에서 말했듯이, 상태 공간의 크기는 만약 필요하다면 얼마든지 늘일 수 있으므로, 이러한 식의 제한이 문제 풀이의 일반성을 잃게 만드는 것은 아니다. 또 상태들의 개수를 줄이기 위해, 원숭이가 바나나를 가졌는지 여부는 무시하기로 하자. 즉, 초기 상태에서 원숭이는 당연히 바나나를 갖고 있지 않고, 이것은 목표 상태에 이르기 직전까지 마찬가지이므로, 우리는 목표 상태 직전의 상태 (middle, onbox, middle, hasnot)를 새로운 목표 상태로 간주하여, 초기 상태로부터 이 새로운 목표 상태에 이르는 것으로 원래의 문제를 재구성할 수 있다. 이렇게 되면, 상태 벡터의 4 번째 요소는 불필요하게 되므로 모든 상태들의 개수는 12 개가 된다(원숭이가 방바닥 위에 있는 경우: 9 (= 3 × 3)개, 상자 위에 있는 경우: 3개). 또 walk 연산은 방바닥 위(onfloor)에서만 가능하므로 모두 18 (= 9 × 2)개이고, push 연산은 원숭이의 위치와 상자의 위치가 동일할 때 방바닥 위에서만 가능하므로 6 (= 3 × 2) 개이며, climb 연산은 원숭이의 위치와 상자의 위치가 동일할 때 상자 위(onbox)에서만 가능하므로 3 개이다. 이제 grasp 연산은 당연히 불필요하다.

하나의 상태에 대한 DNA 시퀀스는 기본적으로 3 개의 단위를 필요로 한다. 예컨대 상태 (atdoor, onfloor, atwindow)는 (그림 7)과 같이 나타낼 수 있다:

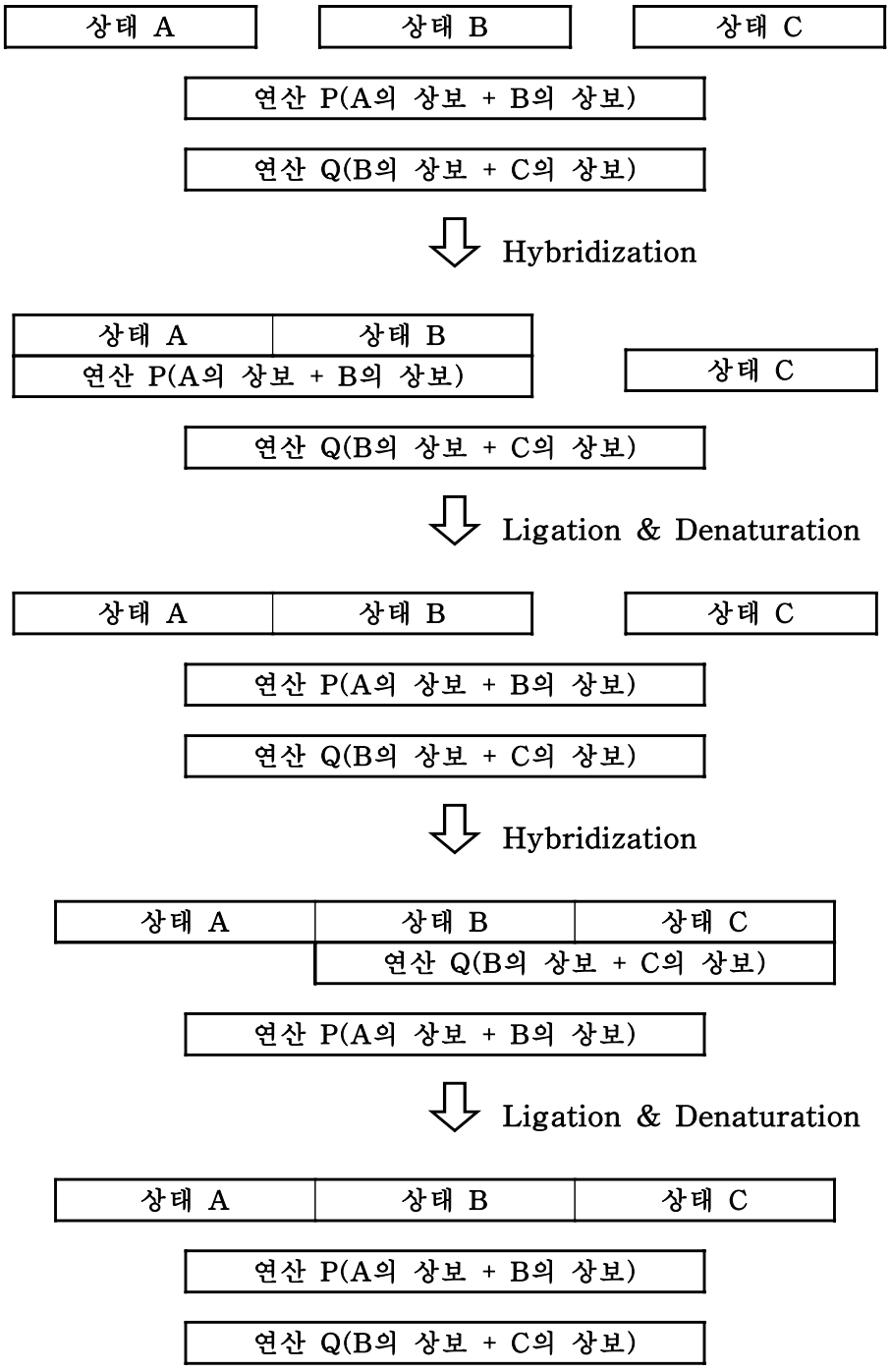
(그림 7) 하나의 상태에 대한 DNA 시퀀스

연산은 두 개의 상태를 매핑(mapping)시켜 주는 역할을 하므로 6 개의 단위를 필요로 한다. 예컨대 상자가 창 옆에 있을 때 원숭이가 방문 옆에서 방 가운데로 걸어가는(walk) 연산은 다음 (그림 8)과 같이 나타낼 수 있다(단, '^atdoor', '^onfloor', '^atwindow', '^middle'는 각각 'atdoor', 'onfloor', 'atwindow', 'middle'의 상보적 염기 서열을 뜻한다). 즉, 하나의 연산에 대한 시퀀스는 하나의 상태를 나타내는 그것보다 2 배로 길고, 따라서 그 연산의 가능한 논항 쌍에 대응하는 두 시퀀스와 완벽한 Watson-Crick 상보 결합을 이룰 수 있다.



(그림 8) 하나의 연산에 대한 DNA 시퀀스

이와 같은 방법으로 원숭이와 바나나 문제를 해결하기 위해 요구되는 지식들을 추상적인 수준에서 DNA로 표현해 내었다. 이제 남은 것은 추론 과정을 거쳐 실제로 문제를 해결하는 일이다. 이 과정을 도식적으로 나타내면 (그림 9)와 같다:



(그림 9) 추상적인 관점에서 본 문제 해결 과정

상태 A를 상태 B로 전이시키는 연산 P가 존재하는 경우 Hybridization을 통해 P에 해당하는 연산-시퀀스를 매개로 하여 두 상태-시퀀스는 서로 인접하게 되며, Ligation 및 Denaturation을 거쳐 두 상태-시퀀스는 연결된다. 유사하게, 상태 B를 나타내는 시퀀스는 상태 C를 나타내는 그것과 다시 연결될 수 있다. 이러한 과정을 여러 번 반복함으로써 상태들의 연속체를 표현하는 DNA 시퀀스들이 생성된다. 이제 이렇게 생성된 시퀀스들 중 한쪽 끝은 초기 상태를, 그리고 다른 쪽 끝은 목표 상태를 나타내는 것들만이 원하는 해가 된다. 따라서 해를 도식적으로 나타내면 (그림 10)과 같다:

상태 1	상태 2	상태 3	……	상태 N-1	상태 N

(그림 10) 최종 해에 대한 DNA 시퀀스

단, 여기서 초기 상태인 상태 1과 목표 상태인 상태 N은 각각 (그림 11)의 위 그림, 아래 그림과 같다:

atdoor	onfloor	atwindow	상태 1
middle	onbox	middle	상태 N

(그림 11) 초기 상태, 목표 상태에 대한 DNA 시퀀스

4.2. 시뮬레이션 1

4.2.1. DNA 시퀀스 부호화

어떤 상태에 대한 DNA 시퀀스를 실제로 코딩하기 위해 다음 두 가지 방법을 생각해 볼 수 있다: (1) 상태 벡터를 구성하는 각각의 성분들을 DNA 시퀀스로 나타내고, 이러한 성분-시퀀스들의 연속체로 하나의 상태를 표현해 낸다(제 3절의 표현 방법). 곧, 원숭이 및 상자의 위치를 나타내는 3 종류의 시퀀스들과 원숭이가 상자 위에 있는가 여부를 나타내는 2 종류의 시퀀스들을 차례로(원숭이 위치-원숭이가 상자 위에 있는가 여부-상자 위치) 연결하여 하나의 상태를 표현해 내는 방법이다. (2) 구성 성분에 관계없이, 12개의 상태들을 각각 서로 다른 DNA 시퀀스들로 나타낸다.

첫 번째 방법은 생성된 성분-시퀀스들의 연속체들, 곧 상태를 나타내는 시퀀스들이 가능한 한 서로 유사하지 않도록 애초에 각각의 성분에 해당하는 시퀀스를 디자인해야 하기 때문에, 실제 코딩이 어렵다는 단점을 갖는다. 그러나 서로 다른 상태 i, j에 공통적인 성분이 존재하는 경우, 두 상태 i, j를 나타내는 DNA 시퀀스가 유사하므로, 실제 반응 시 두 시퀀스가 비슷한 행동 양상을 보이게 되기 쉽다는 장점이 있다. 반면에 12개의 상태들을 각각 서로 다른 DNA 시퀀스들로 표현하는 방법은 디자인이 쉽다는 장점이 있으나, 동일한 성분을 포함하는 상태들 간의 유사성을 표현하기 어렵다는 단점이 있다.

상태들 간의, 곧 시퀀스들 간의 유사성은 DNA를 이용한 실제 실험에서는 반응의 효율성이란 측면에서 중요한 특징이나, 컴퓨터 시

뮬레이션에서는 그다지 고려할 필요가 없다. 본 논문에서는 오직 컴퓨터 시뮬레이션을 통해 결과를 확인하고자 하므로, 두 번째 방법을 써서 DNA 시퀀스를 디자인하도록 한다. 실제로 이 작업은 서울대학교 바이오지능 연구실에서 개발한 DNA 컴퓨팅 시뮬레이터인 NACST(Nucleic Acid Computing Simulation Toolkit) [Zhang & Shin, 1998]를 써서 수행되었다. NACST는 시뮬레이션에 사용되는 DNA 시퀀스들을 진화 알고리즘(Genetic Algorithm)을 이용해 최적화시킨다. <표 1>에는 각각의 상태를 나타내는 12 개의 시퀀스들이 정리되어 있다(a: atdoor, b: atwindow, c: middle).

상태 (상태번호)	시퀀스(5′ - 3′)
a 0 a (0)	GACCCTGCTG GAGAGACTAC CCCTTTAGAC
a 0 b (1)	GATCTCAGAC ACAGAATTGT GTCGACTAGG
a 0 c (2)	CTTCGTCGTC TCCGACAGGA ATGAATTGCA
a 1 a (3)	GTAGTTCATG CCACAAGCAC ACTATTTGGA
b 0 a (4)	CAACTTGAGA TTGTCACGGC TGGGCGAGGC
b 0 b (5)	GTTCGTTGCG GTTCCTCGGC CCTAAAAGTC
b 0 c (6)	GGCCTTGAAG CCGGCAAGTT AAATAAAGTG
b 1 b (7)	ATGCACTATG ATTGCGCTCT AGTCACACAC
c 0 a (8)	TCTGCGGCAG CGAGATATGT ATCATGGTAA
c 0 b (9)	CAGCAGGGGT TAGTCAGTAG CGGGGTTTTA
c 0 c (10)	TTCCTAGGCG TATGACTAGA GGAATGAGGC
c 1 c (11)	ATTTCGAGCG GCAATTCGTT CAAGATGTAT

<표 1> NACST를 이용하여 생성된 12개의 상태-시퀀스들

4.2.2. 시뮬레이션 설계 및 절차

DNA 컴퓨팅에 대한 시뮬레이션 과정을 설계하면 다음과 같다:

(1) **Hybridization & Ligation**: 원숭이가 이동할 수 있는 다양한 경로 생성

(2) **Affinity separation**: 초기 상태와 목표 상태를 모두 포함하는 경로 분리

(3) **PCR** : 위의 단계 (2)에 부합하는 경로 증폭

(1) 먼저 하나의 튜브 안에 가능한 모든 상태-시퀀스들(12 개)과 모든 연산-시퀀스들(27 개)을 넣고 Hybridization과 Ligation, 그리고 Denaturation을 반복적으로 수행한다. 그 결과, Hybridization과 Ligation을 통해 상태들 간의 연결이 이루어지고, Denaturation을 통해 상태-시퀀스들과 경로-시퀀스들이 다시 분리되어 다음 사이클(cycle)에서의 반응을 준비한다. 이어 다시 Hybridization 및 Ligation을 거쳐 이전 사이클에서 생성된, 연결된 상태-시퀀스들은 이제 튜브 내의 다른 상태-시퀀스들 및 경로-시퀀스들과 확률적으로 반응하게 된다. 이러한 사이클을 반복하면서 원숭이의 이동 가능한 경로들 중 상당수가 생성되게 된다.

(2) 다음 단계에서는 Affinity Separation이라는 실험 기법을 이용하여, 원숭이의 출발 상태와 목표하는 상태를 모두 포함하는 경로들만을 분리해 낸다. Affinity Separation은 Magnetic Bead에 특정한 시퀀스를 붙여서 테스트 튜브 내의 시퀀스들과 상보 결합반응을 일으킨 다음, Bead 상의 시퀀스와 결합한 시퀀스들을 그렇지 않은 시퀀스들로부터 분리해 내는 기법이다. 실제 실험에서 원하는 경로들

만을 분리해 내려면, 출발 상태에 해당하는 시퀀스와 상보 결합하는 시퀀스가 부착된 Bead를 이용하여 출발 상태를 포함한 경로들을 걸러낸 후, 다시 목표 상태에 해당하는 시퀀스와 상보 결합하는 시퀀스가 부착된 Bead를 이용하여 목표 상태를 포함하는 경로들을 걸러내면 된다.

(3) 마지막으로 출발 상태를 표현하는 시퀀스와 목표 상태를 표현하는 시퀀스를 각각 Primer로 사용하여 PCR을 수행한다. 이 단계를 통해 원하는 경로들만을 (만약 존재할 경우) 증폭하게 된다.

이상의 절차를 보다 자세히 기술하면 아래와 같다.

(1) Hybridization & Ligation

초기에는 12 개의 상태-시퀀스들과 27 개의 연산-시퀀스들만이 각각 균일한 농도로 존재하고, 한 차례의 Hybridization 및 Ligation을 통해, 두 개의 상태(혹은 경로)-시퀀스와 하나의 연산-시퀀스가 반응하여 하나의 긴 경로-시퀀스와 연산-시퀀스를 생성하는 것으로 가정하였다. 따라서 반응이 반복적으로 진행됨에 따라 짧은 경로-시퀀스들은 점차 줄어들고 상대적으로 긴 경로-시퀀스들이 새로이 생성될 것이다. 그리고 연산-시퀀스들의 농도에는 변화가 없을 것이다. 한 사이클 당 이러한 Hybridization 및 Ligation이 일정한 회수 ($REACTION^{h}_{MAX}$)만큼 일어나는 것으로 모델링하였다. 그리고 이 과정은 총 $CYCLE^{h}_{MAX}$ 사이클 동안 반복된다.

Hybridization 및 Ligation 과정을 자세히 살펴보면 다음과 같다: 우선 반응에 참여할 두 개의 경로-시퀀스는 현재까지 생성된 경로-시퀀스들 중에서 10 개를 무작위로 고른 다음 (중복 허용), 이 중에

서 현재 가장 높은 농도를 가진 경로-시퀀스를 고르는 일을 두 번 반복하여 선택하였다. 이 방식을 사용하면 현재 가상의 튜브 안에 다수가 존재하는 경로-시퀀스들이 Hybridization 및 Ligation에 참여할 확률이 높아지므로, 실제 반응 결과에 가까운 시뮬레이션 결과를 얻게 되리라 기대한다.

위의 과정을 통해 선택된 두 경로-시퀀스는 이들과 완벽하게 상보 결합할 수 있는 연산 시퀀스가 존재하는 경우에만 Ligation을 통해 보다 긴 경로-시퀀스를 만들 수 있다고 가정하였다. 연산-시퀀스와 상보 결합할 경우, 반응 전에 선택된 두 경로 시퀀스의 수는 각각 1 씩 줄이고, Ligation을 통해 생성된, 보다 긴 경로-시퀀스의 수는 1 만큼 늘리도록 하였다. 만약 새로 생성된 경로-시퀀스가 현재까지 생성되지 않은 것이라면, 이것을 현재까지 생성된 경로-시퀀스 목록에 추가하도록 하였다.

(2) Affinity Separation

여기서는 Bead에 부착된 시퀀스가 한 종류뿐이라고 가정하였다. 이 때, 가상의 튜브 내에 존재하는 모든 경로-시퀀스들은 (1) Bead에 붙은 시퀀스와 상보 결합할 수 있는 것들과 (2) 상보 결합할 수 없는 것들의 두 종류로 나뉠 수 있는데, 각각에 대하여 다음의 반응을 수행하였다:

(1)의 경우, 현재 튜브 내에 존재하는 수에 수율(yield)을 곱한 만큼의 개수를 가지도록 하였다. 그리고 (2)의 경우, 각각의 경로-시퀀스는 상태-시퀀스들의 연결로 이루어져 있으므로, 경로-시퀀스를 이루는 상태-시퀀스들 중 Bead에 붙은 시퀀스와 상보 결합할 확률이

가장 높은 부분을 찾은 다음, 그 지점에서 상보 결합할 확률 p_{hyb}을 계산하였다. 다음에, 0~1사이의 수를 임의로 생성하여 그것이 p_{hyb} 보다 작으면 (1)과 같은 반응을 수행하였고, 그렇지 않으면 현재 가상 튜브 내에 존재하는 수를 0으로 만들었다.

본 시뮬레이션에서는 이러한 Affinity Separation을 초기 상태와 목표 상태에 대하여 각각 한 번씩 거치도록 하였다.

(3) PCR

Primer로 시퀀스 $primer_1$(5′ 방향), $primer_2$(3′ 방향)가 주어진다고 가정하였다. 이 때, 가상 튜브 내의 경로-시퀀스들은 Primer와의 관계에 따라 다음과 같이 나누어진다:

(a) 경로-시퀀스의 가장 끝 부분과 Primer가 상보적인 경우
(b) 경로-시퀀스를 이루는 상태-시퀀스 중 하나와 Primer가 상보적인 경우
(c) 어느 상태-시퀀스와도 상보적이지 않은 경우

PCR 시뮬레이션 과정에서는 각각의 Primer에 대하여 위의 세 가지의 경우로 나누어 다음의 반응을 매 PCR 사이클마다 $REACTION_{MAX}^{PCR}$만큼 수행하였다. 그리고 이것을 일정한 사이클 수 ($CYCLE_{MAX}^{PCR}$)만큼 반복하였다.

(a)의 경우, 첫 부분과 $primer_1$이 상보적인 경우나 마지막 부분과 $primer_2$가 상보적인 경우에는 해당 경로-시퀀스의 개수를 1 증가시켰고, 나머지 경우에는 개수의 변화가 없게 하였다. 이것은 PCR 중

Extension에는 방향성이 존재하기 때문에 일정한 방향으로만 Extension이 일어나는 것을 모델링한 것이다.

(b)의 경우, 첫 부분에서 $primer_2$와 상보적인 부분까지, 또는 $primer_1$에서 상보적인 부분에서 끝부분까지에 해당하는 부분 경로-시퀀스가 생성되게 되는데, 이러한 경로-시퀀스가 이미 존재하는 경우에는 해당 시퀀스의 개수를 1 증가시켰고, 그렇지 않은 경우에는 기존의 경로-시퀀스 목록에 추가한 다음에 이것의 개수를 1 증가시켰다.

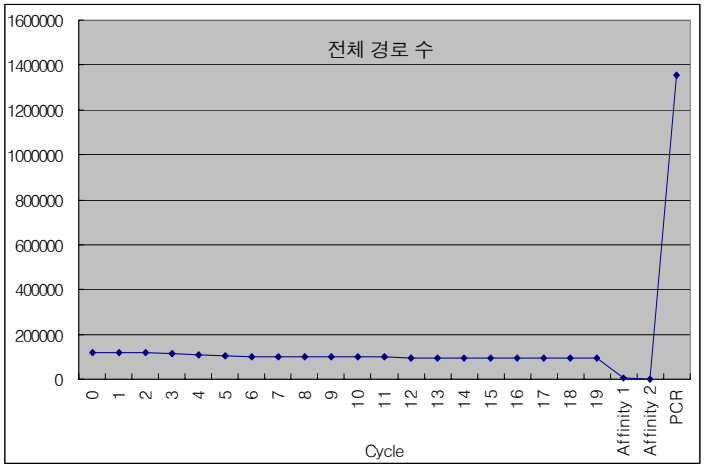
(c)의 경우, 제 4.3.2절 Affinity 과정의 Step 4와 비슷하게, 각 Primer와 가장 상보 결합할 가능성이 큰 부분 시퀀스를 찾아 (2)에서와 같이 부분 경로-시퀀스를 생성하여 처리한다. 이것은 PCR 과정 중에 발생할 수 있는 Primer와 경로-시퀀스들 간의 잘못된 반응도 모델링하기 위한 것이다.

4.2.3. 시뮬레이션 결과

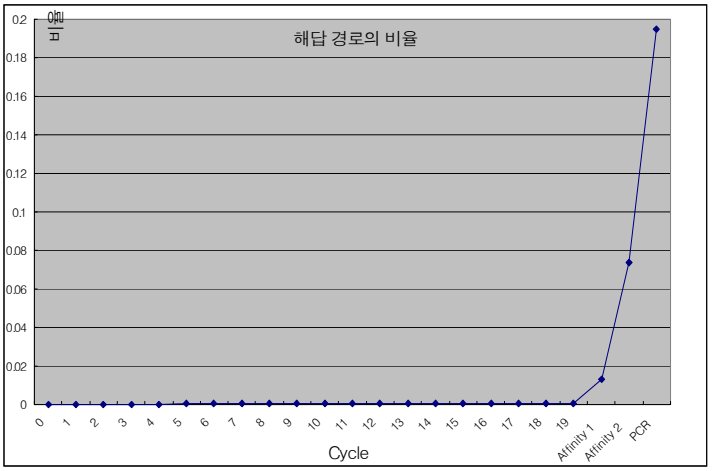
이번 시뮬레이션에서는 $REACTION^{h}_{MAX}=50000$, $CYCLE^{h}_{MAX}=20$, $REACTION^{PCR}_{MAX}=50000$, $CYCLE^{PCR}_{MAX}=20$, $YIELD=0.1$로 초기 조건을 설정하였다.

우선 시뮬레이션이 진행됨에 따른 전체 경로의 개수와 해답 경로의 비율은 (그림 12) 및 (그림 13)과 같다. (그림 12)에서 전체 경로의 개수는 처음에 주어진 상태-시퀀스들의 개수를 줄곧 유지하다가 Affinity Separation을 두 번 거친 후 급격히 줄어든 다음, PCR 후 다시 증가하는 것을 볼 수 있다. 또한 (그림 13)에서 해답 경로의

비율은 Affinity 전까지는 매우 낮은 비율이었지만, Affinity Separation을 두 번 거친 후 0.08까지 증가하였고, PCR 이후에는 0.2 정도까지 증가하였다. 따라서 본 연구의 시뮬레이션 결과는, 계획한 실험 단계를 거치면 해답 경로의 비율이 점차 증가할 것이라는 예측을 가능하게 해 준다.

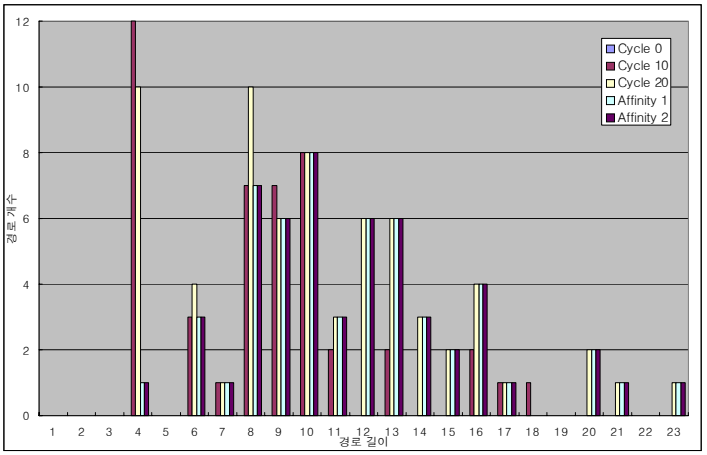


(그림 12) 전체 경로의 개수

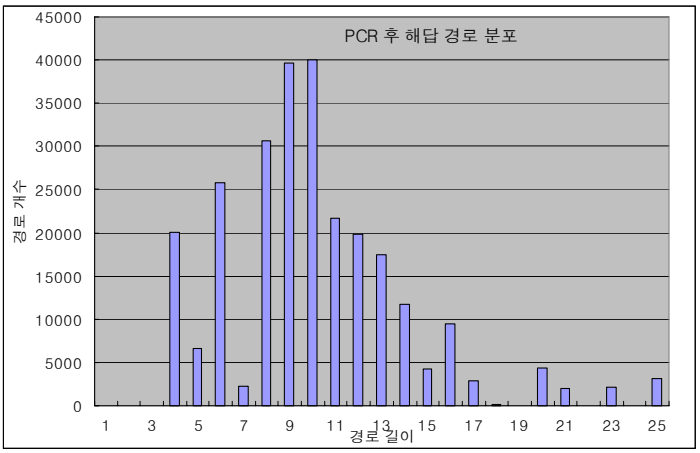


(그림 13) 전체 경로에 대한 해답 경로의 비율

(그림 14)와 (그림 15)는 각각 Affinity Separation 이후 생성된 경로들과 PCR 이후 생성된 경로들의 다양성을 나타내고 있다. 이러한 다양성은 DNA 컴퓨팅의 특징인 대규모 병렬 탐색의 결과이다. (그림 14)에서는 반응 초기(사이클 10)에 짧은 길이의 경로들이 많이 생기고, 사이클이 거듭될수록 보다 긴 길이의 경로들이 점차 발생하는 것을 볼 수 있다. 또한 (그림 15)에서는 Affinity Separation을 거치면서 상당히 적은 양으로 줄어든 짧은 길이의 해답 경로들의 개수도 PCR을 통해 많이 증가한 것을 볼 수 있으며, 해답이 아닌 경로에 한 부분으로 포함되어 있던 해답 경로(길이 25)가 새로이 생성된 것도 또한 관찰할 수 있다. 이것은 PCR 이전에는 아예 존재하지 않았던 경로이다.



(그림 14) 다양한 길이의 해답 경로들



(그림 15) PCR 후 해답 경로들의 분포

4.3. 시뮬레이션 2

4.3.1. DNA 시퀀스 부호화

제 4.1절에서 추상적인 수준으로 기술된 문제 해결 과정은 개념적으로는 명료하나, Hybridization, Ligation 및 Denaturation의 반복 적용을 포함하고 있어 이것을 실제 실험으로 구현하는 데에는 기술적으로 어려움이 많다. 따라서 성공적인 실험을 위해서는 문제 해결 과정의 구상을 가능한 한 현실적으로 수정할 필요가 있다. 일단 본 절에서는 실제 실험에 앞서 수행하고자 하는 실험과 보다 더 유사한 새로운 버전의 시뮬레이션을 설계하고자 한다. 구체적인 시뮬레이션 절차는 제 4.3.2절에 기술하기로 하고, 여기서는 새로 디자인된

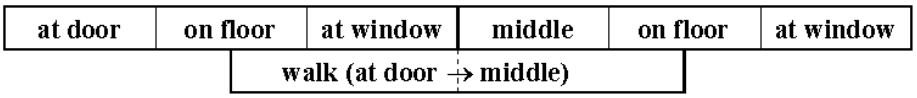
12개의 DNA 상태-시퀀스들만을 아래 <표 2>와 같이 제시한다. 이것은 NACST에서 생성된 <표 1>의 코드를 실험 휴리스틱에 근거하여 다소 간의 수정을 거친 것이다. 본 절의 시뮬레이션과 제 4.4절의 실험에는 모두 <표 2>의 시퀀스 디자인이 적용된다.

상태 (상태번호)	시퀀스(5′ - 3′)
a 0 a (0)	GACCCTGCTG GAGAGACTAC CCCTTTAGAC
a 0 b (1)	CCCGCCCTAG CCATTCAGCA CTGTTACGCA
a 0 c (2)	CTTCGTCGTC TCCGACAGGA ATGAATTGCA
a 1 a (3)	GTAGTTCATG CCACAAGCAC ACTATTTGGA
b 0 a (4)	CAACTTGAGA TTGTCACGGC TGGGCGAGGC
b 0 b (5)	GTTCGTTGCG GTTCCTCGGC CCTAAAAGTC
b 0 c (6)	GGCCTTGAAG CCGGCAAGTT AAATAAAGTG
b 1 b (7)	ATGCACTATG ATTGCGCTCT AGTCACACAC
c 0 a (8)	TCTGCGGCAG CGAGATATGT ATCATGGTAA
c 0 b (9)	CAGCAGGGGT TAGTCAGTAG CGGGGTTTTA
c 0 c (10)	CAAAACCTCT CCGAACAAAT CTAACCAAAT
c 1 c (11)	ATTTCGAGCG GCAATTCGTT CAAGATGTAT

<표 2> 실제 실험에 사용된 상태-시퀀스들

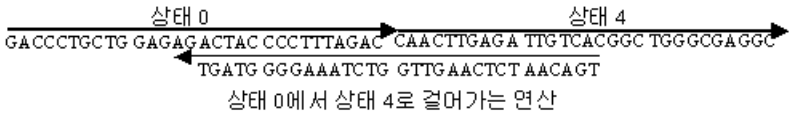
4.3.2. 시뮬레이션 설계 및 절차

Hybridization, Ligation 및 Denaturation을 단 한 차례만 수행하기 위해 연산-시퀀스에 대한 디자인을 수정한다. 예컨대 방문 옆에서 방 가운데로 걸어가는 연산을 도식적으로 나타내면 다음 (그림 16)과 같다:



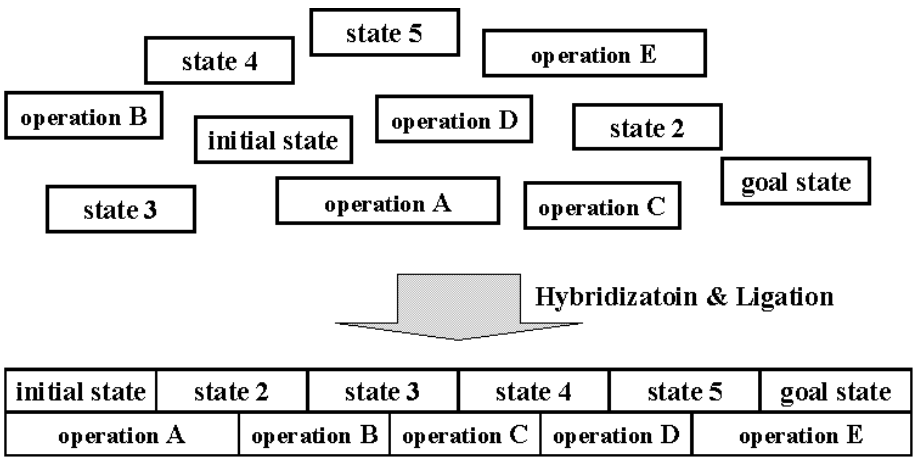
(그림 16) 연산-시퀀스 디자인에 대한 수정된 도식

곧, <표 2>에서 상태 0에서 상태 4로 걸어가는 연산에 대한 실제 DNA 시퀀스는 (그림 17)에서와 같이 나타내어진다.



(그림 17) 연산-시퀀스의 한 예

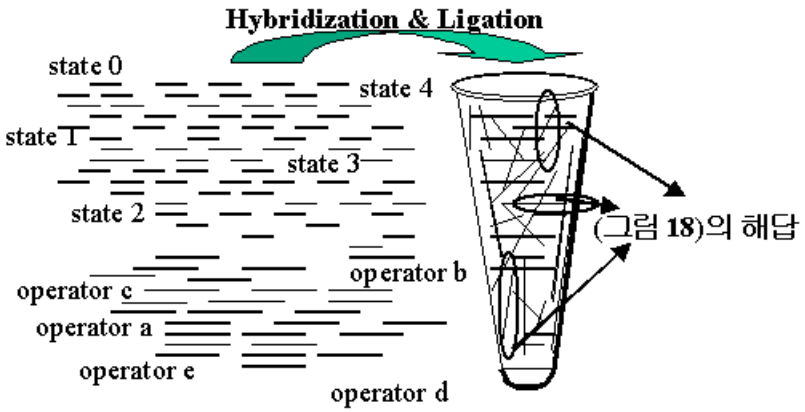
이렇게 상태-시퀀스의 길이와 연산-시퀀스의 길이를 같게 만들어 줌으로써, 제 4.2절의 시뮬레이션 1과는 달리 단 한 차례의 반응을 통해서도 (그림 18)에서처럼 MBP의 해가 생성될 수 있다:



(그림 18) 문제 해결 과정에 대한 수정된 도식

이러한 과정은 실제로 (그림 19)에서와 같이, 모든 종류의 DNA

시퀀스들이 혼합된 튜브 안에서 시퀀스들 사이의 대규모 상호 작용을 통해서 진행되게 된다.



(그림 19) 튜브 안 대규모 상호 작용을 통한 해답-시퀀스의 생성

새로운 시뮬레이션 과정을 대략적으로 기술하면 다음과 같다:

(1) DNA 시퀀스들 튜브에서 혼합

(2) Hybridization & Ligation: 원숭이가 이동할 수 있는 다양한 경로 생성

(3) PCR: 초기 상태와 목표 상태를 모두 포함하는 경로들만 분리

(4) Gel electrophoresis : PCR 결과 생성된 해들 확인

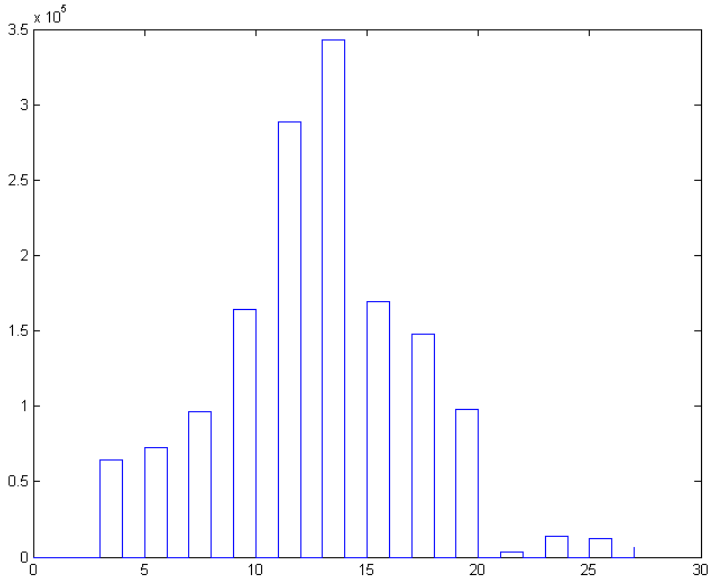
이 과정은 본 절의 시뮬레이션과 제 4.4절의 실험에 공통적으로 적용된다. 단, 시뮬레이션 상에서는 시퀀스들을 직접 확인하는 것이 가능하므로, 이 경우 단계 (4)는 불필요하다. 단계 (2)와 (3)에 대한 시뮬레이션 세부 사항은 제 4.2절에서와 대동소이하다.

4.3.3. 시뮬레이션 결과

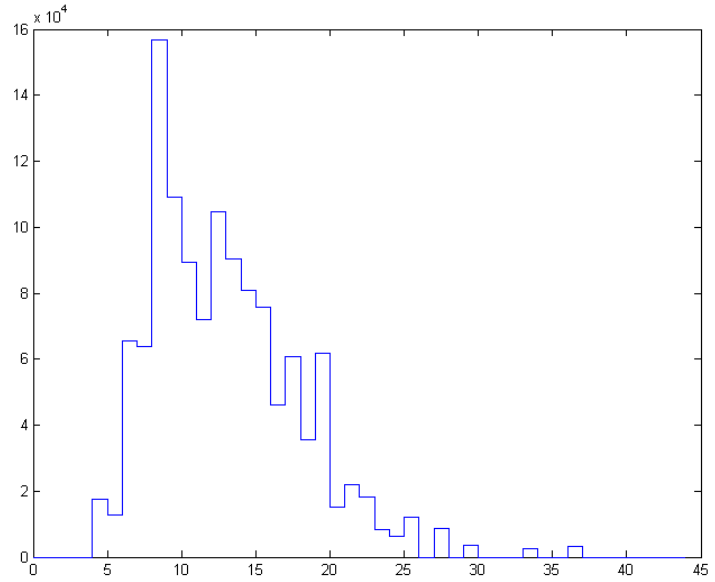
이번에도 초기 조건은 $REACTION_{MAX}^{PCR}=50000$, $CYCLE_{MAX}^{PCR}=20$, $YIELD=0.1$이다. 이 조건 하에서 다시 두 가지 종류—(1) 작은 규모와 (2) 큰 규모의 시뮬레이션을 각각 수행하였다. 큰 규모의 시뮬레이션에서는 지금까지와 마찬가지로, 원숭이의 위치와 상자의 위치가 취할 수 있는 값을 '창 옆(atwindow)', '방문 옆(atdoor)', '방 가운데(middle)'의 3 가지로 제한한데 반해, 작은 규모의 그것에서는 범위를 더 줄여 '창 옆(atwindow)'과 '방 가운데(middle)'의 2 가지로만 제한하였다. 이는 실험 비용의 절감을 위해서는 시퀀스 종류를 줄일 필요가 있기 때문이다. 그러나 제 3장에서 이미 설명한 바대로, 이러한 제약이 문제 해결 방식의 일반성을 잃게 하는 것은 아니며, 원숭이와 상자의 위치 값이 두 가지라 할 지라도 이로부터 역시 무수히 많은 종류의 경로-시퀀스들과 무수히 많은 종류의 해답-시퀀스들이 원리 상 생성될 수 있다. 예컨대 원숭이가 방문 옆과 상자 사이를 왔다 갔다 하는 경우의 수는 원칙적으로 무한히 많다.

실제 실험에 대응하는 것은 작은 규모의 시뮬레이션이다. 만약 실험 결과와 작은 규모 시뮬레이션 결과가 잘 들어맞는다면, 큰 규모의 시뮬레이션 결과도 충분히 신뢰할 만 하다고 말할 수 있다.

PCR 이후 전체 경로-시퀀스들에 대한 해답 경로-시퀀스들의 비율은 약 95.5%에 달하며, 이는 전기영동이 제대로 작동한다면 gel 상에서 해들을 확인하는 데 특별한 문제가 없음을 뜻한다. (그림 20)과 (그림 21)은 각각 작은 규모와 큰 규모의 시뮬레이션 결과를 보여 준다. 역시 시뮬레이션 1에서와 마찬가지로, PCR 이후 매우 다양한 길이의 경로들이 생성되었음을 알 수 있다.



(그림 20) 작은 규모 시뮬레이션에서 PCR 후 해답 경로들의 분포



(그림 21) 큰 규모 시뮬레이션에서 PCR 후 해답 경로들의 분포

4.4. 실험

4.4.1. 실험 설계 및 절차

기본적인 실험 과정은 제 4.3.2절의 시뮬레이션 절차와 동일하다. 이 실험에서는 <표 2>의 0번, 1번, 3번, 4번, 5번, 7번 상태-시퀀스들과 이에 따른 8 가지 연산-시퀀스들이 사용되었다. 실험 절차를 보다 상세히 기술하면 아래와 같다.

(1) 올리고 정제

각 올리고는 5′ 말단을 인산화한 후 PAGE를 이용하여 정제를 한다. 이는 올리고 1 nmol 당 10 unit의 T4 인산화 효소(Life Technologies)를 넣어 총 100 ul의 부피로 만든 후, 37 도에서 한 시간을 유지하여 만든다. 반응 완충용액의 조건은 70 mM tris-HCI, pH 7.6, 10 mM $MgCL_2$, 100mM KCl, 1mM 2-mercaptoethanol and 1mM ATP (Sigma, St. Lois, MO, USA)이다. 다음으로 인산화 효소의 활성을 없애기 위해 95 도에서 10 분 간 가열해 준다.

(2) 올리고의 Hybridization

각 올리고를 100 pmol 농도로 섞은 후, 초기 denature 위해 95 도에서 5 분 간 유지한다. 다음으로 hybridization을 위해 iCycler thermal cycler (Bio-rad, USA)를 이용해 온도를 분당 1 도씩 내려 16 도까지 낮춘다.

(3) Hybridization 거친 올리고의 ligation

Ligation 과정은 T4 DNA ligase를 처리한 후 16 도에서 12 시간

을 iCycler thermal cycler를 이용하여 유지한다. 반응 완충용액은 50 mM Tris-HCI (pH 7.8), 10 mM $MgCL_2$, 5mM DTT, 1mM ATP, and 2.5 g/ml BSA. 3.2.6을 포함하고 있다.

(4) 올리고 정제를 위한 전기영동

Ligation된 올리고는 2% Agarose gel에서 전기영동을 진행한다. 전기영동 완충용액은 100 mM Tris-HCl, pH 8.3, 89mM boric acid, and 2 mM EDTA을 포함하고 있다. 전기영동 동안에는 Bio-rad Model Power PAC 3000 electrophoresis unit을 이용하여 100 W (6V/cm)를 유지한다. 그 결과, gel 상에서 정확한 정답에 대응하는 사이즈의 밴드(band)들과 답이 아닌 사이즈의 밴드들을 모두 확인할 수 있었다. 이것들을 PCR을 거치게 하여, 정답에 대응하는 사이즈의 밴드들만을 다시 확인할 수 있다.

(5) 올리고 정제

올리고 정제는 QIAquick Gel Extraction Kit(QIAGEN, USA)를 이용하여 진행된다. 사용된 정제 kit에는 QG buffer, PE buffer, EB buffer, PB buffer가 포함되어 있다. 정제를 위해 전기영동을 진행한 gel에서 정답을 포함할 수 있는 모든 범위의 gel을 잘라 tube에 넣은 후 QG 완충용액을 3 배의 부피로 넣어준다. 다음 vortexing 후 50 도에서 10 분 간 유지하고, 다시 isopropanol을 처리한다. 이 tube를 13000 rpm에서 원심 분리한 후 PE 완충용액 0.75 ml를 첨가하여 씻어준다. 다음으로 다시 13000 rpm에서 원심 분리한다. 원심분리된 샘플을 elution을 위해 EB 완충용액(10mM Tric-Cl, pH8.5)이나 증류수를 넣어 보관한다.

(6) 정답의 PCR 증폭

PCR을 위해 primer와 template를 10 pmol 농도로 PCR premix(10mM Tris-HCL, pH 7.8, containing 50 mM KCl, 1.5 mM $MgCL_2$, 0.1% Triton X-100, 0.2 mM dNTP, and 1 U DNA Taq polymerase (Genenmed, Korea)와 섞어 총 20 ul 부피로 만든 후 PCR 반응시킨다. PCR 반응 조건은 <표 3>에 표시하였다.

단 계		시 간	실험 1 (온도)	실험 2 (온도)	사이클 수
Denature		5 min	96℃	96℃	1
PCR	Denature	30 sec	96℃	96℃	30
	Anealing	40 sec	58℃	62℃	
	Extension	30 sec	72℃	72℃	
Extension		3 min	74℃	72℃	1

<표 3> PCR 반응 조건

(7) 결과 확인을 위한 전기영동

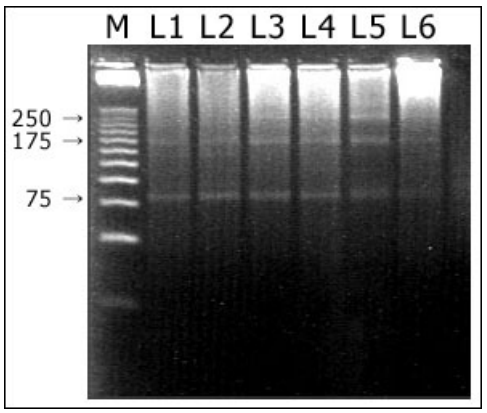
증폭된 PCR 결과물을 다시 2% gel 상에서 전기영동하여 확인한다.

4.4.2. 실험 결과

실험 결과는 (그림 22)와 같다. 여기서 M은 사이즈 마커(size marker)를, L1부터 L6까지는 3 개의 테스트 튜브와 2 종류의 온도 조건에 따른 서로 다른 PCR 결과물들을 나타낸다. L1 / L2, L3 / L4, L5 / L6 각각은 동일한 튜브 내 올리고들에 대해 전기영동을 건 결과인데, 이 때 L1, L3, L5는 PCR 시 Anealing의 온도 조건을 58℃로 한 것이고, L2, L4, L6는 62℃로 한 것이다(<표 3> 참조).

그 결과, 전기영동 상 4 개의 밴드가 (그림 22)에서와 같이 확실하게 확인되었고, 이것은 특히 L5에서 가장 뚜렷하게 보인다.

관찰된 4 개의 밴드는 미리 예측한 해답과 모두 일치한다. 예컨대, 원숭이가 바나나를 얻게 되는 가장 간단한 경로, 즉 최적해는 방문 옆에 있던 원숭이가 상자로 가서, 그 상자 위로 바로 올라가는 것인데 이것은 3 개의 상태 (atdoor, onfloor, middle), (middle, onfloor, middle), (middle, onbox, middle)로 구성되어 있다. 본 실험에서 하나의 상태는 30bp로 코딩된 시퀀스로 표현되고, 초기 상태와 목표 상태를 나타내는 PCR primer로는 각각 25bp의 시퀀스가 사용되므로 가장 짧은 해답-시퀀스는 80bp인데, 이것은 (그림 22)의 가장 아래 밴드에 정확히 상응한다. 실험 전 예상되는 밴드의 길이는 80bp, 140bp, 200bp, 260bp, 320bp 등인데, 이 중 (그림 22)에서는 140bp 길이의 시퀀스를 제외한 나머지 4가지 종류가 검출되었다. 그러나 아주 희미하게나마 140bp 밴드도 존재하는 것을 알 수 있다 (특히 L1, L2).



(그림 22) PCR 결과에 대한
electrophoretogram

4.5. 실험 결과 토론: BFS와 확률적 추론 과정

전통적인 접근 방식, 예컨대 Prolog를 써서 구현한 MBP 풀이는 기본적으로 깊이 우선 탐색(depth-first search, DFS)이며, 만약 최단 경로가 존재할 경우 이것의 발견을 보장해 준다. 그런데 DNA 컴퓨팅을 기법으로 MBP를 해결할 경우, 다양한 해를 찾을 수 있다는 것은 장점이나 과연 이 경우에도 최적해의 검출이 보장될 수 있는 것일까?

(그림 6)에서 설명한 바와 같이, DNA 컴퓨팅은 넓이 우선 탐색(breadth-first, BFS)을 기반으로 하기 때문에, 에러를 고려하지 않는 이상적인 실험 하에서는 또한 원칙적으로 최적해가 반드시 발견되어야 한다. 왜냐하면 BFS에서는 가능한 모든 경로들이 병렬적으로 탐색되기 때문이다. 이는 제 4.2절과 제 4.3절에서의 두 종류의 시뮬레이션 결과를 통해서도 알 수 있다. 모든 경우에 최적해가 발견되었다.

그러나 DNA 컴퓨팅은 대규모 병렬 처리 외에 일종의 확률적인 생화학 반응이라는 중요한 특성을 갖는다. 실험 과정 중 매 순간 어떤 종류의 에러가 발생할 가능성을 배제할 수 없으며, 따라서 운이 나쁘면 원하는 반응이 발생하지 않음으로 인해 최적해를 찾지 못하게 될 수도 있다. 따라서 이는 헤밀토니안 경로 문제(HPP)나 순환 외판원 문제(Traveling Salesman Problem, TSP) 등 지금까지 DNA 컴퓨팅 기법이 적용되었던 모든 연구에도 공통적으로 제기될 수 있는 사항이다 [Adleman, 1994][Shin et al., 1999]. 물론 용액 1 mole에는 대략 아보가르도 수(6×10^{23})에 해당하는 개수만큼의 분자들이 존재하며, 이들 사이에 일어나는 상호 작용은 상상을 초월할 정

도로 대규모로(massively) 또 불필요하게 중복해서(redundantly) 발생하므로, 만약 반응 시간이 충분히 주어진다면, 최적해가 생성되지 않을 확률은 거의 무시해도 좋을 정도로 낮게 될 것이다. 게다가 최적해는 대개의 경우 가장 간단한 시퀀스로 표현되기 때문에, 다른 해들에 비해 검출이 더 용이하다고 할 수 있다. 그러나 아무튼 DNA 컴퓨팅 기법을 사용하여 모든 경우에 항상 최적해를 찾을 수 있다고 100% 장담할 수 없다는 것만은 분명하다.

그렇다면 과연 확률적 반응이라는 DNA 컴퓨팅의 특성이 반드시 약점이기만 할까? 규칙에 기반한 고전적인 인공지능 시스템은 결점이 없는 논리적 및 수학적 추리력을 갖고 있다. 그러나 이와 대조적으로 인간은 항상 논리적으로 사고하는 것은 아니며, 때때로 심각한 오류를 보이기도 한다 [Cheng & Holyoak, 1985][Johnson-Laird & Steedman, 1978][Cosmides, 1989]. 그렇지만 결과적으로 인간의 추론 혹은 결정 내리기는 현실 문제들의 맥락에서 오히려 강력한 힘을 발휘하며, 이는 규칙에 고지식하게 얽매이지 않는 대신 상당한 정도의 자유도를 유지한 채 주어진 상황을 고려하여 판단할 수 있는 데에 기인한다고 보인다. 인공지능의 본연의 목적은 인간 지능에 대한 모방, 즉 사람이 잘 하는 문제는 똑같이 잘 하고 잘 못하는 문제에 대해서는 역시 어려움을 겪는 인공적인 시스템을 구축하는 데에 있으므로, 이런 관점에서 DNA 컴퓨팅의 이러한 특성은 오히려 인공지능의 입장에서는 장점이 될 수도 있다. 물론 사람의 사고의 기저에 있는 메커니즘이 DNA 컴퓨팅의 그것과 동일 혹은 유사하다고 이야기하는 것은 너무나 과도한 주장일 것이다. 그러나 최소한 기제가 아닌 결과의 측면에서는, 이전 접근 방식에서는 찾아볼 수 없던, 인간의 사고 및 행동 양식과의 공통점을 어느 정도 보여준다.

5. 결론

지금까지 원숭이가 바나나를 얻게 되는 다양한 경로들을 발견할 수 있음을 보였다. 이것은 단순히 초기 상태로부터 목표 상태로 전이할 수 있는가에 대한 답(Yes-No)만을 주는 것이 아니라, 목표 상태에 이르는 다양한 해법들을 제공해 준다는 점에서 제 3.2절에서 살펴 본 Prolog를 사용한 방식과 다르다. 우리에게 보다 의미 있는 작업은 사람이 고안할 수 있는 다양한 해법들을 인공적인 추론 기관이 또한 생성해 낼 수 있다는 것을 보이는데 있다고 생각한다.

이와 관련하여, 인공지능 구현을 위한 바람직한 방법론이란 관점에서 원숭이와 바나나 문제에 대한 시뮬레이션 결과는 DNA 컴퓨팅에 또 다른 의미를 부여한다. 이 점에 대해, 앞서 제 3.3절에서 설명한 바를 한 번 더 도식적으로 정리하면 다음과 같다: 고전적 인공지능의 해결 방식에는 근본적으로 어려움이 있는데, 그것은 제 3.1절의 방법 (1)을 적용할 경우 계산학적인 난점에 봉착하게 되고, 방법 (2)를 적용할 경우 인공지능을 추구하는 본래의 목적을 충족시키지 못하게 되기 때문이다. 반면, DNA 컴퓨팅을 이용한 해결에 관해서는 현 단계에서 현실적으로 어려움이 있다. 본 논문에서처럼 방법 (1)을 사용하는 것은 개별 문제마다의 기호 구체화(symbol grounding)라는 점에서 실험자에게 시퀀스 디자인의 부담과 이에 따른 경제적 비용을 지우며, 방법 (2)를 사용하는 것은 현재의 기술 수준으로는 사실상 불가능하기 때문이다. 이렇듯 두 가지 해결 방식이 모두 문제점을 갖고 있으나, 논리적이나 물리적인 가능성이 아닌, 현실적인 가능성은 얼마든지 변화될 수 있기 때문에, 향후 DNA 컴퓨팅 기술이 발전하게 되면, 계산하는 데 소요되는 비용은 점점

떨어지게 될 것이고, 또 DNA 시퀀스들을 지금보다 훨씬 더 자유롭게 조작할 수 있게 되면서 기호 구체화 문제도 또한 해결되게 될 것이다. 바꾸어 말하면, DNA 컴퓨팅은 매우 새로운 분야인 만큼, 다름 아닌 바로 그 이유로 인해 연구할 만한 충분한 가치와 당위성을 갖는다. 이것은 본 논문의 연구 방법론의 필요성을 지지해 주는 강력한 논거가 된다고 말할 수 있다.

요컨대 본 연구의 주된 성과는 지금까지 살펴본 원숭이와 바나나 문제와 같이 그래프 탐색의 그것으로 환원될 수 있는 문제의 경우, 병렬 처리가 이루어지는 프로세서들 사이의 대규모 상호 작용에 의한 넓이 우선 탐색은 더 이상 절차적 관점을 요구하지 않는다는 점과 이로 인해 백트래킹 기법의 도입이나 무한 루프에 빠지는 현상에서 벗어날 수 있다는 점을 보임으로써, 문제 해결을 위한 논리적 추론에 있어 생물학 기반의 인공지능이 기여할 수 있는 새로운 측면을 밝힌 데에 있다 하겠다. 이러한 결과는 DNA 컴퓨팅 기법이 발산적 사고 혹은 창발성(emergency)을 설명하고 시뮬레이션하기 위한 자연스러운 도구가 될 수 있다는 점을 시사한다. 또 설(J. Searl) 등 생물학적 자연주의를 주장하는 진영에 대해 인공지능 구현을 위한 생물학 칩(bio-chip)의 제작 가능성이라는 희망을 준다. 고전주의 대 연결주의라는 친숙한 대립 구도의 관점에서 볼 때, DNA 컴퓨팅 시스템 자체는 어떤 고정된 구조(architecture)를 갖고 있지 않은, 극단적인 연결주의 시스템의 일종으로 간주될 수 있다. 이 시스템에서 발생하는 놀랍도록 복잡한 확률적인 생화학 반응을 원하는 대로 통제하게 된다면, 지금까지의 규칙 기반의 전통적 방식으로는 도저히 접근할 수 없는 어떤 지능적 특성들을 우리는 언젠가 인공적으로, 그러나 매우 자연스럽게 구현할 수 있을지 모른다.

참 고 문 헌

박의준, 이인희, 장병탁 (2002),「DNA 컴퓨팅을 이용한 삼단논증의 결론 증명」, 『한국정보과학회 가을 학술발표 논문집 (II)』, 제 29권, 제 2호, 한국정보과학회, 382-384쪽.

박의준, 이인희, 장병탁 (2003),「DNA 컴퓨팅을 이용한 원숭이와 바나나 문제 해결」, 『인지과학』, 제 14권, 제 2호, 한국인지과학회. (게재 예정)

이광형, 이병래 (1996),『인공지능』, 한국방송대학교 출판부, 서울.

장병탁 (2002),「분자정보처리기술」,『전자공학회지』, 제 29권, 제 3호, 한국전자공학회, 286-298쪽.

Adleman L. M. (1994), "Molecular Computation of Solutions to Combinatorial Problems", *Science*, vol. 266, pp. 1021-1024.

Anderson J. R. (1995), *Cognitive Psychology and Its Implication*, W. H. Freeman (국문 번역본: 이영애 옮김,『인지심리학과 그 응용』, 이화여자대학교 출판부, 서울, 2000)

Bratko, I. (1990), *Prolog Programming for Artificial Intelligence*, 2nd Ed., Addison-Wesley Pub. Co., Wokingham.

Cheng, P. W. and Holyoak, K. J. (1985), "Pragmatic Reasoning Schemas", *Cognitive Psychology*, vol. 17., pp. 391-416.

Cosmides, L. (1989) "The Logic of Social Exchange: Has Natural Selection Shaped How Human Reason? Studies with the Wason Selection Task", *Cognition*, vol. 31, pp. 187-276.

Johnson-Laird, P. N. and Steedman, M. (1978), "The Psychology

of Syllogisms", *Cognitive Psychology*, vol. 10., pp. 64-99.

Kobayashi, S. (1999), "Horn Clause Computation with DNA Molecules", *Journal of Combinatorial Optimization*, vol. 3, pp. 277-299, 1999.

Köhler, W. (1956), *The Mentality of Apes*, Routledge & Kegan Paul, London.

Lee, I.-H., Park, J.-Y., Jang, H.-M., Chai, Y.-G., and Zhang, B.-T. (2003), "DNA Implementation of Theorem Proving with Resolution Refutation in Propositional Logic", *Lecture Notes in Computer Science*, vol. 2568, pp. 156-167.

Lim, H.-W., Yun, J.-E., Jang, H.-M., Chai, Y.-G., Yoo, S.-I., and Zhang, B.-T. (2003), "Version Space Learning with DNA Molecules", *Lecture Notes in Computer Science*, vol. 2568, pp. 143-155.

Mihalache, V. (1997), "Prolog Approach to DNA Computing", *Proceedings of the International Conference on Evolutionary Computation*, pp. 249-254.

Newell, A. (1990), *Unified Theories of Cognition*, Harvard University Press (국문 번역본: 차경호 옮김, 『통합인지이론』, 아카넷, 서울, 2002)

Newell, A. and Simon, H. (1963), "GPS, a Program that Simulate Human Thought", in Feigenbaum, E. and Feldman, J. (eds.), *Computer and Thought*, McGraw-Hill, New-York, 1995, pp. 279-293.

Setubal, J. C. and Meidanis. J. (1997), *Introduction to Computational Molecular Biology*, PWS Pub. Co., Boston.

Shin, S.-Y., Zhang, B.-T., and Jun, S.-S. (1999), "Solving Travelling Salesman Problems Using Molecular Programming", *Proceedings of the 1999 Congress on Evolutionary Computation*, vol. 2, pp. 994-1000.

Uejima, H., Hagiya, M. and Kobayashi, S. (2001), "Horn Clause Computation by Self-Assembly of DNA Molecules", *Preliminary Proceedings of the Seventh International Meeting on DNA Based Computers*, pp. 63-71.

Wasiewicz, P., Janczak, T., Mulawka, J. J., and Plucienniczak, A. (2000), "The Inference Based on Molecular Computing", *International Journal of Cybernetics and Systems*, vol. 31, no. 3, pp. 283-315.

Wason, P. C. and Johnson-Laird, P. N. (1972), *Psychology of Reasoning: Structure and Content*, MIT Press, Cambridge.

Williams, R., *The Monkey and the Banana*, Tutorial Materials, School of Computing, University of Tasmania, Australia, http://www.comp.utas.edu.au/units/kxa252/tutorials/MonkeyBan.pdf (2002. 12. 1 현재).

Zhang, B.-T. and Shin, S.-Y. (1998), "Molecular Algorithms for Efficient and Reliable DNA Computing", *Proceedings of the Third Annual Genetic Programming Conference*, pp. 735-742, Morgan Kaufmann.

Abstract

The Monkey and Banana Problem is an example commonly used for illustrating simple problem solving. It can be solved by conventional approaches, but this requires a procedural aspect when inferences are processed, and this fact works as a limitation condition in solving complex problems. However, if we use DNA computing methods which are naturally able to realize massive parallel processing, the Monkey and Banana Problem can be solved effectively based on the breadth-first search. In this paper, we design a method of representing the problem using DNA molecules, and show that solutions are generated through bio-lab experiments. The results are contrary to those of conventional approaches in that at least 4 solutions containing the optimal are founded out. That is, DNA computing overcomes the limitations of conventional approaches.

Keywords : Monkey and Banana Problem, Logical Inference, DNA Computing

Student Number : 2001-23113

감사의 글

고마우신 분들이 많습니다. 그러나 이 지면을 통해서는 본 논문과 직접적으로 관련된 분들께만 감사의 말씀을 드리고자 합니다.

먼저 바이오지능 연구실의 이인희, 남진우 학형, 정말 감사합니다. 바쁜 연구 활동 중에도 두 분께서 귀한 시간 쪼개어 각각 시뮬레이션과 실험 부분을 도와 주셨기에 결국 이 논문이 완성될 수 있었습니다.

DNA 컴퓨팅이라는 생소한 분야의 논문 심사를 기꺼이 맡아 주신 철학과 김영정, 언어학과 신효필 교수님께 깊이 감사 드립니다. 특히 김영정 교수님께서는 제게 논리학과 C 프로그래밍 언어를 가르쳐 주셨고, 저를 인지과학 협동과정으로 이끌어 주셨으며, 마침내 제 졸업까지 가능하게 해 주셨습니다. 두 교수님의 조언과 충고, 그리고 비판 덕분에 또한 제 논문이 이렇게 나올 수 있게 되었습니다.

마지막으로, 지도교수이신 컴퓨터 공학부 장병탁 교수님께 진심으로 감사 드립니다. 설득력 있도록 논문을 구성하는 방법으로부터 학문하는 사람이 마땅히 지녀야 할 자세에 이르기까지, 교수님께 지도를 받은 일년 남짓한 짧은 기간 동안 너무나 많은 것을 배웠습니다. 연구실에 있는 동안 스스로 알아서 보다 적극적으로 연구에 임하지 못한 점이 그저 죄송스러울 따름입니다.

이 논문을 사랑하는 부모님께, 그리고 아내 윤정에게 바칩니다.