공학석사학위논문

DNAgram: Anagram 문제 해결에 관한 분자 컴퓨팅 시뮬레이션 연구

DNAgram: Molecular Simulation of Anagram Problem Solving

2003년 2월

서울대학교 대학원
협동과정 인지과학전공
이 은 석

DNAgram: Anagram 문제 해결에 관한 분자 컴퓨팅 시뮬레이션 연구

DNAgram: Molecular Simulation of
Anagram Problem Solving

지도교수 장병탁

이 논문을 공학석사 학위논문으로 제출함

2002年 10月 日

서울대학교 대학원
협동과정 인지과학전공
이 은 석

이은석의 공학석사 학위논문을 인준함

2002年 12月 日

심사위원장 김정오 印

심 사 위 원 장병탁 印

심 사 위 원 김기현 印

초 록

본 연구는 주어진 영어 철자들만으로 실제 영단어를 형성하는 퍼즐인 anagram 문제 해결 과정을 분자 컴퓨팅으로 시뮬레이션하여 anagram의 해결에 관여하는 인간 지각의 한 측면을 고찰하고자 한다.

본 연구는 두 가지의 가설을 채택했다. 먼저, 지능이 내성으로는 접근 불가능한 층위에서 대규모의 병렬 처리들의 상호작용에 의해 창발되어 상위 인지로 이어질 것이라는 D. Hofstadter 등의 제안이 anagram 문제 해결에 어떻게 적용되는지 검토하였다. 이어 구체적으로 anagram의 해결 중 많이 언급되는 pop-out solution은 의미론적 처리기의 관여 없이, 입력된 항목들을 단순히 재배열하여 작용하고 제약의 집합들을 확률적으로 통계처리해서 가능하다는 가설을 검토하였다. 이를 검토하기 위해 시뮬레이션을 시행했다. 시뮬레이션의 방법론으로 L. Adelman이 처음 사용한 분자 컴퓨팅을 채택하여 가설들이 함축하는 병렬성을 효율적으로 실현하고자 했다. 본 실험의 목표는 인간의 anagram 문제 해결 방법 중 일부분을 정확하게 흉내내는 것이다. 따라서 심리학적으로 최대한 그럴듯한 양상을 도출하기 위해 문제 영역을 최소화했다. 컴퓨터 모의 실험은 anagram 문제를 해결하는 과정에서 사전 검색 방법을 사용하지 않음으로써 searching solution이 개입될 여지를 제거했고, 철자들 간의 결합강도 규칙만을 시뮬레이션 프로그램의 지식으로 투입해서 의미론적 문제 해결이 일어날 수 있는 가능성도 제거하였다. 이와 같은 조건에서 문제해결이 가능한지를 확인하였다. 이 실험에서 정답 단어가 도출됨을 관찰하였다. 이후의 실험에서는 규칙의 활성화 정도를 투입되는 시퀀스의 양으로 간주하고, 이 양을 변화시켜 그에 따른 프로그램의 문제 해결 결과를 관찰하였다. 정답 출력의 양은 결합강도 규칙의 양이 커질수록 늘어났으나 임계치에 도달하면 더 이상 늘어나지 않았다. 또한 규칙의 강도가 강할수록 오답이라도 정답에 더 가까운 답의

양이 늘어났고, 그렇지 않은 경우의 단어들의 수는 상대적으로 줄어들었다.

시뮬레이션 실험으로 다음과 같은 결론을 내릴 수 있었다. 먼저 소규모의 하위 규칙(rule; constraint)만으로도 입력된 철자 항목들을 조작하여 지각 프로세스가 일어날 수 있음을 확인했다. 따라서 다양한 규칙들을 실험에 적용할 수 있는 가능성을 보였다. 또한 실험 결과로부터 유연한 개념 형성에 효율적인 분자 컴퓨팅의 방법론적 특성을 확인하였다. 이는 인지 기능의 시뮬레이션에 있어서 기호주의, 연결주의 인공지능과의 그 차이를 반영하는 것으로 생각된다.

주요어: Anagram, 분자 컴퓨팅, 병렬 처리, 철자 제약 규칙

목 차

1. 서론

1. 1 연구의 목표

본 연구는 wordplay의 일종인 anagram을 분자 컴퓨팅으로 해결하는 프로그램을 통해 인간의 지각(perception)과 창의성(creativity)의 핵심 측면을 고찰하는 것이다. Anagram은 아무 순서 없이 흩어져 있는 철자들의 집합을 입력으로 받으면, 그것들을 재배열하고 좀더 그럴듯한 순서로 배치해서, 종국에는 그럴 듯한 영어 단어를 만드는 퍼즐이다. 예를 들어 'tderhun'이라는 anagram이 주어지면 'thunder'라는 단어로 재배열해서 이 문제를 해결한다.

Anagram의 해결은 작은 문제이지만, 지능의 중요한 단면을 예증한다. Anagram의 해결에 사용되는 방식들이 다른 영역에서도 그대로 사용되기 때문이다. 즉 사람은 늘 anagram에 사용되는 방식에서처럼, 수많은 작은 조각들을 섞어보고juggling, 결합combining하여 더 큰 조각들로 만드는 작업을 한다. 그렇게 해서 새롭고, 유의미하고, 강한 무언가를 만들어내는 것이다. 간단하게는 multi-level cleaving, splicing, regrouping, reordering, rearranging의 조작(operation) 등을 그 예로 들 수 있다. 이 조작들은 음악, 문학, 예술, 새로운 과학적 발견, 발명 등의 창조 행위(creation process)에 늘 관여하는 핵심적인 요소이다.

이러한 조작들은 모든 인간의 보편적인 인지 기능인 동시에, 의식적으로는 잘 인식하지 못하는 하위적 기능이라는 점에서 주목을 요한다. 본 논문은 다음과 같은 가정을 취했다. 즉, 지능은 인지 행위의 기저에 위치한 하위 지각 작업, 방대하게 상호작용하는 병렬 작업(parallel processes)을 통해 창발될 수 있다는 것, 또 이 작업들은 수 밀리초 사이에 일어나고 내성(introspection)으로 그 내용에 대한 접속이 불가능하다는 것이다.[1] 또 매우 추상적이고 때로는 말로 설명하기 힘든 카테고리들로 분류

1) [Hofstadter, D. R. (1995)]

될 수 있는, 지각의 한 층위, 어떤 특정한 모듈에도 속하지 않는 지각의 층위가 있을 수 있다는 가정이 그것이다.

본 논문은 전술한 바와 같은 anagram을 해결하는 프로그램인 DNAgram을 제안하고, 그 실행 결과를 통해 언어 지각 처리에 따르는 인지 기능을 고찰하였다. DNAgram은 단순한 작업을 구현하지만 복잡한 dynamic memory structure를 취하며, emergent computation의 원칙에 기반한 제어 구조를 가지고 있다. 이 구조 안에서, 아주 많은 소규모의 계산 행위들의 통계적 결과가 보다 상위 차원의 행동으로 귀결된다. 이 모델의 완전한 구현은 인간의 창의적 문제 해결의 메커니즘에 대한 고찰이 될 것이다.

1.2 연구의 범위 및 내용

Anagram 해결에 간해 인지심리학적 연구와 컴퓨터 프로그램은 여러 연구 결과와 성과가 있었다. 본 논문에서는 언어의 하위 지각 처리라는 측면을 고찰하기로 한다. 이 하위 지각 처리 과정이 중요하고 기본적이라고 생각한다.

앞서 언급한 바와 같이 본 논문은 인간 언어 지각의 문제를 단어 인식, 단어 중심의 측면에서 살펴보는 것이므로, 한 가지 방법이 아닌 학제간 연구 방법론을 적용해야 한다. 먼저 이론적으로는 전통적인 기호주의와 연결주의 인공지능의 성과를 통해 인지 기능의 구현에 관한 철학적 논의, Anagram 해결에 관한 심리학적 선행 연구들을 중심으로 하위 지각의 중요성, 문제점을 고찰하였다. 실험은 분자 컴퓨팅의 방법을 사용하였다. 일련의 모의 실험을 이용하여, anagram 해결에 따르는 양상을 분석하고 이를 심리학적으로 해석하였다. 이러한 두 접근 방식으로 하위 지각의 측면과 이에 대한 모의 실험 결과가 어떻게 나타나는지 검토해볼 수 있다.

1.3 논문의 구성

본 논문은 모두 여섯 장으로 구성되어 있다. 서론인 1장에서는 연

구의 목표, 범위, 그리고 내용을 약술하였다. 본론은 크게 세 부분으로 나뉘어지는데, 먼저 2장에서는 본 논문의 주제와 연관되는 선행 연구들을 크게는 철학, 심리학 분야로 구분하고, 이를 인지 기능의 구현에 관한 인공지능, 지각, 하위 지각에 관한 이론적 연구, anagram에 관한 선행 연구들을 정리하였다. 끝으로 실험적 방법으로 사용한 분자 컴퓨팅에 대해 간략하게 소개하였다. 3장에서는 인지과학에서 anagram이 왜 중요한 소재가 되는지, 또 그에 따른 실험적 방법론으로 왜 분자 컴퓨팅을 사용하는지에 대해 설명하였다. 이어 4장에서는 본 논문이 제안하는 시뮬레이션 프로그램인 DNAgram의 구조에 대해 설명하고, 5장에서 이 프로그램의 구체적인 구현 절차와 그 결과를 통해 anagram 해결 과정에서 나타난 인지 기능을 어떻게 해석할 수 있는지 논의하였다. 마지막 6장은 4, 5장의 내용을 통해 도출된 결론에 대해 논의하고 앞으로의 연구 과제를 논의하는 것으로 맺었다.

2. 연구 배경

2.1 객관주의와 고전주의 인공지능

인공지능이 표상 형성의 문제를 심각하게 고려하면서, 그 다음의 문제는 분명히 존재하는 인간의 상위 지각 프로세스의 유연성을 어떻게 다룰 것인지가 대두되었다. 맥락과 하향적 영향으로, 사물과 상황은 인간에게 여러 다양한 방식으로 이해된다. 인공지능의 표상 또한 그에 상응하는 유연성을 확보할 수 있을 것인가. 윌리엄 제임스(William James)는 인지 표상의 이러한 측면을 인식했다.[2)]

> There is no property ABSOLUTELY essential to one thing. The same property which figures as the essence of a thing on one occasion becomes a very inessential feature upon another. Now that I am writing, it is essential that I conceive my paper as a surface for inscription... But if I wished to light a fire, and no other materials were by, the essential way of conceiving the paper would be as a combustible material... The essence of a thing is that one of its properties which is so important for my interests that in comparison with it I may neglect the rest... The properties which are important vary from man to man and from hour to hour... many objects of daily use-as paper, ink, butter, overcoat-have properties of such constant unwavering importance, and have such stereotyped names, that we end by believing that to conceive them in those ways of conceiving them than any others; there are only more frequently serviceable ways to us.

---

2) [James, W. (1990)] pp. 222-224.

제임스는 우리가 사물이나 상황을 다른 시간에 다른 표상으로 가진다는 것을 지적하고 있다. 표상적 절차는 주어진 맥락의 압력에 맞추어 적응한다.

그러나 초기 인공지능 연구는 사물, 상황, 카테고리의 표상의 지각에 대해 전적으로 객관주의를 취했다. 조지 라코프(George Lakoff)가 지적했듯, "객관주의에 의하면, 현실은 단 하나의 독특하고 정확한, 완벽한 실재로서의 구조로 나타난다. 이 구조는 인간 이해와는 전혀 별개로 존재한다.On the objectivist view, reality comes complete with a unique correct, complete structure in terms of entities, properties and relations. This structure exists, independent of any human understanding."[3] 이 객관주의는 철학에서는 활발한 논의가 잦아지는 시점이었음에도, 초기 인공지능의 대부분은 이 가정을 수용했다(이미 비트겐슈타인Wittgenstein은 언어와 현실 간의 엄격한 대응이 부적절함을 보였다).

대부분의 전통적 인공지능 연구가 이루어졌던 바탕이 되는 The Physical Symbol System Hypothesis은, 사고(thinking)는 기호적 표상의 조작을 통해 일어나며, 이 표상들은 atomic symbolic primitive들로 구성되어 있다는 주장을 확실히 했다.[4] 이러한 기호적 표상은 그 경계가 엄격한 all-or-none의 성격을 띠었으며, 맥락에 따른 변화에 반응하는 데 있어 섬세하게 이동하는 표상의 성격을 담는 것은 어려웠다. 사실상 결과는 객관주의의 주장에 따른 사실(reality)의 모습처럼 고정되고 엄격한 사실의 구축이었다.

1970년대 중반, 소수의 인공지능 연구자들이 이러한 엄격한 표상적 framework와는 궤를 달리 하는 시도를 시작했다. 데이비드 마(David Marr)는 다음과 같이 언급했다.[5]

---

3) [Lakoff, G. (1987)] p. 159.
4) [Newell, A. & Herbert A. Simon, (1976)] pp. 81-110.
5) [Marr, D. (1982)] p. 358.

1. The chunks of reasoning, language, memory, and perception ought to be larger than most recent theories in psychology have allowed. They must also be very flexible, and incorporating this requirement precisely will not be easy.

2. The perception of an event or object must include the simultaneous computation of several different descriptions of it, that capture diverse aspects of the use, purpose or circumstances of the event or object.

연결주의는 이러한 점에서 인공지능의 중요한 발전을 이끌어냈다고 볼 수 있다. 분산된 표상이 높은 맥락의존context-dependent 양상을 보이는 연결주의 모델은 표상의 유연성을 획득하는 데 중요한 기여를 했다.[6] 이 모델은 내부 프로세싱에서 표상적인 primitive가 없다. 대신 각 표상은 다차원 공간에서의 벡터이고, 공간의 위치는 외부 자극의 변화에 유연하게 적응된다. 결과적으로, 한 카테고리의 구성원이 다 같은 기호적 구조에 의해 표상되지 않는다. 오히려 개개의 사물들은 그것들이 제시되는 맥락에 의존해 약간씩 다른 방식으로 표상된다. recurrent connection을 가진 연결망에서는, 표상은 모델의 내부 현재 상태에 더욱 민감하다. 표상에 대해 유연한 접근을 취하는 다른 연구 중에서는 홀랜드Holland의 것을 들 수 있다. 이 모델은 유전학적 방법론을 서서 다양한 상황의 여러 측면들에 반응할 수 있는 'classifier' 집합을 만들어낸다.

이러한 모델들에서는, 유연한 지각 과정이 표상적 내용을 바탕으로 한 행동에 통합된다. 그렇게 함으로써 전통적인 방법으로는 힘들었던 다양한 상황에 대한 강한 반응을 취하게 만들었다. 그럼에도 불구하고, 이

6) 자세한 내용은 [Rumelhart, D. E., James L. McClelland, & the PDP Research Group (1986)] 참조.

모델들은 여전히 어느 정도 시작 단계에 불과하며, 생산해내는 표상들 또한 전통주의 모델에서 보인 위계적으로 구성된 표상들처럼 복합적이지도 않다.

현재의 인공지능 연구는 많은 하위 층위의 지각 작업들과 상위 층위의 인지 과정을 분리해서 모델링하는 것이 가능하다는 가정을 대부분 가지고 있다. 이 가정 하에서, 표상들이 대부분의 경우 개발자의 손에 의해 만들어지고, 이는 각각 하위 층위의 편의에 따라 조작된다.

인지 행위로부터 표상 형성 과정을 분리하는 것은 불가능하다고 본다. 인지 행위에 명백히 드러나는 일종의 유연성을 제공하기 위해서는 표상 형성 과정과 그 표상들의 조작 사이의 끊임없는 상호작용이 필요할 것이다.

2.2. 지각에 관한 문제

인지과학에서 가장 중요한 문제들 중 하나는 사람이 주변환경으로부터 끊임없이 들어오는 방대한 양의 raw data를 어떻게 이해하는가에 대한 것이다. 인간 지각의 본질은 엄청난 data의 chaos로부터 규칙을 추출해내는 마음의 능력에 있다. 이런 능력으로 인해 인간은 시각장에 잡힌 움직임을 감지하기도 하고, 목소리에 실린 감정을 느끼고, 게임에서의 위협을 지각하며, 현재의 정치적 사건을 과거의 정치적 사건에 비유하기도 한다.

지각이 여러 층위에서 일어난다는 것을 인식한 것은 아주 오래 전부터이다. 임마누엘 칸트Immanuel Kant는 마음의 지각적 작업(percpetual work)을 두 부분으로 나누었다. 감각의 기관(the faculty of sensibility)은 가공되지 않은 감각 정보(raw sensory information)을 집어내고, 이해 기관(the faculty of understanding)은 이러한 정보들을 정합적이고 유의미한 세계에 대한 경험으로 조직화한다. 칸트는 감각 기관이 그리 흥미롭지는 않다고 보았고, 이해 기관에 훨씬 큰 노력을 들였다. 그는 상위 차원의 지각적 프로세스를 수반하는 세부적인 모델을 제안하고, 이해를 12가지 카테고리로 분류했다.[7)]

오늘날 칸트의 모델이 기이할지는 몰라도, 그의 근본적인 직관은 아직도 유효하다. 지각 프로세스는 스펙트럼을 가진다고 볼 수 있는데, 편의상 우리는 이 스펙트럼을 두 부분으로 나눌 수 있다. 칸트의 감각 기관에 대충 상응하는 하위 지각(low-level perception)은 다양한 감각 형식들로부터 들어온 정보들을 초기 처리한다. 다른 쪽의 상위 지각(high-level perception)은 이 정보를 좀더 포괄적으로 처리하면서 개념(concept)을 들여와 raw material로부터 의미를 추출하고, 개념적 층위(conceptual level)에서 상황을 이해한다. 이 층위는 물체의 인식부터 추상적 관계 파악에 이르고 정합적인 전체로서의 상황 이해까지 이어진다.

하위 층위의 지각은 '이해'와는 거리가 있다. 그러나 상위 지각은 인지의 핵심 문제와 가장 밀접한 관계를 가지고 있다. 상위 지각은 정신적 표상 문제로 연결된다. 표상은 지각의 열매다. raw data가 정합적인 전체로 변하기 위해서는, filtering과 organization의 과정을 거쳐 조직화된 표상으로 처리되고, 마음에 의해 어떠한 목적으로 쓰일 수 있게 되어야 한다. 표상에 대한 가장 큰 의문은 그 표상의 구조에 관한 것이다. 또한 중요한 문제는, 어떻게 이 표상들이 지각 과정을 거쳐 형성되는가이다. 표상-형성의 프로세스는 많은 중요한 문제를 불러일으킨다. 어떻게 표상은 맥락에 의해 영향을 받는가? 우리의 상황 지각이 필요할 때마다 어떻게 강하게 재형성되는가? 지각 과정 중 어디에 개념이 끼어드는가? 어디에서 의미가 들어오고, 어디에서, 그리고 어떻게 이해가 출현하는가?

이 글은 상위 지각이 다른 인지 프로세스와 어떻게 관련을 맺는지, 인공지능 영역에서 인지 모델링에 어떻게 지각 과정을 통합시켜야 할 것인지를 조명할 것이다. 인공지능에서 많은 작업들이 지각 과정과는 별도로 개념 층위의 프로세스(conceptual process)를 모델링해왔다. 그러나 이러한 접근은 인간의 마음에 대한 만족할 만한 이해를 얻는 길이 되지 못할 것이다. 하위 지각 과정을 제거하고 표상만으로 조절되는 시스템은 본질적인 인지 행위를 구현하지 못하는 한계가 있다.

---

7) [Hofstadter, D. R. 1995] p. 169. 에서 재인용.

2.3 Subcognitive Processes

지각이 무의식의 깊이에서 갑자기 자발적으로 튀어오르는 경우가 있다. 자신이 지각에 관련된 작업을 하고 있다고 완전히 의식하더라도, 그 '구성성분(ingredient)'들은 아래에 위치한 무의식의 재료로부터 위의 의식적인 마음으로 올려 전해진다. 지각 생성과 이해를 담당하는 프로세스는 subcognitive하고, 지각 작업의 힘은 두뇌의 subcognitive한 성격에 깊이 의존한다는 것이 분명해 보인다.

상호작용하는 수많은 subcognitive process들이 상위 차원의 cognitive behavior로 떠오른다는 개념이 이 연구의 핵심이다. 지능의 컴퓨터 모델이 고도의 parallel subcognitive process들에 기반하는 것이 필요한지 아닌지에 대한 문제는 인공지능 연구에서 가장 중요한 철학적 질문 중 하나다. 이 질문은 인공지능 연구를 크게는 두 개의 진영으로 나누어왔다. 본 연구는 컴퓨터 모델의 행동이 수많은 무의식적인 병렬 프로세싱의 상호작용으로 출현하지 않는 이상, 인간 cognition의 유연성을 컴퓨터 모델은 결코 이루어낼 수 없을 것이라는 믿음에 바탕을 두고 있다.

1950년대 말에 이르러 점차 내성주의적인 프로토콜이 실험 심리학 진영에 다시 스며들기 시작했으며 컴퓨터를 통해 인지 모델링을 하려는 초기 시도들도 이루어지기 시작했다. 이런 프로토콜에 기댄 모델들 중 가장 눈에 띄는 것은 Newell, Shaw, and Simon의 General Problem Solver (GPS)이다.[8)]

비록 Newell, Shaw, and Simon은 인간 인지에서의 subcognitive process들의 존재를 인식했지만, 이러한 process들이 지능적 행동 모델에 포함되어야 하는지에 대해서는 부정적이었다. 그들의 모델은 deterministic, top-down approach를 취했다. 이 접근법은 60, 70년대의 AI 연구에 커다란 영향을 끼쳤다. 최근에는 연결주의 모델들에 의해 상위차원의 인지 행동 생성에 하위차원 프로세스가 중요하지 않다는 개념이 서서히 무너지고 있다. 연결주의는 인지 모델링이 연구되어야 할 적절한 차원은 Newell 등

8) [Newell, A., and Herbert Simon (1963)] pp. 279-293.

이 기술한 기호 차원symbolic level(본질적으로 세계의 사물들이 그들의 프로그램의 symbol에 대응된다)의 아래에 있고, 뉴런 차원, 혹은 그 위에 존재한다고 주장한다.[9]

DNAgram은 기호주의와 연결주의 사이에 위치한다. GPS 같은 rule-based, top-down 방식과 대부분의 연결주의 모델의 bottom-up 방식을 결합한다. 두 가지 방식은 계속해서 상호작용하고, 점진적으로 프로그램이 부여받은 문제에 대한 표상을 구축할 수 있게 된다.

2.4 Anagram 숙련자의 문제 해결 방식에 대한 선행연구

Novick은 anagram 문제가 검색search과 pop-out으로 해결된다고 생각하고, anagram에 숙련된 참여자와 초보적인 참여자들의 해결방식을 관찰하였다.[10]

검색 전략은 초기 단계부터 의도적인 노력을 통해 이루어진다. 예를 들어 'tderhun'을 해결할 때 먼저 생각나는 'hunter,' 'under,' 'tender' 등을 검토하고, 이 후보들이 철자를 다 쓰지 못하였거나, 다른 철자를 더 요구하게 됨을 평가한다. 이 전략은 작업기억 안에 즉각적인 결과가 적재되기 때문에, 해결자들은 부분 정보의 점진적인 저장을 보고한다. 그러나 때로는 어떤 의식적인 인지 없이도 마음에 갑자기 해답이 튀어올라(pop-out) 목표가 해결될 때도 있다.

Novick과 Cote의 연구에서, 숙련된 anagram 해결자들은 평균적으로 pop-out solution을 쓰는 경향이 덜 숙련된 해결자들보다 더 많았다. 광범위한 연구들이 전문가와 초보자가 주의를 집중하는 문제의 특징의 큰 차이를 보고하고 있다. 초보자들은 해결과는 무관한 피상적 특징들에 우선적으로 주의를 기울이는 반면, 전문가들은 해결과 유관한 구조적 특징에 우선적으로 주의를 기울인다. 이 때문에, 덜 숙련된 피험자들의 해결 시간에 발음 가능성pronounceability이 더 큰 영향을 미친 반면, 숙련된 피험자들

---

9) [Smolensky, P. (1988)]
10) [Novick, L. R., & Cote, N. (1992)] pp. 450-455.

의 해결 시간에는 철자 규칙 제약spelling constraint이 더 큰 영향을 끼쳤다.

Keil, Smith, Simons, and Levin에 따르면, 개념(concept)의 실시간 사용에서는, 추론의 유사성에 근거한 구성성분(similarity-based component)이 먼저 실행되고, 그 다음 설명에 근거한 구성성분(explanation-based component)이 실행된다고 일반적으로 가정된다.[11] 그러나 이러한 유사성 선행(similarity-first) 가정에 반하는 현상이 숙련된 피험자들에게는 관찰되었다. 숙련된 피험자들은 피상적인 정보만큼이나 구조적 정보도 초기에 실행된다는 것이다.

숙련된 피험자일수록 pop-out solution을 생성해낼 가능성이 높기 때문에, Novick은 pop-out solution은 anagram 철자들의 재배열에 가해지는 constraint들을 병렬적으로 처리한 결과라는 가설을 세웠다. 이 가설이 암시하는 바는 첫째 (anagram 해결 영역에서) 문제 해결에 익숙해질수록 순차(sequential) 처리에서 병렬 처리로 점차 전환된다는 것이다. 둘째는 비록 pop-out과 search solution 공히 부분적 정보(partial information)의 점진적인 축적에 의존하고 있더라도, 이 '두 해결 형태가 질적으로 다르다'는 Gestalt claim에 신빙성이 있다는 것이다.

숙련된 피험자들은 문제로 제시된 철자들이 실제 단어였을 때와 실제로 존재하지 않는 단어의 철자들이 주어졌을 때의 해결 시간에서 큰 차이를 보였다. 주어진 철자들이 spelling constraint에 아주 잘 맞는 경우였을 때, pop-out process가 중요한 역할을 하는 듯하다. 그러나 주어진 철자들이 비단어였을 때는 문제 해결에 걸린 시간은 병렬처리와 순차처리 시간의 가산보다도 더 많이 걸렸다.

영어 철자의 constraint를 최대한 만족시키는 병렬적인 철자의 재배열 과정은, 최종적으로 한 단어로 수렴되고 해결을 초래하지만, 비단어를 구성하는 철자들의 경우 해결을 낳지 못한다. 생성된 최종적인 철자배열이

11) [Keil, F. C., Smith, W. C., Simons, D. J., & Levin, D. T. (1998)] pp. 103-135.

단어가 아닐 수 있다. 이렇게 되면 해결자는 철자 재배열의 과정을 의도적으로 교체해야 한다. 이것은 해결을 찾기 위해 철자 제약을 따라야 할 필요가 없는 방식이다. 고도 숙련자들은 병렬적인 pop-out 프로세스를 사용할 가능성이 높기 때문에, 단어를 구성하는 철자들이 주어진 경우와 비단어를 구성하는 철자들이 주어진 경우의 해결 시간 차이가 고도 숙련자들에게서 더 크다는 것이다.

3. DNAgram과 Anagram

3. 1 DNAgram의 개요

전술한 바와 같이, 본 논문의 연구를 위해 실험적 방법으로 DNA Computing을 사용하였고, anagram 해결 프로그램인 DNAgram을 제안하였다. DNAgram의 구성구조가 갖는 가장 중요한 목표는, anagram 문제를 해결하는 데 인간이 사용하는 방법을 가능한 한 정확하게 흉내내는 것이다. 사실 더 구체적으로는 anagram에 능숙한 인간의 그것을 흉내낸다. 인간은 그림 1에서처럼 주어진 철자들의 집합으로부터 영단어를 찾아낸다. DNAgram은 주어진 철자들의 집합으로부터 '영단어와 같은' 후보단어를 생성한다.

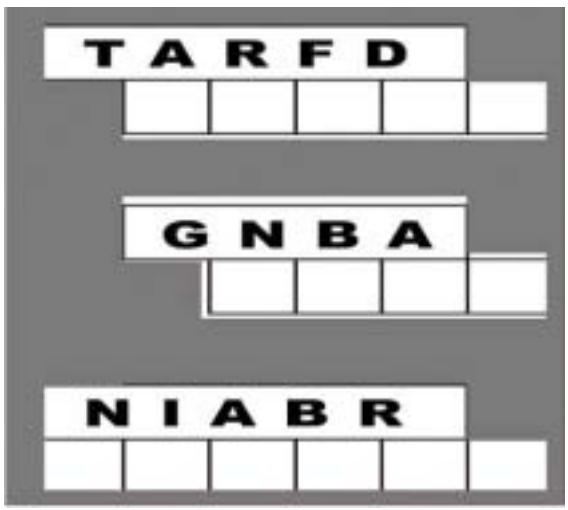

그림 1. Anagram 퍼즐

DNAgram은 영단어 사전을 상정하지 않는다. 단지 크게 보면 다음과 같은 지식만을 갖는다.

1. 어떻게 영어에서 자음, 모음 뭉치(cluster)들이 낱개의 철자들로부터

형성되는지 안다.

2. 이 뭉치들로부터 어떻게 음절(syllable)이 형성되는지 안다.
3. 이 음절로부터 단어가 어떻게 형성되는지 안다.

즉 DNAgram은 따로 떨어진 원자 단위(unit)들로 시작해서, 이들로부터 천천히 뭉치(개념을 갖는 molecule, chunk의 단위)를 형성해낸다.

왜 사전이 필요없는가? 일단 이 부분은 구현하고자 하는 인지적 과정과는 무관하기 때문이다. 능숙하게 anagram creating을 하는 사람의 머릿속에서는 후보가 될 만한 영단어들이 아주 빠른 속도로 형성되었다가 사라진다. 이들은 한마디로 문제 해결 방법으로 탐색searching보다는 '팝아웃pop-out'을 훨씬 큰 비중으로 사용한다. 즉, 이들에게 이 게임의 진행 과정은 완벽하게 무의식적이다. 문제 해결 과정이 해결자의 의식적인 방침대로 진행되지 않는다. 해결자는 '그저 바라만 본다.' 해결자들이 만들어내는 단어 후보군들 중 상당수가 진짜 영단어가 아니라는 점은 아주 중요한 점이다. 그들은 흩어진 조각들로부터 단지 아주 작은 지식만을 기반으로 정합적인 전체를 즉각적으로, 무의식적으로 형성하는 것이다.

물론 이런 과정 속에서 형성된 후보 단어가 실제 영단어인지 확인하는 작업이 있을 것이지만, 이는 DNAgram의 중요한 목표와는 무관하며, 또 대량의 사전 지식 베이스를 구축하는 것은 기존 모델들이 모두 다 썼듯이, 아주 간단한 문제에 속할 뿐이다.[12)]

3.2 Anagram 퍼즐 과제

일군의 철자들을 다시 영단어처럼 배열하는 것이 왜 중요한가? Anagram을 만드는 것은, 비록 아주 작은 문제이지만, 지능의 아주 중요한 단면을 예증한다. 이 anagram에 사용되는 방식들이 다른 영역에서도 그대로 사용되는 경우가 많기 때문이다. 즉 사람은 늘 anagram에 사용되는 방

12) 사전 검색을 통해 Anagram을 해결하는 프로그램들에 대한 자세한 설명은 [Dewdney, A. K. (1984)]을 참조.

식에서처럼, 수많은 작은 조각들을 섞어보고juggling, 결합combining하여 더 큰 조각들로 만드는 작업을 한다. 그렇게 해서 새롭고, 유의미하고, 강한 무언가를 만들어내는 것이다. 간단하게는 multi-level cleaving, splicing, regrouping, reordering, rearranging의 조작(operation) 등을 그 예로 들 수 있다. 이 조작들은 음악, 문학, 예술, 새로운 과학적 발견, 발명 등의 창조 행위(creation process)에 늘 관여하는 핵심적인 요소이다.

또한 지각 실행과정도, 다양한 층위에서 형성되는 일시적인 구조물들과 영구적인 구조물들이 상호적으로 적용되면서 작은 것들로부터 더 큰 단위를 구축하는 작업으로 볼 수 있다. 그러므로 흩어진 철자들로부터 정합적인 구조물을 만들어내는 이 놀이도 지각에 밀접하게 관련되어 있다.

청취 (예를 들어 언어, 음악 등의) 같은 시간적 감각에서는, 모든 층위에서 그 구성 성분들의 고유한 선형적 순서가 존재하지만, 그 구성성분들의 경계는 주어지지 않는다. 이 경계는 여러 번 시행하고 실수가 발생한 후에 파악된다. 가장 단순한 층위에서(가장 짧은 시간 단위에서) 좋은 추정guess이 일어나면, 더 상위에서 만들어지는 구조물이 더 정확해진다. 각 층위에서 추정을 잘 함으로써, 시스템은 자신이 추구하는 가장 상위에서의 구조물을 잘 만들 수 있을 것이다. 한편, 전체 과정도 잘 조직화되어야 한다. 그렇게 해야 각 층위에서 일어나는 추정 실패들을 언제든지 감지하고 수정할 수 있기 때문이다.

또 그림이나 사진 등 정지된 장면에 대한 시각에 대한 비시간적 감각에는 고유한 주사(scanning) 순서가 없다. 그리고 다양한 층위에서의 경계선이 어딘지에 대한 결정도 훨씬 복잡할 것이다. 그러나 여전히 이 과정 또한 작은 단위들로부터 더 큰 지각적 단위를 구축해내는 것은 본질적으로 같다.

어떤 형태의 지각이든, 수많은 행위가 전후좌우로 반복해 움직이면서 일어나게 마련이다. 즉 일시적 구조가 구축・파괴・재결합・재배열 등의 작업들을 겪어나가는 것이다. 독립적인 각각의 작업과정들이 어떻게 상호적으로 영향을 끼치는지, 구조가 어떤 식으로 결합되거나 해체되는지,

어떤 종류의 것들이 안정적인 구조물을 만드는지, 이미 만들어진 구조가 적당하지 않아 보일 때 새로운 구조를 만들기 위해서는 어떤 방법이 가장 쉬운 것인지 등에 대한 세부적인 결정을 내릴 수 있는 메커니즘을 가진 시스템만이 위에 기술한 작업들을 수행할 수 있을 것이다. DNAgram은 이러한 이론적인 측면을 쉽게 분석, 논의 할 수 있도록 실용적 목표를 두기보다는 anagram이라는 작은 영역을 과제 영역으로 택했다.

3.3 DNAgram과 Parallelism

DNA 컴퓨팅의 기본적인 개념은, DNA의 정보 저장 능력과 화학반응 능력을 이용하여 연산을 수행하는 것이다. DNA 분자들간의 화학반응은 기존 컴퓨터의 순차적 연산방식과는 달리 그 반응이 병렬식으로(즉, 모든 가능한 반응이 시험관 내에서 한꺼번에 순간적으로) 일어난다는 특성을 가지고 있다. 이는 마치 컴퓨터에서 병렬 프로세서처럼 대용량의 병렬 계산을 필요로 하는 문제들을 풀 수 있는 가능성을 제시해준다.

1994년, L. Adleman은 기존의 개념과는 매우 다른 새로운 개념의 연산, 즉 DNA 컴퓨팅에 대한 논문을 발표했다. 이를 기점으로 많은 DNA 컴퓨팅에 대한 연구들이 진행되고 있다. Adleman은 DNA의 각 염기들이 서로 상보적으로 결합하는 특징과 여러 생화학적인 실험 방법들을 이용하여 Hamiltonian 문제를 해결했다.[13)]

DNA 컴퓨팅이 최근에 부각되고 있는 이유는 DNA 분자가 가지고 있는 막대한 병렬성에 있다. 불과 1M의 농도의 DNA 용액을 생각하면, 약 1ml 정도의 액체에 대략 6×10^{20} 개 정도의 DNA 분자가 존재하고, 이러한 막대한 수의 DNA 분자 하나하나를 이용하여 정보를 저장하며 계산을 하기 때문에 그 병렬성은 매우 크다. 또한 생화학적인 실험방법들(hybridization, ligation, PCR 등)은 한꺼번에 많은 분자들을 동시에 처리할 수 있는 능력을 제공해준다. 따라서 DNA는 훌륭한 연산 능력을 가지고 있다고 할 수 있다.[14)] DNAgram은 이와 같은 DNA 컴퓨팅의 특성을 활용했

13) [Adleman, L. (1994)] pp. 1021-1024.

다.

DNAgram은 병렬 시스템이다. 이는 기존의 순차적sequential 컴퓨팅에서 이루어진 병렬 시스템과도 궤를 달리한다. 프로그램의 analogy는 세포활동에서 찾을 수 있다. 한 세포 안에서는 많은 신진대사적 활동이 다양한 공간적 위치에서 동시에 발생한다. 각각의 활동은 그것이 동화작용anabolic적(예; strand bonding)이건 이화작용catabolic적이건(예; strand breaking) 효소enzyme에 의해 수행된다. 가장 전형적인 동화작용 효소는 두 분자를 결합시키는 데 사용되는 것이다. 효소는 공간을 떠돌다가 오직 자신과만 결합될 수 있는 부분을 가진 구조와 만나면 거기에 달라붙는다. 그렇게 해서 새로운 구조가 탄생하고, 이는 또다른 효소와 만날 수 있는 구조가 된다. 따라서 작은 단위로부터 큰 구조로 무작위적인 구축이 이루어진다. 이 무작위성에도 불구하고, 정교하게 설계된 화학적 통로 때문에 최종 구조의 구축 작업은 효과적, 안정적으로 이루어진다.

DNAgram은 이런 parallelism의 구현이 가장 잘 이루어질 수 있는 방법으로 분자 컴퓨팅을 택했다. 분자 컴퓨팅의 실제 실현 방식이 위에 기술한 세포활동 비유와 가장 흡사할 뿐 아니라 비유라고도 할 수 없을 정도로 비슷한 영역에서의 활동이라는 점(DNA 염기서열의 특정 염기간의 수소 결합을 이용한다는 점만이 크게 다른 점이다), 방대한 병렬주의massive parallelism가 가상적(순차적sequential 컴퓨터에서 작동하는)이 아닌 실제로 실행될 수 있다는 점이 그 이유이다.

---

14) [Zhang, B.-T., (2001)]

4. DNAgram의 구조

4.1 규칙: 소규모 지식 베이스

DNAgram은 사전(dictionary) 검색에 의지하지 않고 비전통적인 방식으로 anagram 문제를 해결한다. 소규모의 규칙(rule)들이 문제를 해결한다. 이 규칙들은 DNAgram의 function이라고 할 수 있는데, 철자군(letter group)을 구축하거나 파괴할 때 쓰인다. 이 규칙들은 문제가 주어지면 천천히, 위계적으로 구성된 철자군을 구축해서 그럴듯해 보이는 답의 후보들을 만들어낸다. 이 프로그램의 핵심 원칙 중 하나는, 바로 이 규칙 적용의 자유경쟁적(competitive), 확률통계적(stochastic) 성격이다.

DNAgram은 보통 아무렇게나 흩어진 철자들을 받아서 그럴듯한 영어 단어 한 개를 추출해낸다. 보통의 anagram 프로그램들은 단어 사전을 검색해서 문제를 해결하지만, DNAgram는 진짜 현실 세계의 단어를 생산하는 데는 완전히 무관심하다. 이 프로그램은 단지 그럴듯한 영어 단어로 보이도록 철자들을 묶는 규칙들만 가지고 있다. 예를 들어서, 4개의 철자 o, g, u, h가 주어졌을 때, 이 프로그램은 이들을 ouhg로 묶는 것보다는 ough로 묶는 게 훨씬 그럴듯하다는 것을 안다.

다양한 철자들을 묶는 프로세스는 영어에서 철자 뭉치cluster의 빈도수frequency에 대한 인간의 잠재적 지식에 기반한다. 이 프로세스는 또한 인간이 문자를 읽을 때 계속해서 작동하는 과정이기도 하다. 그런데, 보통 인간은 이 프로세스를 전혀 의식하지 못한다. 워낙 자동적으로 일어나기 때문이다. 하지만 때때로 위계적으로 아주 그럴듯해 보이는 경로를 따라서 단어를 읽으려 하다가 읽지 못하는 경우가 생긴다. 즉 아무 생각 없이 읽어낸 단어가 현실의 영단어가 아닌 경우가 생기는 것이다. 그럴 때는 읽었던 대로의 그 철자군을 의식적으로 해체해야만 한다. 예를 들어, 'shelflist'라는 단어의 경우를 보자. 'fl'은 영어에서는 아주 흔하게 음절(syllable)의 초성에 나타나는 자음 뭉치이다. 이 사실이 머릿속에 떠올랐을 때, 전체 단어를 'shel-flist'로 파싱(parsing)하게 된다. 파싱된 두 부분 모

두 영어 단어처럼 보인다. 하지만 공교롭게도 두 단어 모두 사전에 없는 단어이다. 따라서 이 두 비단어들을 각기 해체해보려고 시도하게 된다. 여러 가지 시도가 있을 수 있다. 그중 'fl'의 'f'를 'shel' 의 'l' 뒤로 이동시키는 시도도 하게 될 것이다. 여기에는 'lf'가 음절의 끝부분에 잘 나타날 수 있는 자음 클러스터라는 지식이 작용하게 된다. 그렇게 해서 'shelf-list'로 정확한 파싱이 이루어지고, 단어의 의미도 떠오른다.

핵심은, 새롭고 더 나은 구조를 만들어내기 위해서는 이전의 구조를 해체하려고 하게 되는데, 사실 현실 세계에서는 어떤 구조가 해체되어야 하는지, 언제 해체해야 하는지 분명하지 않을 때가 훨씬 많다. 그렇기 때문에 위의 예에서 'shel-flist'로 인식하고서 그 의미를 모를 경우, 사전을 찾아볼 수도 있을 것이다. 하지만 anagram 문제 해결의 영역에서는, 사전을 찾는 순간 영역을 벗어나는 것이 된다. 즉 anagram 퍼즐게임의 핵심적인 요소가 파괴되는 것이다. 그렇기 때문에 anagram에 능숙한 사람들은 사전을 찾아보는 것을 최대한 미루고, 또 사전을 찾는 것을 게임에서의 패배로 생각한다.

4.2 병렬 스캐닝

병렬 스캐닝의 기본적인 아이디어는 다음과 같다. "가장 될 만한 것부터 비례해서 에너지를 투입한다." 문제 해결에서, 어떤 후보 해결책이 유망해 보이고, 점점 진화를 하는 것으로 보이면, 프로그램은 자신의 resource 중 더 많은 부분을 그 해결책을 발전시키는 데 투입하고, 다른 덜 유망한 후보 해결책들에게는 덜 투입하는 것이다. 이는 John Holland가 논의했던 exploration-versus-exploitation 문제와도 밀접한 관계를 맺고 있다. 설명을 위해서 그가 제시한 '두 팔 달린 슬롯머신two-armed-bandit 문제'의 간단한 예를 살펴보자.[15]

만약 동전을 넣는 투입구가 두 개 달린 슬롯머신이 있다고 하자. 한쪽 투입구에서는 세 번을 넣으면 한 번 상품이 나오고, 한쪽 투입구에서

15) [Holland, J. H. (1975)] pp. 76-83.

는 두 번에 한 번이다. 플레이어는 이 사실을 모른다. 따라서 많이 따기 위해 최대한 많은 칩을 준비한다. 만약 어떤 투입구 쪽에서 더 많은 승률을 가졌는지 안다면, 그쪽으로만 집중적으로 칩을 넣을 것이다. 하지만 그것을 모르기 때문에, 플레이어는 환경을 탐색하기 위해 처음에는 어느 정도의 칩을 사용할 수밖에 없다. 환경 탐색중 어떤 슬롯이 더 괜찮은지 느끼기 시작하면, 플레이어는 그 슬롯에 점점 더 많은 칩을 넣게 된다. 다른 슬롯에는 따라서 점점 더 적은 칩을 넣는다. 그러나, 좋은 슬롯을 발견했다고 하더라도, 전적으로 확신하는 경우는 드물다. 따라서 반드시 '나쁜' 슬롯을 때때로 '점검'하게 된다. '좋은' 슬롯을 발견했다는 확신이 강해질수록, '나쁜' 슬롯에 대한 점검 회수는 점점 적어진다. 그리고 나쁜 슬롯을 점검하는 회수가 적어질수록, 더 많은 칩(resource)을 좋다고 믿는 슬롯을 개발(exploitation)하는 데 투입하게 된다.

여기서 또한 특기해야 할 것은, 두 슬롯이 달린 기계를 사용하는 데 꼭 순차적인serial 방식을 쓸 필요는 없다는 것이다. 사실 슬롯에 칩을 넣을 때 완벽하게 병렬적parallel 방식을 쓸 수 있다. (양 손으로 각각의 두 슬롯에 동시에 칩을 넣는다면)

위의 예에서, 문제 해결 방식은 전적으로 병렬주의와 biased randomness에 의존한다. 이것이 바로 병렬 스캐닝의 원칙이다. 댐을 새로 지어 강을 막았을 때 막힌 강물처럼, 각각의 많은 '탐사의 손가락들fingers of exploration'들은 동시에 다양한 협곡과 계곡을 다양한 속도로, 그 계곡 바닥의 윤곽 형세에 따라 바깥쪽으로 나아간다.

DNAgram은 탐색(exploration)과 구조 건축(structure building)을 이러한 '많은 팔이 달린 슬롯머신many-armed-bandit' 스타일로 수행한다. 철자들을 묶거나 해체하는 작은 함수인 규칙들이 자료구조적으로 '매달려hang on' 있다. 각 규칙들은 하나 하나씩 biased, random manner로 선택, 투입된다. 즉, 무작위로 선택되지만 자신이 갖고 있던 경향에 의해 결정되는 것이다. 선택은 사실상 결코 결정적으로(deterministic) 이루어지지 않는다. 가지고 있는 urgency가 높을수록 그 효소가 수행될 가능성이 높다. 규

칙들의 수와 그들의 urgency는 프로그램의 선행 수행에 따라 다르다.

규칙이 수행하는 평가evaluation와 철자묶기(group-building)는 그 전에 이루어진 구조물의 모습에 따라 다르다는 점에서, 규칙들은 서로에게 완전히 독립적이지 않다. 예를 들어, 흩어진 철자 집합이 입력되면, 여기에서 후보 철자 집합이 무작위로(하지만 bias를 가지고) 선택된다. 이 철자 집합은 세 단계의 평가(evaluation)를 거친다. 각 단계는 연이어지며 점점 엄격한 기준을 내세운다. 이 기준으로 그 철자 집합의 quality를 판단하면서 점차 높여간다. 입력된 철자들의 '원시 수프primordial soup'로부터 anagram이 출현하는 이러한 방식이 병렬 스캐닝(parallel scan)이다.

4.3 계산 온도(Computational Temperature)

만약 프로그램이 몇 개의 후보 단어들을 구축했지만, 다음 프로세스, 즉 만들어진 철자 뭉치들과 그 뭉치에 붙지 않은 글자들을 묶어서 단어 형성을 완성하는 작업을 진행하려 할 때마다 그 묶어진 조합이 프로그램의 평가 기준을 넘지 못하면 어떻게 될까?

예를 들어서, 프로그램에 a, b, g, n이 입력되었다고 하자. 프로그램은 어떤 단계를 거쳐 세 글자의 철자 뭉치, bag을 발견하고 아주 완벽하다는 평가를 할 것이다. 이 bag은 아마도 모든 평가 단계를 다 통과하고 anagram을 형성하는 데 쓰이기 위해 작업 공간으로 들어갈 것이다. 따라서 작업 공간은 bag과 n으로 구성된다. 이때 프로그램이 bagn이라는 글롬을 만들려고 한다고 가정해보자. 자음 뭉치 gn이 음절 종성에 놓이는 것은 흔치 않다. 따라서 평가를 통과하지 못한다. 그 다음에는 nbag를 시도한다. 역시 같은 결과가 발생할 것이다. nb는 영어에서는 절대로 음절 초성이 되지 못하기 때문이다. 그렇다면 어떻게 할 것인가. 보통의 AI 프로그램이라면, deterministic backtracking을 통해 뒤로 물러날지도 모른다. 그러나 DNAgram은 deterministic backtracking strategy를 쓰지 않는다. 대신 '계산온도computational temperature'라는 메커니즘에 기댄다.

계산온도는 DNAgram 구조의 또다른 major mechanism이다. 이

것은 규칙의 urgency에 직접적인 영향을 미치는 feedback mechanism이다. 본질적으로 온도는 한 규칙이 선택되어 작동할 확률값을 주는 urgency의 범위를 프로그램에게 지시한다. 매우 낮은 온도는 프로그램인 urgency 값을 거의 deterministic한 방식으로 고려할 것임을 의미한다. 다시 말해서, 한 규칙이 30의 urgency 값을 가지고, 또 하나는 70을 가진다면, 매우 낮은 온도에서는 후자의 규칙이 사실상 언제나 선택된다. 온도가 오르면 전자의 규칙이 선택될 가능성이 점점 높아진다. 아주 높은 온도에서는 두 규칙을 선택할 가능성이 거의 비슷하다. 이때는 urgency의 실제 차이는 거의 중요하지 않게 된다.

특정 시점의 온도는 즉 그 프로그램이 자신이 만들어낸 의사단어가 얼마나 괜찮은지를 보이는 함수이다. 그 시점까지 구축한 구조가 좋을수록 온도는 낮아지고, 나쁠수록 온도는 높아진다.

DNAgram은 아마도 bag과 n을 묶으려고 여러 번 시도하겠지만 실패할 것이다. 계속된 시도와 실패의 반복 속에서 온도는 올라간다. 그러면 보통 매우 낮은 urgency 값을 가진 '해체breaker 규칙'이 작동될 가능성이 높아진다. 결국 해체 규칙이 작동되고 bag을 원래 철자들로 해체한다. 이때부터 프로그램은 다시 자유로운 시도를 하게 된다. 두 번째, 혹은 세 번째 시도에서 프로그램은 ba와 ng의 뭉치를 만들 수 있을 것이고, 'bang'이라는 좋은 의사단어(pseudo-word)를 만들어낼 수 있을 것이다.

5. DNAgram 실험

5.1 구체적 구현 절차

본 실험의 구현절차는 다음과 같다.

1. 각 알파벳을 표상하는 시퀀스 제작
2. 알파벳의 클러스터인 규칙 시퀀스 제작
3. 자/모 친숙성을 표상하는 시퀀스 제작
4. 임의의 스트랜드 구축
5. Double Strand로부터 Single Strand로 분리
6. Gel Electrophoresis
7. Affinity Separation
8. Answering

5.1.1 Step 1: 알파벳 표상 시퀀스

26개 철자의 알파벳을 표상하는 시퀀스를 인코딩하는 단계이다. 각 철자를 나타내는 DNA 코드는 다음과 같다.[16)]

a: GAAATGAGTT	n: TGATTATGTC
b: TGATGCTACA	o: ACATATCGGT
c: GGTTGTGGCG	p: CTCTTCAACG
d: CAGTTATTTC	q: AGAAGAAACT
e: GGCTACGATG	r: CTGTCGAGTG
f: GAGCACTGCG	s: ACATTTCACC
g: AGACGACGAC	t: ATCACGTAAT
h: GTAGAGGTAC	u: CAGAACGGAT
I: TCTCAGAGAT	v: TGTGTAGTGC

16) 시퀀스 생성 방법은 부록 참조.

j: TCGATTTGGA	w: CGCGGCGATG
k: TAGGCAGATG	x: CTATGCTGGG
l: ACAGCACTAC	y: AACTTGGCGC
m: TCGGATAGGA	z: GCATGATCAT

각 시퀀스의 양쪽 끝에는 그림 2에서처럼 자음, 혹은 모음을 나타내는 시퀀스가 붙어 있다. 이는 보통 자음-모음, 모음-자음의 연결이 많은 반면 자음-자음, 모음-모음의 연결은 흔치 않은 데서 기인한다.

그림 2. 위의 시퀀스는 자음(Lc), 아래 시퀀스는 모음(Lv)을 표상한다. 각 시퀀스의 양쪽 끝에는 자/모음과 잘 붙을 수 있도록 반대되는 연결음 표상 시퀀스를 붙여 놓는다.

5.1.2 Step 2: 규칙 시퀀스

규칙 시퀀스는 anagram을 해결하는 인간의 머릿속에 들어 있을 것으로 가정되는 지식 베이스이다. 예를 들어 영어 단어에서 ‘s’와 ’h’는 서로 연결 강도가 강하지만, ‘w’와 ’g’는 연결 강도가 약하다. 규칙 시퀀스들은 이렇게 어떤 철자들이 어떤 철자와 서로 어떤 순서대로 연결이 잘 되어 있는지에 대한 지식을 나타낸다.(그림 3) 따라서, 이 규칙 시퀀스들이 적으면 적을수록 단어 형성은 어려움을 겪을 것이다. anagram에 익숙한 사람일수록 이 시퀀스들의 양이 많아질 것으로 추측된다.

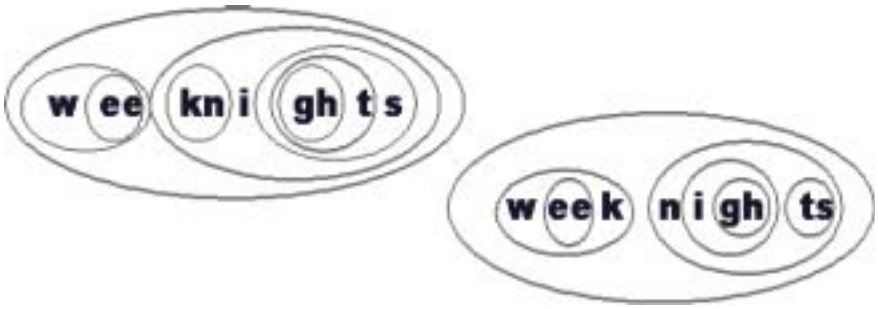

그림 3. 어떤 규칙을 적용하느냐에 따라 'weeknights'는 'wee-knights' 또는 'week-nights'로 읽힐 수 있다.

예를 들어 철자 't' 'h' 'e'가 주어졌다고 하자.(그림 4) 't'와 'h'의 친숙도는 강력하다. 따라서 't-h' chain을 형성할 가능성이 높다. 자음과 모음은 서로 붙을 가능성이 높으므로 나머지 e와도 체인 형성 가능성 높다. 따라서 't-h-e'가 형성될 수 있다.

그러나 만약 자-모의 일반적 친숙도 지식밖에 없어 'h'와 'e'의 결합이 이루어졌다면 't'는 결합할 곳이 없어 아래 그림처럼 't-h-e'의 연결을 나오기가 힘들 것이다.

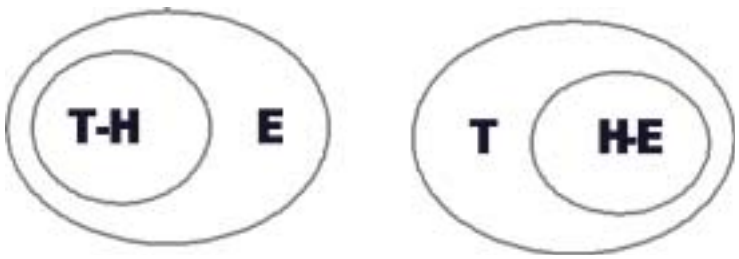

그림 4. 철자 't' 'h' 'e'의 연결.

물론 규칙은 인간의 지식을 표상하는 것이므로 그 수는 마음대로 조절할 수 있다. anagram 초보자의 경우 이 규칙 베이스는 상대적으로 적을 것이며 숙련자의 경우 규칙 베이스 용량은 클 것이다. 규칙들 중 일부는 다음과 같다.[17]

a:	*ai* (bail, pair)	*au* (haul, fault)	*ay* (bay, play)
b:	*bl* (black, bliss)	*br* (brand, broth)	
c:	*ch* (chat, chess)	*cr*(crab, crack)	*ct*(act, fact)
d:	*dr* (drop, drink)	*dw* (dwell)	
e:	*ea* (year)	*ee* (green)	e*i*(reinheir)
f:	*ff*(off, cuff)	*fl*(flab, flash)	*fr*(frog, fresh)
...			
...			

이 규칙 시퀀스의 생성은 DNA 염기의 상보적 결합을 이용한 것으로, 기존의 알파벳 시퀀스의 상보적인 염기 서열을 사용한다. 즉 'ng'의 규칙을 나타내는 시퀀스는 'n'과 'g' 시퀀스와 결합시키기 위해 'n-모음-모음-g'의 시퀀스를 생성한다.(그림 5)

그림 5. 규칙 'ng' 시퀀스의 형태

17) 모의 실험에 사용된 규칙 베이스는 부록 참조.

5.1.3 Step 3: 모음-자음, 자음-모음 규칙 시퀀스

Step 1에서 언급했듯이, 일반적으로 알파벳 연결은 자음-모음, 모음-자음의 경우가 대부분이다. 이같은 자모 친숙성을 고려하여 시퀀스를 생성한다. 이 두 시퀀스는 각각 6개의 염기로 구성되어 있다. 이 시퀀스는 자-모, 모-자의 순서로 상보적인 single strand로 존재하다가, 자음과 모음의 적당한 알파벳 시퀀스와 만나는 순간 두 철자를 결합하는 역할을 한다. 또 이 시퀀스는 각 알파벳의 양끝에 붙어서 이 자-모, 모-자 시퀀스와 결합함으로써 철자 클러스터를 생성하는 역할도 한다.

그림 6에서처럼, 각 철자를 나타내는 DNA 코드 양끝에는 자음 철자인 경우 모음을 나타내는 시퀀스를 붙이고, 모음 철자인 경우 자음을 나타내는 시퀀스를 붙인다. 이렇게 해서 자-모의 일반적인 친숙성을 구현한다. 자음을 나타내는 코드는 GCATAG, 모음의 경우는 GGATGG이다. 따라서 자음의 양끝에는 GGATGG가 붙어 있고, 모음의 양끝에는 GCATAG가 붙어 있다. 또 이러한 자-모 결합 annealing을 위한 자-모 시퀀스는 그 상보적 코드를 이용해 자-모 시퀀스는 CGTATC-CCTACC, 모-자 시퀀스는 CCTACC-CGTATC가 된다. 이렇게 구현한 알파벳 시퀀스 집합은 그림 7과 같다.

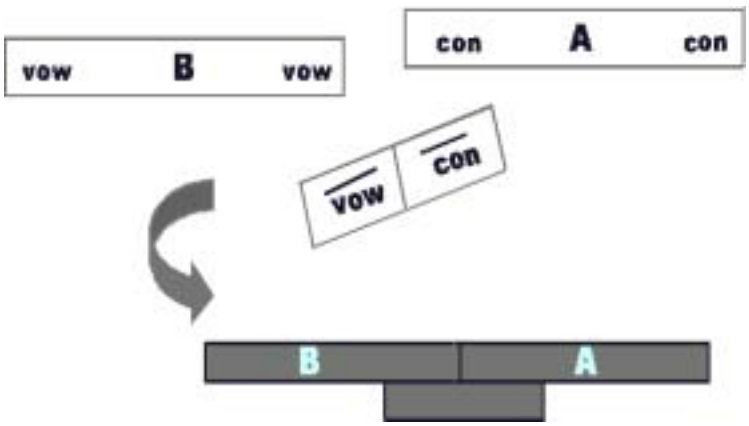

그림 6. 알파벳 'B'와 'A'의 결합 형태

Alphabet	DNA sequence
A	GCATAGGAAATGAGTTGCATAG
B	GGATGGTGATGCTACAGGATGG
C	GGATGGGGTTGTGGCGGGATGG
D	GGATGGCAGTTATTTCGGATGG
E	GCATAGGGCTACGATGGCATAG
F	GGATGGGAGCACTGCGGGATGG
G	GGATGGAGACGACGACGGATGG
H	GGATGGGTAGAGGTACGGATGG
I	GCATAGTCTCAGAGATGCATAG
J	GGATGGTCGATTTGGAGGATGG
K	GGATGGTAGGCAGATGGGATGG
L	GGATGGACAGCACTACGGATGG
M	GGATGGTCGGATAGGAGGATGG
N	GGATGGTGATTATGTCGGATGG
O	GCATAGACATATCGGTGCATAG
P	GGATGGCTCTTCAACGGGATGG
Q	GGATGGAGAAGAAACTGGATGG
R	GGATGGCTGTCGAGTGGGATGG
S	GGATGGACATTTCACCGGATGG
T	GGATGGATCACGTAATGGATGG
U	GCATAGCAGAACGGATGCATAG
V	GGATGGTGTGTAGTGCGGATGG
W	GGATGGCGCGGCGATGGGATGG
X	GGATGGCTATGCTGGGGGATGG
Y	GGATGGAACTTGGCGCGGATGG
Z	GGATGGGCATGATCATGGATGG

그림 7 자-모 시퀀스와 결합된 알파벳 시퀀스 집합

5.1.4 Step 4: 단어 생성

Step 1에서 생성한 시퀀스들을 한꺼번에 결합시키면 무작위 시퀀스 결합이 이루어진다. 그러나 그림 8에서와 같이 Step 2와 3에서 생성한

시퀀스들을 여기에 첨가하면 규칙 시퀀스와 자모 시퀀스들의 결합으로 적절한 단어가 생성된다. 이렇게 해서 만들어진 시퀀스는 Watson-Crick의 상보적 결합을 이루어 보통은 Double Strand 형태를 띤다.

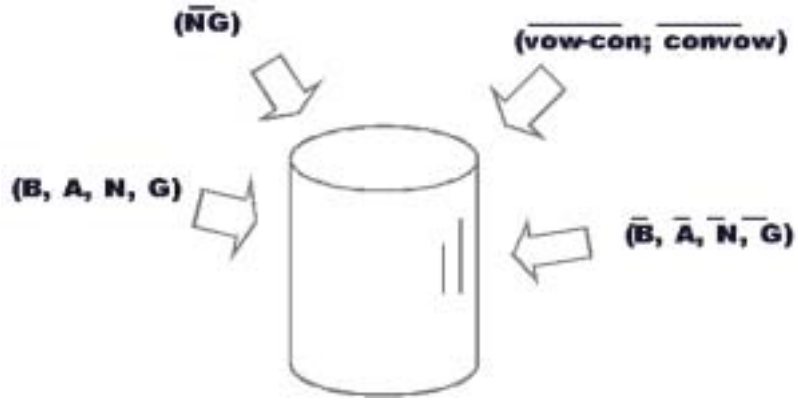

그림 8. 'bang'의 단어 시퀀스 생성

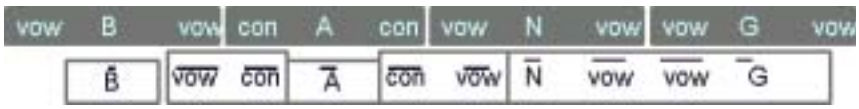

그림 9. 단어 'bang'가 형성된 경우의 형태

그림 9에서처럼 목표로 삼은 단어 시퀀스가 생성된다. 그러나, 원치 않은 단어가 생성되는 경우도 있다. 그림 10에서 볼 수 있듯이 'bag'나 'nag', 'ang' 같은 경우가 그렇다.

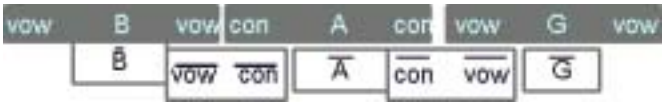

그림10. 단어 'bag'가 형성된 경우의 형태

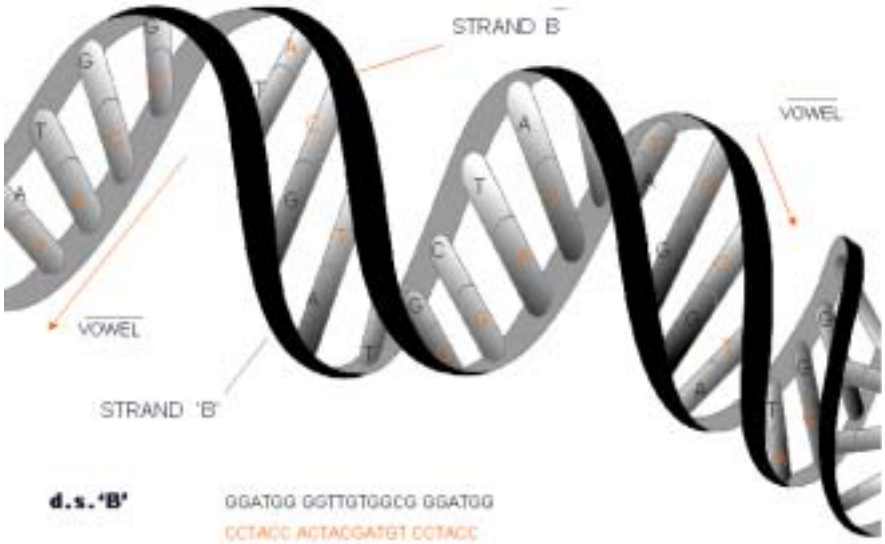

그림 11. 시퀀스 생성 후 알파벳 'b'의 형태

5.1.5 Step 5: Double Strand to Single Strand

생성된 double strand를 온도를 높여 그림 12처럼 single strand로 분리한다. 이 strand를 정답 후보로 설정하고 heating을 통해 분리된 다른 시퀀스들이 이후 다시 결합할 수 있도록 만든다.

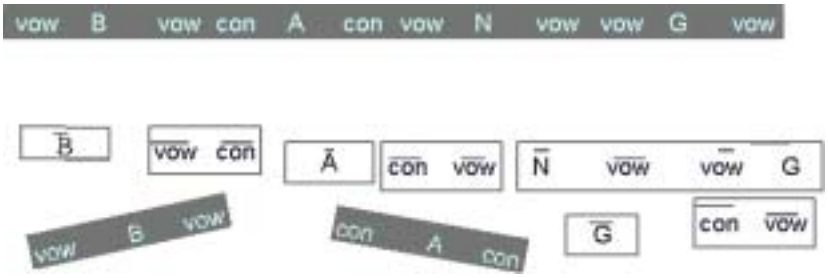

그림 12. double strand로부터 'bang'을 표상하는 single strand 추출

5.1.6 Step 6: Gel Electrophoresis

단어 생성이 무작위로 이루어졌기 때문에, 여러 가지 다른 답이 나올 수 있다. Step 6과 Step 7은 이러한 오답 단어 시퀀스를 제거하기 위해 사용되었다. 정확한 답이 가져야 할 조건은 위의 규칙들이 적용되는 것 외에, 요구하는 단어 시퀀스가 정확히 n 개의 철자로 구성되어야 하고, 문제로 제시된 각 철자들이 한 번씩 사용되어야 한다.

먼저, 요구하는 단어 시퀀스가 정확히 몇 개의 철자로 구성되었는지, 즉 원하는 정답 시퀀스의 길이는 Gel Electrophoresis를 통해 알아낼 수 있다.(그림 13) 예를 들어 'bang'의 경우 4개의 철자로 구성되지 않은 strand는 제거한다.

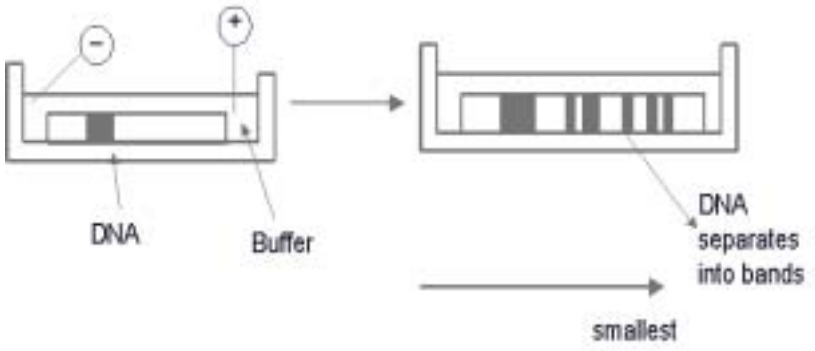

그림 13. Gel Electrophoresis

5.1.7 Step 7: Affinity Separation

남은 시퀀스들 중 문제로 주어진 철자를 한 번씩 사용하는 시퀀스만을 골라내고 나머지는 제거한다. 최종적으로 튜브 안에 남은 시퀀스는 정답이 된다.(그림 14)

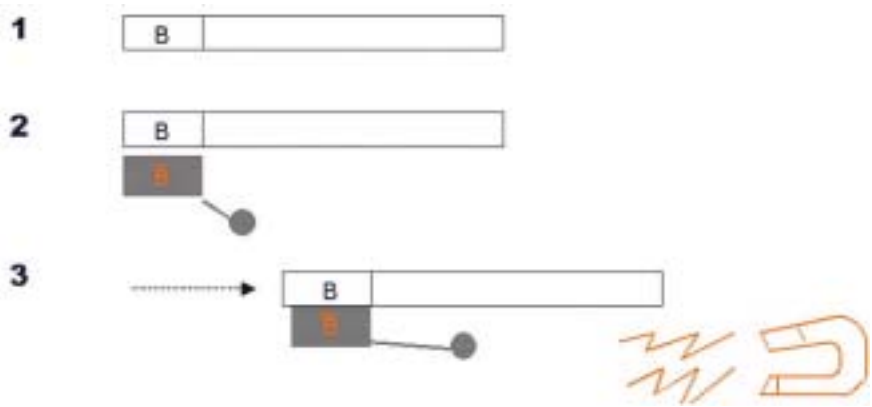

그림 14. Affinity Separation: 'b'를 가진 시퀀스만을 골라낸다.
'b'의 상보 시퀀스를 probe molecule로 붙인 bead를 이용한다.

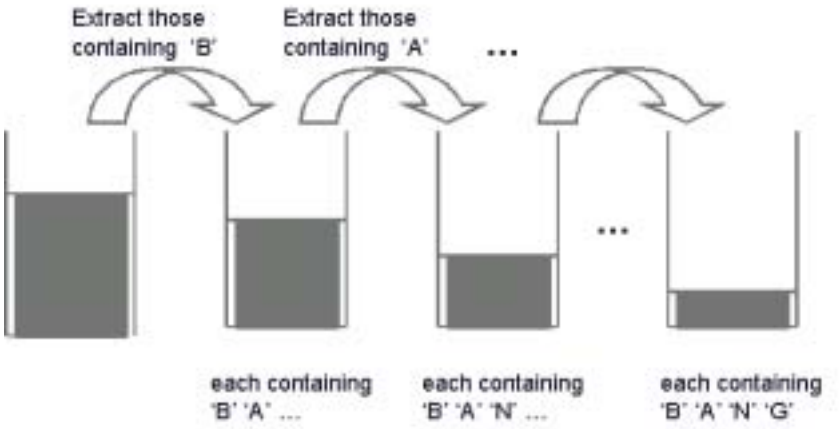

그림 15. 목표 시퀀스를 가지고 있지 않은 시퀀스들을 점차 제거하여 정답을 찾는다.

5.1.8 Answering

남은 시퀀스가 있으면 그 시퀀스를 정답으로 간주하고, 시퀀스가 없으면 정답이 없는 것으로 보고한다.

5.2 컴퓨터 모의 실험

위에서 언급된 모든 절차들을 적용하여 주어진 anagram 문제를 해결할 수 있는지 살펴보기 위해 컴퓨터 모의 실험을 수행하였다. 특히 알파벳의 연결 규칙지식 시퀀스의 양을 조절하여 문제를 시행해보고 지식 시퀀스의 양과 정답 단어로 남겨진 시퀀스의 양을 비교하여 살펴보았다.

먼저, 적용된 연결 규칙 시퀀스의 양을 조절하는 것은 anagram 퍼즐을 해결하는 사람에게 행해진 규칙 학습의 강도를 조절하는 것에 해당한다. 규칙에 대한 학습이 어느 정도 강화된 사람은 그 규칙을 적용하여

anagram을 해결하려 할 것이다. 그러나 학습이 되어 있지 않으면 그 규칙을 적용하지 않을 것이고, 적용이 되었다 하더라도 그것은 임의적인 결과라고 할 수 있다. 다음으로 이와 같이 규칙 시퀀스의 학습량이 클 때와 작을 때의 오답 단어, 즉 'gang' 'bag' 'gab' 'nag' 'ang' 같은 단어들의 산출량을 비교해본 것은 규칙 학습의 적용이 오답 단어들에게도 적용이 되어 나타나는지, 산출량의 변화가 있다면 그것이 규칙 학습량과 어떤 관계를 가지는지를 알아보기 위함이었다.

규칙 시퀀스의 종류는 실험자가 가지고 있는 지식을 직관적으로 정리해 넣었다. 앞서 언급한 바와 같이, 이 규칙 시퀀스에 객관적 수치의 테이블은 DNAgram에 존재하지 않는다. 또 심리학적 실험으로 입증하거나 비교할 수치도 존재하지 않는다. 그 이유는 첫째 DNAgram은 anagram을 해결하는 expert system이 아니기 때문이다. 단지 특정 개념을 유연하게 형성하는 추상적 모델이다. 따라서 '정확한 수치'를 구하는 것은 의미가 없다. 또다른 이유는 완벽한 수치 테이블 자체가 anagram 해결자의 머릿속에 없기 때문이다. 규칙 집합은 오히려 주관적인 '성향'에 가깝다. 따라서 실험자가 대충 정한 규칙 집합이 심리학적 사실성에 충실하지 못할 이유가 전혀 없다.

물론, 규칙 집합이 random하게 설정되었다면, 즉 여러 사람의 성향이 뒤섞인 채 설정되었다면, 그 결과의 전체적인 행동 성향은 이해 불가능할 것이다. 마치 화성인이 anagram을 푼 결과처럼 보일 것이다. 따라서 규칙 테이블은 실험자 일인이 최대한 가지고 있는 규칙 집합을 구성하여 투입했으며, 이로써 프로그램의 결과가 그럴듯하게 보일 수 있을 것이라 보았다.

프로그램은 Java-based GUI로 구성하였다.(그림 16) 프로그램에는 각 알파벳을 입력하고 그 시퀀스의 양을 임의대로 입력할 수 있다. 또 규칙 베이스 윈도우를 따로 만들어 역시 원하는 규칙과 그 양을 입력할 수 있다. 프로그램은 정답 단어를 명시해서 출력할 뿐 아니라 오답으로 자체 평가한 후보 단어들도 같이 출력한다. 각 단어들의 hybridization 횟수도

명시 출력되도록 설계하였다.

그림 16. DNAgram의 인터페이스

5.2.1 방법

해결 과제: ‘b′ ′a′ ′n′ ′g′ 철자들을 입력하였다. (1) 정답인 ’bang′을 출력하는지 살펴보았다. (2) 규칙 베이스에 들어 있는 ‘ng′의 양을 조절하여 이 지식이 정답 단어 출력에 어떠한 영향과 관계를 가지는지 살펴보았다. 먼저 ’ng′ 규칙 시퀀스를 넣지 않고 단지 ’b′ ′a′ ′n′ ′g′만을 입력하였을 때 ‘bang′를 출력하는지의 여부를 보고, ’ng′ 규칙을 넣었을 때의 양상을 비교하였다. 정답 ‘bang′과 함께 출력된 ’bag′ ′gab′ ′nag′ 등등

의 후보 단어들의 규모를 보면 'ng' 규칙을 넣었을 때의 규모를 비교할 수 있을 것이라 생각하였다. 그 다음 'ng' 규칙의 양을 일정 정도씩 변화시키면서 정답 산출 규모의 변화를 살펴보았다. 'ng'의 양이 늘어나는 것은 영어에서 철자 뭉치cluster의 빈도수frequency에 대한 인간의 잠재적 지식, 그 연결 강도가 강해지는 것을 의미할 수 있다고 생각하였다.

각 철자 시퀀스의 입력 개수는 각각 1000개, hybridization 횟수 1000개로 고정시키고, 모의 실험을 진행하였다. 실제 DNA 컴퓨팅 실험과 비교하면 무척 적은 숫자이지만 최소한 본 연구의 기본 함의를 유지할 수 있을 것으로 보았다.

5.2.2 실험 결과

먼저 'ng' 규칙 시퀀스를 각각 0개, 1000개 투입했을 때 프로그램이 각각 어느 정도의 정답 후보 단어를 내놓는지, 또 그 양은 어느 정도인지 살펴보았다. 'ng' 시퀀스를 0개 투입시 실험을 1000회 시행하여 그 평균값을 산출한 후 출력된 후보 단어들의 개수를 그림 17에 정리하였다. 정답인 'bang'은 0.5회 출력되었고, 다른 유사 후보 단어들(nag, bag, nab, gab)의 출현 횟수가 상대적으로 많았다. 'gang'은 상대적으로 적게 나타났다. 그 다음 'ng' 시퀀스를 1000개 투입시 실험을 역시 1000회 시행하여 같은 방법으로 출력된 후보 단어들의 개수를 그림 18에 정리하였다. 눈에 띄는 것은 정답인 'bang'이 상대적으로 많게(4.9회) 나타났고, 유사 후보 단어인 'gang'이 많아졌다는 것이다. 특히 유사 후보 단어인 'ang'은 앞의 조사에서는 거의 산출되지 않았으나 이번에는 5.4회 산출되었다.

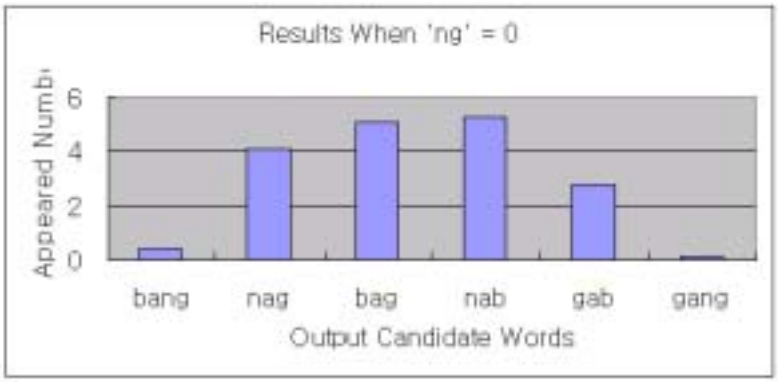

그림 17. 규칙 'ng' 시퀀스가 0개 투입되었을 때

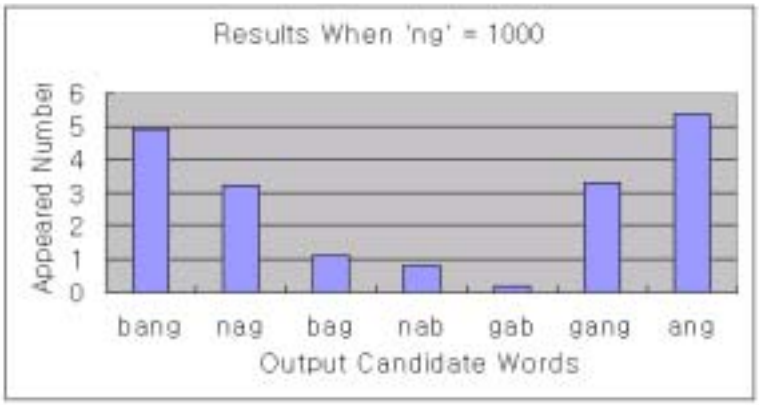

그림 18. 규칙 'ng' 시퀀스가 1000개 투입되었을 때

두 번째로 규칙 'ng' 시퀀스를 0개부터 1500개까지 50개 간격으로 투입 후 실험을 각각 100회씩 시행한 결과를 평균으로 산출하여 정답 'bang'의 출력 개수를 조사하였다. 'ng'가 0개였을 때는 'bang'의 출현 횟수가 0.5회였지만, 이미 'ng' 개수 50개 투입시부터 그 횟수가 2.5개로 상

승했다. 그러나 이후 지속적으로 출현 횟수가 상승하지는 않았다. 투입 개수가 1000개일 때 출현 횟수가 4.9회로 가장 높았고, 이후 1500개 투입까지 그 횟수는 비례하지는 않았다. 이와 같은 결과는 규칙이 학습된 후 정답 산출량이 급등했고, 이후로는 점차 상승했다는 점에 비추어 심리학적으로 주목할 필요가 있다. 이 결과를 그림 19에 정리하였다.

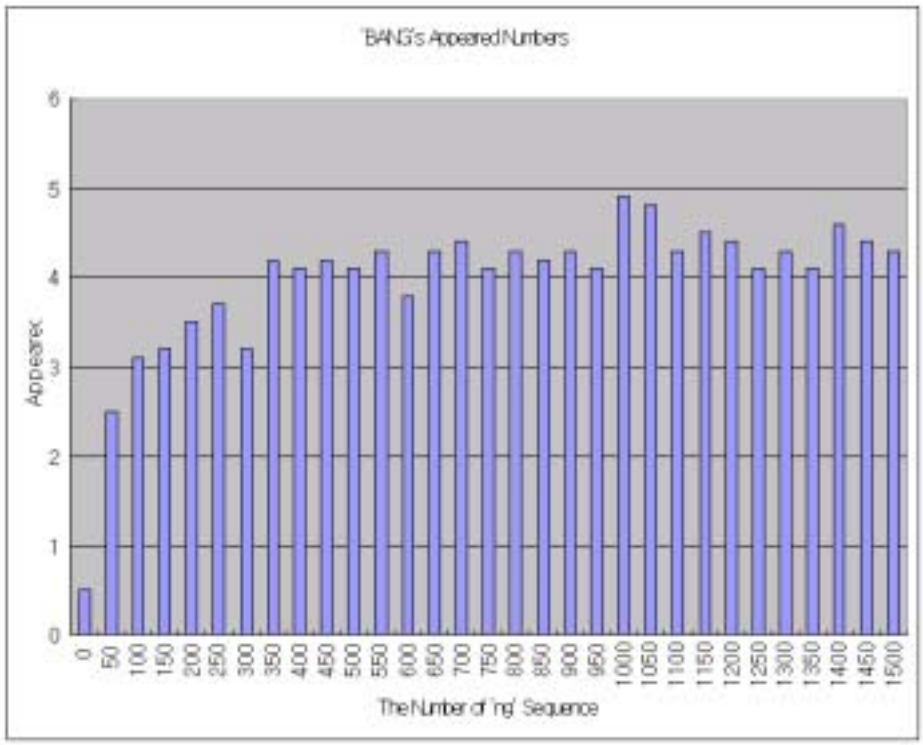

그림 19. 규칙 'ng' 시퀀스를 0개부터 1500개 투입했을 때
'bang'의 출력 횟수 변화

6. 결론

6.1 연구 결과 요약

지금까지 anagram 퍼즐 문제를 해결하는 DNA 컴퓨팅 시뮬레이션을 통해 지각 과정의 작은 단면을 살펴보았다. 먼저 anagram이라는 작은 영역이 왜 인간 인지의 중요한 측면을 보여주는지에 대해 심리학적 선행 연구를 통해 밝히려 했고, 하위 지각이 어떻게 상위 지각으로 연결되어 이루어지는지 철학과 심리학 분야의 연구들을 분류하여 살펴보았다. 하위 지각의 무수한 작용들이 병렬적으로 일어남을 모델링하기 위해 DNA 컴퓨팅이 필요함을 보였고, 기존의 인공지능 이론들을 살펴보고 이들의 뒤를 잇는 새로운 방법론으로서 DNA 컴퓨팅에 대해 고찰하였다. Anagram은 심리학 분야에서 많이 연구되었다. 또한 anagram을 기존의 순차적 컴퓨팅으로 해결하는 프로그램도 흔하다. 이 소재는 독립적인 연구 주제로서도 의미가 있겠으나, 다양한 방법을 통한 통합적인 접근도 중요하다고 생각한다. 앞에서 언급했듯이 이 주제는 인간 인지의 중요한 단면을 보이고 있으며, 특히 지각 과정의 특징적이면서 중요한 부분을 복합적으로 끌어내는 까닭이다.

순수 이론적 연구를 보완하는 실험적 방법으로서 DNA Computing을 이용하였다. 모의 실험을 통해 문제 해결에 있어 searching solution과 pop-out solution 중 pop-out solution의 프로세스를 시뮬레이션했다. pop-out solution은 입력된 아이템들의 단순한 재배열에 가해지는 constraint들을 병렬적으로 처리한 결과일 수도 있음을 실험을 통해서 보였다. 소규모의 constraint set만으로도 원하는 답을 얻을 수 있다는 결과에 주목할 필요가 있다. 이는 주어진 constraint set이 암묵적으로 memorized 되었을 때, 이것이 문제 해결에 있어 피상적 특징 성분이 아닌 구조적 성분으로 바로 작용한다는 것을 의미한다. 즉 문제 해결에서 초보자는 피상적 성분을 사용하고, 전문가는 추론이 뒷받침되어야 가능한 구조적 성분을 초기에 잘 사용한다는 점을 상기할 때, 실험 결과가 이 점을 뒷받침한다는

것이다.

6.2 향후 과제

본 연구의 연장선상에서 수행되어야 할 과제는 크게 두 가지 측면을 들 수 있다. 그 하나는 심리학적, 언어학적으로 anagram에 적용될 수 있는 constraint들에 대한 연구이고, 또 하나는 DNA Computing의 기술적 측면의 연구이다. 먼저 constraint의 측면에서, 본 연구에 적용된 규칙 집합을 확장하여 심리학적으로 더욱 유연한 문제 해결을 추구할 필요가 있다. 실제 지각 실험을 병행하여 본 연구의 결과를 강화하는 경우도 검토해보아야 할 것이다. 인지과학이 표상 형성의 문제를 심각하게 고려하는 한, 그 다음의 문제는 분명히 존재하는 인간의 상위 지각 프로세스의 유연성을 어떻게 다룰 것인지가 될 것이다. 이러한 유연성의 근원을 상위 지각 층위에서만 찾아서는 안 된다고 생각한다. 대규모 병렬 프로세스가 하위 지각에 어떠한 영향을 끼치는지에 대해, 또 이 하위 지각 프로세스들이 상위 지각으로 연결되면서 어떤 constraint들의 영향을 받는지에 대한 연구가 필요하다.

DNA Computing은 대규모 병렬 프로세스를 구현하기에 가장 적합한 방법론으로 생각된다. 본 연구는 이러한 이유로 DNA Computing을 사용하였지만, 아주 적은 수의 기본적인 연산만을 적용했다. 기술적으로 적지 않은 제약이 있는 것도 사실이나, 기존의 방법들을 더욱 정교하고 적절하게 적용시켜 실험을 진행할 필요가 있다. DNA Computing의 기술적 특징들을 잘 적용한다면 인지 과정의 시뮬레이션에 있어서 또다른 가능성을 제시할 것이라고 생각한다.

참고문헌

Adleman, L. (1994) “Molecular computation of solutions to combinatorial problem,” *Science* 266: 1021-1024.

Dewdney, A. K. "Computer recreations: a computational garden sprouting anagrams, pangrams, and few weeds," *Scientific American* 251, no. 4 (October 1984): 20-27.

Elman, J. L. (1990) "Finding structure in time," in *Cognitive Science*, vol. 14, pp. 179-212.

Feynman, R. (1961) “There's plenty of room at the bottom,” *Minaturization*, D. Gilbert (ed.), Reinhold, pp. 282-296.

Holland, J. H. (1975) *Adaptation in Natural and Artificial Systems*. Ann Arbor, Mich.: University of Michigan Press, pp. 76-83.

—. et al. (1986) *Induction*. Cambridge, MA: Bradford Books/MIT Press.

Humphrey, G. (1963) *Thinking*. New York: Wiley and Sons.

James, W. (1990) *The Principles of Psychology*. Chicago: Encyclopedia Britannica, Inc., 2nd ed. of *The Principles of Psychology,* 1890. New York: Henry Holt. pp. 222-224.

Keil, F. C., Smith, W. C., Simons, D. J., & Levin, D. T. (1998). "Two dogmas of conceptual empiricism: Implications for hybrid models of the structure of knowledge," *Cognition*. 65: 103-135.

Kirkpatrick, S., C. D. Gelatt, Jr., & M. P. Vecchi. (1983) "Optimization

by simulated annealing," *Science* 220: 671-680.

Lakoff, G. (1987) *Women, Fire, and Dangerous Things: What Categories Reveal about the Mind*. Chicago: University of Chicago Press. p. 159.

Maley, C. C. (1998) "DNA Computation: Theory, Practice, and Prospects," *Evolutionary Computation* 6(3): 201-299.

Marr, D. (1982) *Vision*. San Francisco: W. H. Freeman & Co., p. 358.

Mitchell, M. (1993) *Analogy-Making as Perception*, Cambridge: The MIT Press, pp. 235-244.

Newell, A., & Simon, H. (1963) "GPS, a program that simulate human thought," in Feigenbaum, E., & Feldman, J. (eds.) *Computer and Thought*, New York: McGraw-Hill, pp. 279-293.

—. (1976) "Computer science as empirical inquiry: symbols and search," in J. Haugeland (ed.), *Mind Design II,* Cambridge: The MIT Press, pp. 81-110.

Novick, L. R., & Cote, N. (1992) "The nature of expertise in anagram solution," *Proceedings of the Fourteenth Annual Conference of the Cognitive Science Society,* Hillsdale, NJ: Erlbaum, pp. 450-455.

Rumelhart, D. E., McClelland, J. L., & the PDP Research Group, (1986) *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Vol. I: Foundations,* Cambridge, MA: Bradford Books/MIT Press.

Shank, R. C. (1980) "Language and memory," *Cognitive Science*, 4(1): pp. 243-284.

Smolensky, P. (1988) "On the proper treatment of connectionism," *Behavioral and Brain Sciences* 11(1): 1-14.

—. (1987) "Connectionist modeling: neural computation / mental connections," in J. Haugeland (ed.), *Mind Design II*, Cambridge: The MIT Press, pp. 233-250.

Zhang, B.-T. (2002) "Molecular Information Processing Technologies (in Korean)," *The Magazine of the Institute of Electronics Engineers of Korea*, 29(3): 58-70.

부록

부록 1. 각 시퀀스 생성 방법.

알파벳, 자-모음 시퀀스는 Genetic algorithm을 사용하여 만들었다. 시퀀스를 만드는 문제이므로 답은 시퀀스들이 되고, 적합한 정도는 시퀀스들이 서로 얼마나 다른지, 실험 조건은 일정한지 등이 된다. 시퀀스를 만들 때 고려한 조건은 다음과 같다.

- 전체 시퀀스에서 G나 C가 차지하는 비율이 50% 정도에 가까울 것
- 같은 염기가 연속해서 많이 나타나지 않을 것
- 서로 다른 시퀀스끼리 많이 비슷해지지 않을 것
- 서로 다른 시퀀스끼리 많이 상보결합하지 않을 것

부록 2. 규칙 시퀀스 테이블.

Alphabet	Rule	Example
a	ai	pair
	au	fault
	ay	play
e	ea	meat
	ee	green
	ei	vein
	eu	europe
	ey	monkey
i	ie	friend
o	oa	boast
	oi	void
	oo	hook
	ou	count
u	ue	fuel
	ui	fruit
b	bl	black
	br	brand
c	ch	chat
	cr	crook
	ck	back
	ct	fact
d	dd	add
	dr	drop
	dw	dwell

Alphabet	Rule	Example
f	ff	cuff
	fl	flab
	fr	frog
	ft	left
g	gg	egg
	gh	laugh
	gl	glow
	gn	assign
	gr	grab
k	kn	knight
l	lb	bulb
	ld	hold
	lf	self
	lk	milk
	ll	kill
	lm	film
	lp	help
	lt	belt
m	mb	bomb
	mp	camp
n	nd	land
	ng	bang
	nk	bank
	nt	hunt

Alphabet	Rule	Example
p	ph	graph
	pl	plant
	pr	prompt
	pt	kept
r	re	refine
s	sc	scan
	sh	ship
	sk	desk
	sl	slab
	sm	smack
	sn	snack
	sp	spell
	ss	mass
	st	stack
	sw	swim
t	th	bath
	tr	track
	tt	matt
	tw	twist
w	wh	whim
x	xt	text
z	zz	fuzz

ABSTRACT

DNAgram: Molecular Simulation of Anagram Problem Solving

Eun-Seok Lee
Interdisciplinary Program in Cognitive Science,
Graduate School of Seoul National University

This paper describes a molecular simulation, called DNAgram, that models how people make anagram puzzle. This work demonstrates the procedure of extracting plausible English word answer by molecular computing, which exposes new information and provides and new perspective on existing anagram creation models and its methods.

This research chose two hypotheses: 1) D. Hofstadter's assertion that intelligence emerges out of the interactions of many thousands of parallel processes that take place within milliseconds and are inaccessible to introspection, 2) L. Novick's assertion that pop-out solutions result from parallel processing of the constraints on the rearranged order of the anagram letters, whereas search solutions result from a serial hypothesis testing process. To investigate these, I performed the computer simulation, under the principles of DNA computing, which was used by L. Adleman, to accomplish the parallelism implied in hypotheses above.

For the psychological plausibility of the experiment, this research excluded the use of searching solution with dictionary database. Only with constraints table of given anagram letters, the program eliminated the possibility of semantic procedure.

I found that DNAgram produced the right answer word under this condition. Then, I assumed that the number of constraint rules inserted would be the factors that affect solution difficulty as a function of anagram problem solving. I increased gradually the number of constraints, from 0 to 1500, during the experiment. The result was also consistent with the hypothesis. The right answer extraction number was radically increased after the insertion of constraints, and increased gradually till the constraint number 1,000 gradually. After the constraint number 1,000, answer extraction number had no intense change. This result should primarily reflect operation of the pop-out process of the anagram problem solving.

This solution results makes sense from the current perspective that only with a few constraints, linguistic perceptual processes can be occurred. And this work showed that DNA computing can be a useful methodology for cognitive simulation.

Keyword: Anagram, DNA Computing, Parallel Process, Constraints Set

감사의 글

고맙습니다... 먼저 이 논문의 처음부터 끝까지 부족한 저를 이끌고 도와주신 장병탁 선생님께. 이 논문에 하나라도 칭찬 받을 점이 있다면 그것은 모두 선생님의 것입니다. 언제나 제 짧은 생각 하나하나 놓치지 않고 들어주시고 연구를 진행할 수 있도록 격려해주신 선생님 면목에, 부족하나마 이 논문을 계속 쓸 수 있었습니다. 또 뵐 때마다 저의 부족함을 일깨워주시는 김정오 선생님께. 논문의 문제점을 세심하게 짚어주시고, 앙상한 논문에 풍성한 의미를 넣어주신 그 노고에 감사드립니다. 논문 심사 때나 수업 때 선생님께서 지적해주신 것들이, 제게는 비단 이 논문뿐 아니라 인지과학이라는 학문에 대한 성찰의 대상이 되었습니다. 늘 따뜻한 관심과 함께 논문의 심사를 맡아주신 김기현 선생님께도 깊은 감사의 뜻을 전하고 싶습니다.

부족한 제 아이디어가 논문의 형식이나마 갖추게 된 데는 다음 분들의 수고 때문입니다. 같이 실험 절차를 논의하고 시뮬레이터를 만든 윤지은 씨, 실험 과정을 찬찬히 살피고 제언을 준 영균, 시퀀스를 만든 이인희 씨. 이들의 도움이 없었다면 논문은 나오지 못했을 것입니다.

마음 편히 연구할 수 있었던 것은 무엇보다도 '인지과학 사람들' 덕입니다. 자신 없어하는 후배를 도닥거려주느라 바빴을 종호 형, 지난 2년 내내, 즐겁고 따뜻한 대화 속에서도 날카로운 지적을 늘 잊지 않았던 윤현 형, 바쁜 생활 와중에도 온갖 자료와 경험을 전수해준 소영 누나, 같은 실험실에서 같이 논문 쓰며 힘이 되어준 경수 형, 자주 만날 수 없었지만 늘 관심을 보여준 성호, 현애, 그리고 충명 형에게. 함께 한 시간은 짧지만 많은 것을 가르쳐준 종윤, 정우 형에게. 선배들과의 끊임없는 대화 속에서 셀 수 없이 많은 것을 얻었고, 또 그 베풂에 늘 고마웠습니다.

과정 동기 희정, 희재, 경순, 호준에게. 직장생활을 하다가 진학한 터라, 어색할 수도 있었던 대학원 연구 생활에 자연스럽게 적응할 수 있었던 것

은 이들이 보여준 끊임없는 관심 덕분입니다. 늘 즐겁게 얘기할 수 있었던 후배 의준, 현승, 권현... 그밖에 혹 빠진 사람 있을까 걱정되는 인지과학 협동과정의 모든 이들에게 각별히 감사드립니다.

고마움의 정 이전에 한없는 미안함이 앞서는 이들도 있습니다. 바쁜 직장 생활에도 늘 걱정해주고 격려해준 경훈, 정섭에게, 온갖 짜증을 다 받아주고, 또 온갖 고민을 자신의 것처럼 함께 해준 진경에게, 혼자서 오빠 뒷바라지하느라 바빴을 동생 민영에게, 고맙다는 말을 다시 한번 전하고 싶습니다.

마지막으로 사랑하는 부모님께. 저는 누구보다도 부모님께 가장 큰 신세를 졌습니다. 부모님께만은 언제나 기대어 쉴 수 있었고, 깊은 지혜와 영감, 용기를 얻을 수 있었습니다. 무엇보다도, 철없는 아들을 늘 품어주신 그 끝없는 신뢰가 고맙습니다. 이 논문을 부모님께 드립니다.