論文2000-37CI-3-3

확률적 평가에 기반한 컴퓨터 바둑의 후보 생성 시스템

(A Candidate Generation System based on Probabilistic Evaluation in Computer Go)

金 永 祥 *, 柳 基 永 **

(Young Sang Kim and Kee Young Yoo)

요 약

바둑이 진행될 때마다 적절한 후보 자리를 계산하는 모델이 있다면, 사례를 이용하지 않고도 후보 생성 알고리즘의 표준으로 정립될 수 있다. 본 논문에서는 바둑을 조합 게임론에 따라서 분석하고, 흑, 백간 영역의 차이를 반면의 형세로 나타내는 확률 행렬(PM)을 기반으로 임의의 국면에 대한 후보를 생성하는 후보

본 논문에서 설계된 CGS는 임의의 국면에 돌이 놓여지면 영향력, 안정도, 살아남을 확률 값(PV), 확률 행렬(PM)을 계산하고, 현 국면에 대한 적당한 후보를 생성한다. CGS의 기본 전략은 현재의 반면에 대해서 다섯 개의 후보를 생성하고 그 중에서 PV가 높은 지점을 최종 후보로 선정한다. CGS는 공격보다 방어에 주력하였으며 정석 사례를 사용하는 NEMESIS에 비해서 사례를 전혀 구축하지 않은 CGS가 초반에 있어서 다소 우세함을 보여준다.

Abstract

If there exists a model that calculates the proper candidate position whenever the game of Go is in progress, it can be used for setting up the prototype of the candidate generation algorithm without using case-based reasoning. In this paper, we analyze Go through combinatorial game theory and on the basis of probability matrix (PM) showing the difference of the territory of the black and the white. We design and implement a candidate generation system(CGS) to find the candidates at a situation in Go.

CGS designed in this paper can compute Influence power, safety, probability value(PV), and PM and then generate candidate positions for a present scene, once a stone is played at a scene. The basic strategy generates five candidates for the present scene, and then chooses one with the highest PV. CGS generates the candidate which emphasizes more defence tactics than attack ones. In the opening game of computer Go, we can know that CGS which has no pattern is somewhat superior to NEMESIS which has the *Joseki* pattern.

* 正會員, 濟州漢拏大學 컴퓨터情報科

(Department of Computer Information, Cheju Halla College)

** 正會員, 慶北大學校 컴퓨터工學科

(Department of Computer Engineering, Kyungpook National University)

接受日字:1999年 7月 19日, 수정완료일 : 2000年 3月27日

Ⅰ. 서 론

인간의 추론 행위를 컴퓨터에 실현시키려는 노력은 인공 지능 시스템의 지원 이론으로 널리 주목받아 왔다. 바둑의 경우에는 기존의 추론 행위 표현 방식과 차이가 나는데, 그것은 상대방의 수를 자신의 수와 마찬가지로 고려해야 하며, 다음 수의 생성에 영향을 미치게 된다는 점이다. 바둑은 명확하고 단순한 규칙으

로 정의된 세계에서 두 경기자가 교대로 흑, 백의 돌을 번갈아 놓으면서 자신의 영역을 최대한으로 넓게 구축하는 지능 게임이므로, 컴퓨터 바둑 프로그램은 게임 진행에서의 자신의 행위가 극대화될 수 있도록 다음 수를 스스로 판단하여 생성할 수 있어야 한다. 따라서 바둑 국면의 형세를 정확하게 판단하도록 어떠한 방식이든지 형세 평가의 지능을 심어주어야 하는 문제가 대두되었다.

단순히 빈자리만을 고른다면 컴퓨터가 탐색해야할 영역은 기하급수적으로 방대해진다. 이러한 문제의 해결 방안으로 MIN/MAX 방식이나 AND/OR pruning 방식을 이용한 탐색 공간을 제한하는 방법, 전체 탐색 트리의 단계와 너비만을 제한해서 탐색하는 방법, 정석에 대한 사례를 저장하고 패턴 매칭(pattern matching)을 통해서 동일 패턴을 대응시키는 방법 등의 시간과 공간의 제약을 덜 받는 방법들이 제안되어 왔다. 하지만 바둑은 유기적 인과성이 매우 강한 현실 세계를 그대로 반영하는 게임이기 때문에, 컴퓨터 바둑은 각각의 국면마다 최선의 가치를 가지는 다음의 한 수를 생성할 수 있어야 한다. 이를테면 각 부분 전투가 어떤 과정으로 진행되어 왔으며, 어떻게 끝날 것인가에 대한 전반적인 구도를 자신의 지능으로 구축할 수 있는 새로운 접근 방법을 모색할 수 있어야 한다.

컴퓨터 바둑을 연구하는 그룹들은 사활 문제풀이와 같은 제한된 범위 내에서 이루어지는 특정한 상황에 대한 알고리즘을 개발하거나 전체 프로그램을 개발하는 두 가지 측면으로 분류할 수 있다[1][2][3]. 바둑은 양자간에 점령해야 할 중립적인 지역을 두고, 서로의 이해 대립이 불가피한 전면전이기 때문에 현재의 국면에 대한 형세 평가가 다음 수를 생성하는데 지대한 영향을 미치게 된다.

바둑이 진행될 때마다 적절한 후보 자리를 계산하여 수를 진행하는 알고리즘을 설계한다면 사례를 이용하지 않고도 후보를 생성할 수 있다는 점에 착안하여 본 논문에서는 확률 행렬(Probability Matrix:PM)을 이용한 후보 생성 시스템[4]을 제안한다. 제안된 후보 생성 시스템의 기본 전략은 확률 행렬을 기반으로 국면의 안정도와 사활을 평가하여 임의의 국면에 대한 다음 수를 생성한다. 바둑판의 각 자리에 값이 있다고 가정하고 각 자리에 놓여진 돌들의 영향력과 안정도를 계산한 후, 각 자리가 살아남을 확률 값을 계산하여 그 중에서 큰 값을 가지는 자리들을 후보 자리로 결정한다.

본 논문의 구성을 보면, 제 2장에서는 바둑을 조합 게임론(Combinatorial Game Theory:CGT)에 의해서 분석하고, 흑, 백간의 형세 차이를 나타내는 확률 행렬의 개념을 제시하였다. 그리고 제 3장에서 후보 생성 시스템(Candidate Generation System:CGS)의 설계 구조를 제안하였고, 제4장에서는 구현한 후보 생성 시스템을 실험한 결과와 평가를 보였으며 제5장에서는 결론 및 향후 연구 방향을 논의한다.

Ⅱ. 문제 정의

1. 조합 게임론적 분석

바둑은 두 사람이 흑과 백의 바둑돌로 바둑판의 교차점에 교대로 착수하여 쌍방이 차지한 집의 많고 적음을 따져서 승패를 가리는 제로 합 게임이다. 착수란 바둑돌을 바둑판의 교차점에 놓는 것으로, 바둑이 끝났을 때를 제외하고 교대로 한 수씩 놓는 것을 원칙으로 한다. 한번 착수한 돌은 그 돌에 인접하여 비어 있는 교차점이 있는 동안 계속 존재하며 완전히 둘러 쌓이게 되면 바둑판에서 들어낸다. 이러한 규칙에 준하여 유효한 착수가 끝난 뒤 공배를 메우면 대국이 종료되는데, 집 속에 있는 상대방의 죽은 돌을 들어내어 따낸 돌과 합친 후, 이 돌들을 상대방의 집에 메우고 나서 남아 있는 집의 수가 많은 쪽이 승자가 된다.

CGT는 하나의 게임을 지역적인 여러 개의 부분 게임으로 분해할 수 있다는 점에서 바둑의 행위를 표현하기에 가장 적합하기 때문에 바둑을 분석하는 접근 방법으로 매우 유용하다. 바둑판을 평면 그래프로 표현하면 $G_1=(V_1, E_1)$과 같이 19×19의 교차 정점과 그들을 잇는 선분으로 구성된다.

$$\begin{aligned} G_1 &= (V_1, E_1) \\ V_1 &= \{ (i,j) \mid 1 \le i, j \le 19 \} \\ E_1 &= \{ ((i,j),(i',j')) \mid (i=i' \wedge |j-j'|=1) \vee \\ &\quad (|i-i'|=1 \wedge j=j') \} \end{aligned} \quad (1)$$

이때 G_1에 대해서 돌의 색을 고려하여 표현하면, 식 (2)와 같다.

$$\begin{aligned} & S = \{ Black, White, Empty \} \\ & f: V_1 \rightarrow S \\ & \\ & G=(V, E) \\ & V=\{(v, f(v)) \mid v \in V_1(G_1) \} \\ & E=\{((u, f(u)),(v, f(v)) \mid (u,v) \in E_1(G_1) \wedge f(u)=f(v)\} \end{aligned} \quad (2)$$

그러므로 바둑판은 식 (2)와 같이 흑, 백, 빈자리 중에서 하나의 색을 가지는 정점 V와 이 정점들을 연결한 선분 E의 집합, G=(V, E)로 표현할 수 있다.

그래프 G상에서 Empty가 아닌 같은 색의 돌이 인접한 선분을 따라서 서로의 위치에 도달할 수 있는 경로가 있을 때 이를 이어짐(connected)이라고 하고, 이것을 G에 대한 부분그래프(subgraph)로서 식 (3)과 같이 표현할 수 있다. 두 개의 돌 사이에서 선분의 수 즉, 경로의 길이는 1이다.

$$\text{for } u,\ v \in V(G)$$
$$u \ conn \ v = \{(u = v) \vee \exists w \in V(G), ((u, w) \in E(G) \wedge w \ conn \ v)\} \quad (3)$$

덩어리(block)는 같은 색의 돌들이 모두 이어짐의 조건을 만족하는 경우를 말한다. 즉 ∀u, v ∈ B, B ⊆ V(G)에 대해서 B가 덩어리라는 것은 u conn v의 관계를 만족하는 것이다. 단 f(u) = f(v)는 Empty가 아니어야 한다.

덩어리 B에 인접하는 빈자리(liberty)와 빈자리의 개수 L_0는 식 (4)를 만족한다. 덩어리가 가질 수 있는 정보는 덩어리를 구성하는 돌의 색 및 개수, 인접한 빈자리 수, 안정도 등이며 삶과 죽음을 같이한다.

$$\text{for } B \subseteq V(G),$$
$$Liberty(B) = \{(z,\ f(z)) \mid z \notin B, \exists y \in B, (y, z) \in E(G) \wedge f(z) = Empty\}$$
$$L_0(B) = |Liberty(B)| \ \therefore \ number\ of\ Liberty\ in\ B \quad (4)$$

고리(Chain)는 잠재적인 덩어리를 연결하기 위한 수단이다. C ⊆ V(G) 인 'C가 고리'라는 것은 다음의 예로서 표현할 수 있다. 만약 같은 색을 가지는 두 개의 덩어리 B_1, B_2가 있다고 하자. 여기서 e가 B1에 속하고, f가 B_2에 속하며, 또한 g와 h가 B1, B2 어느 쪽에도 속하지 않는 정점일 때, {(e, g), (e, h), (g, f), (h, f)}와 같은 선분의 집합이 존재하면 B_1, B_2는 g 또는 h에 의해서 새로운 덩어리로 생성된다. 이러한 고리의 색은 반드시 Empty이고, e와 f사이의 경로의 길이는 2이다.

마찬가지로 색이 다른 두개의 덩어리 B_1, B_2가 있을 때 덩어리 B_1에 대한 덩어리 B_2의 경계 정점, B1\B2는 식 (5)와 같이 표현할 수 있다.

$$B_1 \backslash B_2 = \{(x, f(x)) \mid x \notin B_1, x \notin B_2, x \in (Liberty(B_1) \cap Liberty(B_2)\}$$
여기서, $f(x) = Empty$ (5)

만약 B_1과 B_2이 같은 색을 가진다면 경계정점, $B_1 \backslash B_2$는 고리이다. 이렇게 같은 색의 덩어리가 인접하여 경계 정점이 되는 고리를 가지게 되면 이는 새로운 하나의 객체로 표현할 수 있는데 이를 대마(group)라고 하고, 식 (6)과 같이 나타낼 수 있다.

$$Group = (V',\ E')$$
$$V' = \{((x, f(x)) \mid x \in B_1) \cup ((z, f(z)) \mid z \in B_2) \cup ((y, f(y)) \mid y \in B_1 \backslash B_2)\}$$
$$E' = \{(x,\ y) \cup (y,\ z) \cup E(B_1) \cup E(B_2)\}$$
여기서 *Empty*가 아니면서 $f(x) = f(z)$을 만족하면 $(x\ conn\ y)$ 및 $(y\ conn\ z)$ 가 성립된다. (6)

따라서 대마는 B_1과 B_2의 합집합(union) 그래프로서 V(Group)=V(B_1)∪V(B_2), E(Group)=E(B_1)∪E(B_2)를 만족한다.

또한 그래프 G상의 빈자리의 집합을 H라고 정의하면 식 (7)로 나타낼 수 있으며,

$$H = (V'',\ E'')$$
$$V'' = \{(v, f(v)) \mid v \in V(G) \wedge f(v) = Empty\}$$
$$E'' = \{((u, f(u)), (v, f(v)) \mid (u,\ v) \in E(G) \wedge f(u) = f(v)\} \quad (7)$$

이때 식 (8)과 같이 B ⊆ V(G) 인 덩어리의 내부를 Int(B), 덩어리의 외부를 Ext(B)라고 정의할 수 있다.

$$Ext(B) = \{(v,\ f(v)) \mid v \notin B,\ f(v) = Empty \wedge B \subseteq V''(H)\}$$
$$Int(B) = \{(v,\ f(v)) \mid v \notin B,\ f(v) = Empty \vee B \subseteq V''(H)\} \quad (8)$$

따라서 식 (4)로 표현한 덩어리 B의 인접 빈자리에 대해서 Liberty(B) = Ext(B) ∩ V″(H)가 성립되며, 만약 L_0(B) 가 0이면 B는 바둑판에서 더 이상 존재할 수 없다.

Ⅱ. 확률 행렬

1. 확률적 평가 요소

두 경기자, X와 Y가 진행하는 제로 합 게임에서 매번 같은 규칙이 적용된다고 가정하였을 때, 제일 좋은 전략은 X 혹은 Y가 한가지 행위만을 반복하지 않는 것이다. 따라서 바둑의 각 국면마다 착점 위치를 다르게 선정해야 하고, 그 행위는 상대에게 손해를 주는 만큼 자신에게 이득이 되게 하는 제로 합을 유지해야 한다. 이때의 안정점은 Y의 전략 y_1, y_2에 있어서 X가 어떤 행위를 취하더라도 X자신의 이익 혹은 손해에 아무런 변화가 없어야 한다. 이는 Y의 전략에 있어서도 마찬가지이므로 X가 이길 확률은 X에 대해서 최대이고, Y에 대해서는 최소이다.

바둑에서 다음 놓을 자리를 고르는 문제는 같은 국

면이라 하더라도 다양하게 선택하게 된다. 임의의 국면에서 바둑판의 19×19 자리마다 검은 돌 또는 흰 돌이 살아남을 확률이 존재한다. 이는 두 경기자 중에서 X의 차례, 혹은 Y의 차례인가에 따라서 확률 행렬은 19×19자리의 확률들을 모두 더한 것으로 바둑의 형세를 나타내게 된다. 이러한 확률 값은 0 에서 1 사이에 속하며, 1에 가까울수록 검은 돌이 살 확률이 강하며, 0에 가까울수록 흰 돌이 살 확률이 강한 것으로 볼 수 있다[4].

바둑판에서 돌은 가장 기본적인 객체로서 색, 인접 빈자리 수, 안정도, 바둑판에서의 위치, 다른 돌에의 영향력 등의 정보를 가진다. 돌이 덩어리 또는 대마의 일부로 존재하게 될 때 안정도는 그 돌의 생사 여부를 결정짓는 중요한 요소가 된다. 안정도를 계산하기 위한 함수는 [6]에 정의된 바와 같이 fin, adj, state, check, num 등으로 나눌 수 있는데 이를 이용해서 돌 또는 덩어리의 세력을 계산하게 된다. 또한 다른 대마나 덩어리끼리의 연결 정도는 [6]에서 제시한 바와 같이 1-family 에 속하는지 e-family에 속하는지에 따라서 구분되며 어떤 대마가 두 개의 e-family에 속해 있을 때 한쪽이 차단된 경우, 다른 쪽을 선택케 함으로서 대마의 안정도를 높이는 연결이 가능해진다. 혹은 안정도가 낮을 때 이 대마를 포획해서 나의 대마가 1-family를 형성할 수 있는가에 따라 연결 정도의 강약이 결정된다.

임의의 국면에서 각 돌들은 자신의 색깔에 따라서 주위에 영향력(Influence Power : IP)을 끼치게 된다. 만약 같은 색의 덩어리이면 그 덩어리의 IP는 현재보다 확장되고, 다른 색의 덩어리가 인접해 있으면 그 덩어리의 IP는 감소하게 된다. 흑의 IP에 대한 부호는 양수(+)이고, 백의 부호는 음수(-)이다. 각 자리의 IP는 연결의 강약 및 안정도에 따라 0.0에서 1.0사이의 확률 값(Probability Value:PV)로 대응하게 된다. 어떤 한 자리에 대해서 양의 부호를 가지는 IP는 $0.5 < PV \leq 1.0$ 범위의 PV로 대응되며, 음의 부호를 가지고 있다면 $0.0 \leq PV < 0.5$ 범위의 PV로 변환될 수 있다. 어떤 자리의 IP가 0이면 그 자리의 PV는 정확히 0.5이다. 안정도의 계산은 [6]에서 정의된 바와 같이 0이면 죽음을 나타내고 1이면 완생을 의미한다.

2. 확률 함수

바둑판은 흑(Black), 백(White)의 색을 가지는 두 경기자에 의해 변화될 수 있다. 돌이 놓여진 임의의 국면을 χ라 하자. 만약 $y_1,...,y_n$을 χ의 계승자(successor), 즉 χ에서 놓을 수 있는 빈자리들에 의해서 진행된 하나 하나의 국면들이라고 하자. 그러면 χ에서 계승자들의 집합, $\Gamma(\chi) = \{y1,...,yn\}$이다. 또한 $\chi \circ a$는 χ에 a를 착수한다고 정의하면, 현재 국면에서 다음 국면으로의 차례 넘김은 $\Lambda½$ ($\Lambda½ \circ \Lambda½ = \Lambda$) 이다. 따라서 $y_1,...,y_n$은 돌의 놓임과 놓을 차례를 포함한다.

여기서 $C_1,...,C_n$를 놓을 만한 자리를 고를 확률, 즉 χ에서부터 $y_1,...,y_n$이 될 확률이라고 하자. 그러면 호선(互先) 바둑에서 현재의 국면 χ의 확률 함수, 즉 바둑이 종료된 국면에서 각 자리를 흑이 차지할 확률, $P(\chi)$는 식 (9)로 표현할 수 있다.

$$P(\chi) = \sum_{i=1}^{n} C_i P(y_i) \tag{9}$$

계승자들 가운데서 $P(\chi)$가 최대가 되는 것이 m개라면 확률 함수는 식 (10)과 같이 변한다.

$$P(\chi) = \frac{1}{m} \sum_{i=1}^{m} P(y_i) \tag{10}$$

식 (9), (10)에 의해서 현재의 국면 χ로부터 $P(\chi)$를 구하고, 귀납적으로 $\chi \circ a \circ P(\chi)$로부터 $P(\chi \circ a)$를 구할 수 있다. 돌을 놓음으로서, 전 변화량 $\Delta I \triangleq P(\chi \circ a \circ \Lambda½) - P(\chi)$가 되고, 놓을 차례가 넘어감으로서, 후 변화량 $\Delta II \triangleq P(\chi \circ a) - P(\chi \circ a \circ \Lambda½)$가 된다. 따라서 어떤 한 점을 놓고 나서 ΔI와 ΔII의 모든 원소는 같은 부호라고 할 때 확률 함수의 변화량은 식 (11)과 같다.

$$\begin{aligned} \Delta P(\chi \circ a) &\triangleq P(\chi \circ a) - P(\chi) \\ &= \Delta^{I} + \Delta^{II} \end{aligned} \tag{11}$$

마찬가지로 ΔF를 같은 색의 돌(Friend)이 놓였을 때의 변화량이라고 할 때, Δ^{I}와 Δ^{II}의 모든 원소가 같은 부호를 가지고 있다면 식 (12)가 성립한다.

$$\Delta_F = \Delta_F^{I} + \Delta_F^{II} \tag{12}$$

이때 Δ_F^{I}의 모든 원소는 0보다 크거나 같고, Δ_F^{II}의 모든 원소는 0보다 작거나 같다.

그러므로 $a \in \Gamma(\chi)$라면 $\chi \circ a$ 국면에서 변화량을 모두 더한 값은 0이 된다. 임의의 국면에서 어느 한 점에 돌을 놓음으로서 그 교점에 대한 확률 값의 변화량

이 최대가 될 때, 이 자리를 후보 자리라고 하면 이들은 하나 이상이 될 수 있다. 따라서 바둑의 모든 국면에서 후보 자리들의 집합이 있다고 가정하면 어떤 국면 α의 형세는 식 (13)으로 나타낼 수 있으며, α에서의 후보 자리의 집합은 식 (14)와 같다.

$$V(\alpha) \triangleq \sum_{i=1}^{19}\sum_{j=1}^{19} P(\alpha, i, j) \tag{13}$$

$$\Gamma_{Mm(\alpha)} \triangleq \{a \in \Gamma(\alpha) \mid \forall x \in \Gamma(\alpha)(V(\alpha \circ a) \gamma_\alpha V(\alpha \circ x))\} \tag{14}$$

여기서 연산자, $\gamma\alpha$는 식 (15)로 정의한다.

$$\gamma_\alpha \triangleq \begin{pmatrix} \text{'}\geq\text{'}\ Turn(\alpha) = Black(\text{흑 차례로서, V값이가장크도록한다}) \\ \text{'}\leq\text{'}\ Turn(\alpha) = White(\text{백 차례 로서, V값이 가장 작도록 한다}) \end{pmatrix} \tag{15}$$

그러므로 α에서 선택된 후보 자리, (i, j)교점에 대한 확률 함수는, α가 더 메울 곳이 없는 끝난 바둑일 때 식 (16)과 같이 나타낼 수 있다.

$$P(\alpha, i, j) \triangleq \begin{cases} \dfrac{1}{|\Gamma_{Mm(\alpha)}|} \sum_{a \in \Gamma_{Mm(\alpha)}} P(\alpha \circ a, i, j) \\ 1 \ (i,j) : \text{흑 차지} \\ \frac{1}{2} \ (i,j) : \text{빅의 빈자리} \\ 0 \ (i,j) : \text{백 차지} \\ 0.5 < P \leq 1.0 \ (i,j) : \text{흑 영향권} \\ 0.0 \leq P < 0.5 \ (i,j) : \text{백 영향권} \end{cases} \tag{16}$$

국면이 갱신될 때마다 식 (13), (14), (16)은 순환적으로 적용된다. 임의의 국면 α에서의 후보 자리는 식 (13)의 V값을 계산하여, 흑 차례일 경우는 가장 크게 점이다. 따라서 $V(\alpha) = V(\alpha \circ a)$ 이면 a가 후보 자리에서 밀려날 까닭이 없다.

$$a, b \in \Gamma_{Mm(\alpha)} \Rightarrow V(\alpha \circ a) = V(\alpha \circ b)\} \tag{17}$$

그리고 식 (18), (19)에서와 같이 후보 자리들의 집합은 현 국면의 형세와 a를 놓고 난 후의 형세가 변함없이 유지되도록 할 때 그 의미가 있으며, 임의의 국면 α에서의 확률 함수와 $\alpha \circ a$에서의 확률 함수에 대한 변화량의 합은 0이다.

$$\Gamma_{Mm(\alpha)} \Leftrightarrow \{a \in \Gamma(\alpha) \mid V(\alpha) = V(\alpha \circ a)\} \tag{18}$$

$$\sum_{a \in \Gamma_{Mm(\alpha)}} (P(\alpha, i, j) - P(\alpha \circ a, i, j)) = 0 \tag{19}$$

Ⅲ. 후보 생성 시스템(CGS: Candidate Generation System)의 설계

1. 객체 지향적 자료 구조

CGS는 [4][5][6]의 구현 버전이므로 본 논문에서는 [5]에서 정의한 자료 구조를 기반으로 정적인 자료와 동적인 자료로 구분하였으며 CGS의 구현에 사용한 주요 자료 구조는 다음과 같다.

(1) 바둑판 윈도우 클래스

```
class BOARD
{
        RECT            BoardRect;  // 클라이언트 표면 지정
        PTDispWindow TDisp;     // 국면의 정보를 표시하는 윈도우
        Stone           Color_Board[Max_Grid][Max_Grid];  // 현 국면의 돌의 색 저장
        BYTE            Init_Board[Max_Grid][Max_Grid];  // 초기화
        POINT           Dead_Array[Max_Moves];  // 사석을 저장
  public:
        BOARD();
        ~BOARD();
        void        DrawBoard(HDC); // 바둑판을 그려줌
        void        GetValidGrid(POINT& p, POINT *);  // 19 x 19 상의 자리인지 체크
        void        SetColor(BYTE what, POINT& p) // Color_Board에 색을 저장
        void        DrawStone(BYTE what, POINT& p) // 돌을 윈도우에 그려준다
           void        ClearStone(POINT& p) // 돌을 지워서 판에서 들어낸다
        short       GetSequence(POINT& p) // 진행 수순을 얻어 온다
        void        Clear_Board ();
        int         Count_Block (BYTE t);
        int         Check_Dead (int x, int y, BYTE color); // 생사 여부 체크
        int         Move_possible (int x, int y, BYTE color); // 놓을 수 있는 자리 여부
        BOOL        Suicide (int x, int y); // 자살 자리인지 여부
        BOOL        Domination(int x, int y) ; // 패인지 판단
        void        Undo_action();  // 무르기
        void        SaveGame(char *); // 저장
        BOOL        Check(POINT p);  // 유효한 자리인지 체크
};
```

(2) 바둑 판의 형(덩어리 및 색깔 저장)

```
typedef struct gibo_type {
      SHORT   color;
      short      group;
} GiboType;

typedef struct pan_type {
            short            turn;
      GiboType      gibo[Max_Grid][Max_Grid];
      GroupType     *groups[Max_Groups];    // 덩어리의 리스트
} PanType;
```

(3) 윈도우상의 돌 객체

```
class Stone
{
          BYTE    StoneColor;          // 돌의 색깔
      short  StoneSequence;       //현재의  돌이 몇번째 수순인가?
   public:
      void Color_init()  // 빈자리로 해 준다.
      void SetColor(BYTE s )      // 돌의 색깔 지정
      void SetStoneSequence(short s )      // 돌의 수순 지정
      BYTE GetColor()             // 현재 돌의 색깔을 얻음
      short GetStoneSequence()                  // 현재 돌의 수순을 얻음
      BOOL StonePlace()          // 임의의 돌이 있는가? 여부
      HFONT CreateCassFont(int hSize, int vSize); // 돌위에 수순 표시
};
```

2. CGS의 전략

프로그램에 사용된 정적 자료는 각 자리마다 [5]에서 주어진 가중치에 따라 무게를 부여한 것이고, 동적 자료는 돌이 놓인 자리에 이웃한 빈자리의 수라든지 덩어리의 활로와 같은 지역적 자료로서, 국면이 바뀔 때마다 변하게 되고, 없어지거나 새로운 형태로 바뀔 수 있다. 바둑판에 돌이 놓여지면 그 국면에 대한 동적 자료는 다시 계산하여야 한다. 재 계산이 요구되는 동적 자료는 돌이나 덩어리에 대한 IP, 안정도, PV, PM 등으로 구분할 수 있다. CGS의 전체적인 개요도는 그림 1과 같다.

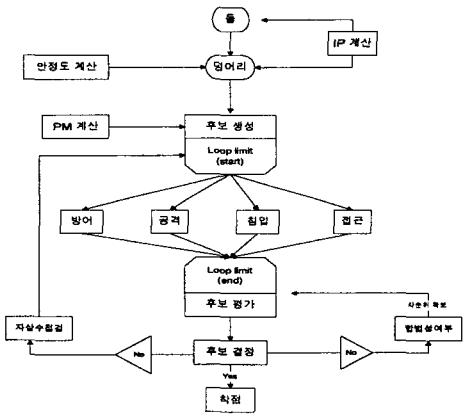

그림 1. 후보 생성 시스템의 개요도
Fig. 1. General chart of CGS

임의의 국면에 돌이 놓여지면 홀로 격리되는지 아니면 덩어리 또는 대마의 일부로 소속되는지에 따라 IP, PV, 안정도, PM의 계산이 달라진다. 본 논문에서 정의하는 PM은 바둑이 진행되는 모든 국면에 대해서 국면의 19×19 자리마다 검은 돌 또는 흰 돌이 살아남을 PV의 산술적 합을 계산하는 구조를 가진다. 여기서 361자리의 PV에 대해서 흑이 살아남을 확률에 해당하는 PV를 PV+라 하고, 백이 살아남을 확률에 해당하는 PV를 PV-라 하면, PV = PV+ + PV- = 1이 성립하며, 임의의 국면 α에서 흑에 대한 형세, PMB(α)와 백에 대한 형세, PMW(α)을 식 (20)과 같이 표현할 수 있다.

$$PM_B(\alpha) = \sum_{i=1}^{19}\sum_{j=1}^{19} PV^{+}(\alpha, i, j)$$
$$PM_W(\alpha) = \sum_{i=1}^{19}\sum_{j=1}^{19} PV^{-}(\alpha, i, j) \qquad (20)$$
$PM_B(\alpha)$가 먼저 계산된다고 가정하면
$$PM_W(\alpha) = 361.0 - PM_B(\alpha)$$

또한 현재의 국면에 대한 IP의 합을 $IP_S(\alpha) = \sum_{i=1}^{19}\sum_{j=1}^{19} IP(\alpha, i, j)$ 로 나타내고, 현 국면에 대한 PM의 차는 PMD(α)= | PMB(α) - PMW(α) | 로 정의한다면 IP와 PM을 기반으로 하여 현재의 국면을 평가할 수 있다.

후보를 생성하기 위한 절차는 다음과 같다. 첫 번째

로 CGS가 흑이고, 적이 백이라고 가정할 때, PMS(α) 와 PMD(α) 에 따라서 매 국면마다 식 (21)과 같은 조건을 검사한다.

$$1\text{단계}\begin{cases} 0.0 < IP_S \leq 180.5 & : \text{귀를 점령} \\ 180.5 < IP_S & : \text{지역 전술 및 수상전} \end{cases} \quad (21)$$

두 번째로 식 (21)의 판단 결과가 지역 전술 및 수상전으로 판단되면 자살수가 아닌지를 점검한 후 식 (22)와 같이 네 가지 주요 전술중의 하나를 수행하게 된다.

$$2\text{단계}\begin{cases} \text{if } 0.0 < PM_D \leq 19.0 & : \text{적이 약하면} \rightarrow \text{공격} \\ & : \text{내가 약하면} \rightarrow \text{방어} \\ \text{else}\begin{cases} \text{if } PM_W > PM_B & : \text{침입} \\ \text{if } PM_B > PM_W & : \text{접근} \end{cases} \end{cases} \quad (22)$$

CGS의 기본 전략은 현재의 국면에 대해서 다섯 개의 후보를 생성하고 그 중에서 PV가 높은 지점을 최종 후보로 선정한다. 돌이 놓일 자리에 대해서 주위의 돌과의 영향력과 안정도를 계산하고, 덩어리의 사활을 적용한다. 적 혹은 나의 덩어리의 연결성을 체크하여 위급한 자리에 대해서는 IP를 재 계산한다. CGS에는 모두 47개의 루틴과 윈도우 화면 표시, 바둑 객체 표현, 현재 국면의 정보, 후보 생성, 진행 자료 처리 등의 모듈을 포함한다.

후보 자리에 대한 영향력은 8방향으로 이웃하는 돌이 있는가를 검사하여 인접한 빈자리가 자살수가 아닐 때만 갱신한다. 적을 방어하는 루틴은 나의 덩어리 중에서 안정도가 2이하인 덩어리들을 골라서 그 활로를 검색하고, 적을 공격하는 루틴은 적의 안정도가 3이하일 때 사용하며, 적의 덩어리사이에 침입하는 루틴 및 붙임 루틴은 해당 덩어리와 나의 돌의 연결성을 평가하여 무게가 높을 때만 사용한다.

Ⅳ. CGS의 모의 실험 및 평가

1. CGS의 구현에 사용한 도구

본 논문에서 제안한 CGS에서는 국면이 바뀔 때마다 IP와 PV, PM, 안정도를 계산하도록 되어 있다. 어떤 돌도 놓여지지 않은 빈자리에 대해서는 0의 IP값을 가지도록 하였다. 본 논문에서 정의한 IP의 범위는 그림 2와 같은 형태를 지니며 CGS의 한 모듈로 동작하도록 설계하였다.

0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	1	3	1	0	0	0	0
0	0	0	1	3	6	3	1	0	0	0
0	0	1	3	6	12	6	3	1	0	0
0	1	3	6	12	25	12	6	3	1	0
1	3	6	12	25	⓪	25	12	6	3	1
0	1	3	6	12	25	12	6	3	1	0
0	0	1	3	6	12	6	3	1	0	0
0	0	0	1	3	6	3	1	0	0	0
0	0	0	0	1	3	1	0	0	0	0

그림 2. 임의의 돌에서 인접하는 지점들에 대한 IP
Fig. 2. IP for adjacent points from a stone.

후보 생성 시스템은 IBM-PC 233Mhz 컴퓨터의 Windows 98 환경의 볼랜드 C++로 제작되었으며, OWL 2.0 라이브러리를 이용하여 바둑판의 메뉴와 돌의 모습, 국면의 정보를 나타내었으며, 각 자리의 PV에 따라서 현재 국면의 형세를 보여주기 위한 표식으로 범례와 같이 사용하였다.

■ : (1.00 ~ 0.85)
▪ : (0.85 ~ 0.70)
▪ : (0.70 ~ 0.55)
□ : (0.00 ~ 0.15)
▫ : (0.15 ~ 0.30)
▫ : (0.00 ~ 0.15)

(범례)

CGS의 후보 생성 과정을 검증하기 위해서 볼랜드 디버거를 이용했으며, 평가를 위해서는 별도의 PC에 대국용 프로그램을 설치하여 마치 두 사람이 대국을 진행하는 것과 똑같이 수행하고, 이때 한쪽이 기권하면 즉시 패배로 인정하도록 하였다. 대국용 프로그램으로는 모두 상품화된 것으로 [7]에서 개발한 NEMESIS와 [8]의 MFGO를 선정하였다.

2. 실험 및 평가

(1) CGS와 NEMESIS

NEMESIS는 Bruce Wilcox와 Walter Reitman이 LISP로 작성한 [7]의 프로그램을 PC상에서 1년여에 걸쳐 C 언어로 재 작성한 것으로, Wilcox는 여러 논문에서 정석 사례를 이용한 다양한 정보 표현 기법과 패턴 매칭을 통해서 최선의 수를 선택하기 위한 알고리즘을 제안하였다. NEMESIS는 1987년부터 1992년까지 세계 컴퓨터 바둑 대회에 참가하여 1989년의 준우승을 비롯한 우수한 성적을 거둔 프로그램이다.

본 논문에서 구현한 CGS와 NEMESIS의 대국은 CGS의 흑선(黑先)으로 시작되었다. 56수까지 진행된

그림 3을 보면, NEMESIS가 좌 하귀를 굳히자 CGS는 3귀를 차지하였다. CGS는 그림 4의 <흑 11>을 차지함으로서 좌변의 영역을 확보하기 시작하였으며, 우상 귀와 우하 귀를 튼튼히 다져나갔다.

반면에 NEMESIS는 <백 10>과 <백 16>으로 중앙 진출의 교두보를 마련함으로서 <백 50>, <백 52>, <백 54>의 수를 두어 <흑 37>, <흑 39>를 압박하고 있다. NEMESIS가 판단한 그림 3의 형세를 보면 흑의 Score는 68, 백의 Score는 62로 나타났는데 전반적으로 본 논문의 CGS가 NEMESIS보다 우세한 것으로 평가하고 있다.

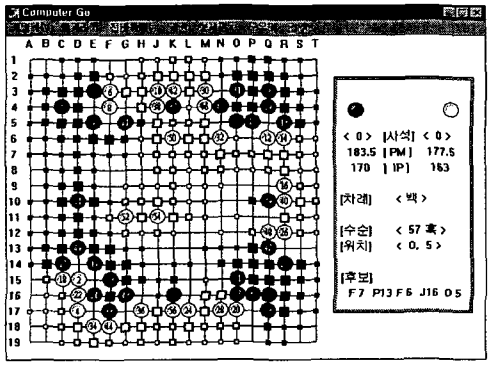

그림 3. CGS의 초반 형세 (57수까지 진행).
Fig. 3. The opening game of CGS(57th move).

<백 56>에 대한 CGS의 후보 생성 과정을 보면, 모두 현 국면의 PMS이 180.5보다 크고 PMD이 7이므로, 좌상의 흑에 대해서는 [F 6], [F 7]을, <백 56>의 차단에 대해서 <흑 55>를 방어하는 [J 16]을, 우상 흑의 눈을 만들기 위한 [O 5], 그리고 우하 흑과 <흑 39>를 연결하기 위한 [P 13]을 생성하였으며 이 중에서 CGS는 그림 3에 제시한 실험 결과와 같이 [O 5]를 선택하였다.

중반 단계에 이르면서 CGS는 <백 58>, <백 70>, <백 72>, <백 82> 등에 의해서 중앙 침입을 차단당한다. 이때 <흑 39>를 살리려는 <흑 75>, <흑 77>의 두 수는 불필요한 접근이었고, <백 94>, <백 98>에 의한 좌상의 흑 일대가 활로를 잃게 되었다. 이에 대해서 CGS는 우하 흑을 지키는 <흑 103>, <흑 105>로 응수하였으며 이때의 국면을 보면, 흑과 백의 PM에 있어서 CGS가 16.5집 정도 손실한 것으로 나타나 있다.

그림 4는 우하 귀의 마무리를 위한 NEMESIS의 대응 수를 보여준다. NEMESIS는 중앙 및 우변, 하변 일대를 차지하고 있으며 흑, 백의 Score는 각각 74, 89로 역전되었다.

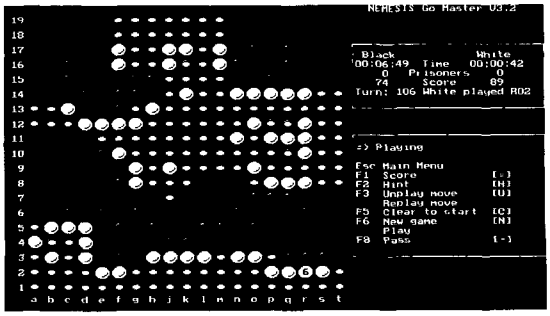

그림 4. CGS의 초반 형세 (57수까지 진행).
Fig. 4. The opening game of CGS(57th move).

종반 단계에 이르면서 NEMESIS는 좌상의 흑 무리를 위협하고 있다. <백 62>를 기점으로 <백 132>, <백 134>는 좌상의 흑 영역을 공격하는 침입으로 해석된다. CGS는 <흑 151>, <흑 153>, <흑 155>의 지역 접근수를 생성하여 <백 6>, <백 8>로 연결되는 백의 덩어리를 공격하려고 시도하다가 <백 156>에 의해서 좌상 흑 대마가 격리되었다. 그림 5의 종반 단계에 이르러서 CGS는 <백 180>의 끝내기 수에 대해서 더 이상의 후보를 생성하지 못하여 게임이 종료되었다.

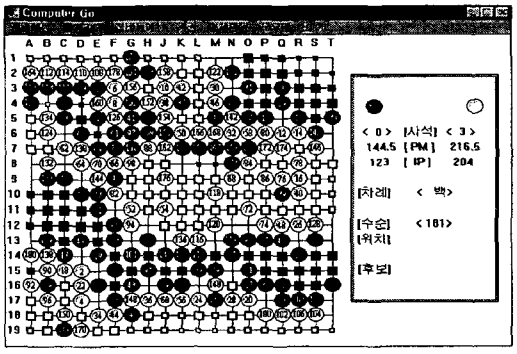

그림 5. CGS의 종반 형세 (181수 끝)
Fig. 5. The end game of CGS(181th move end).

(2) CGS와 MFGO

David Fotland는 MFGO(Many Faces of Go)에서 210개의 패턴을 기반으로 다음에 착점할 순서의 전체 트리를 계산할 수 있게 하였다. MFGO는 응창기 세계 컴퓨터 바둑 대회에 1987년부터 거의 매년 참여하여 1998년 우승을 비롯하여 항상 수위권에 입성하여 왔고, FOST컵 세계 대회에서도 좋은 성적을

거둔 바 있으며, 소프트웨어 시장에서도 많이 팔리는 우수한 컴퓨터 바둑 프로그램이다.

대국은 CGS의 선번으로 시작되었으며 총 대국 시간 은 35분, 대국 결과는 MFGO가 더 우세한 것으로 판정되었다. 그림 6은 53수까지 대국이 진행된 초반전 형세를 나타낸 것으로, 본 논문의 확률적 정보에 의하면 CGS의 PM은 190.8, MFGO의 PM은 162.2로 흑의 유리하다는 것을 보여주고 있다. 백의 52번째 수는 [D 9]로 좌상귀를 압박하는 수가 된다. 이에 대해서 흑은 [E 9], [R 10], [N 14], [M 13], [Q 9]의 다섯 개의 후보를 생성해 내었는데 이 중에서 흑은 [Q 9]로 응수하여 우변을 확보하였다.

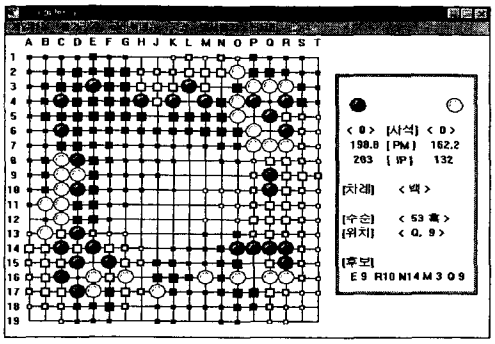

그림 6. CGS의 53수 국면 및 영역 표현
Fig. 6. 53th scene and territories representation of CGS.

사실 [E 9]도 매우 좋은 수로서 백은 54번째 수로 곧바로 [E 9]를 차지하였다. 이 때 우상귀의 흑 네 점은 활로가 차단되어 백과의 사활에 그 운명을 걸게 된 불안한 모습을 띠고 있다.

그림 7은 하변을 굳힌 MFGO가 자신의 영역을 계산하여 표현한 모습이다. 우상귀의 흑 네 점이 포획되어 백의 영역으로 되어 있음을 알 수 있고, 하변과 좌변 일대가 백의 세력으로 굳혀 있음을 알려주고 있다. 중반에 들어서면서 CGS는 좌 상변의 흑 무리를 굳히는데 진력을 쏟고 있으며 무리하게 우변 귀를 침투하였으나 <백 122> [S 16], <백 124> [T 16]에 의해 오히려 흑 다섯점이 차단당한 형세가 되었다.

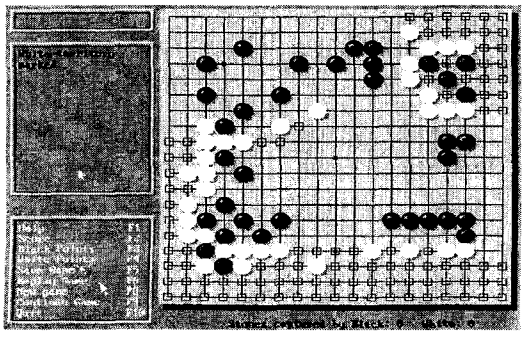

그림 7. MFGO의 69수 및 세력 표현
Fig. 7. 69th move and score of MFGO.

그림 8은 흑과 백이 끝내기에 들어간 모습이다. MFGO에 따르면 백이 50집 이겼다고 자체 평가하였으나, 수동(手動)으로 집 계산을 해 본 결과, 흑 48집, 백 66집으로 덤을 공제하지 않고 백이 18집 승리하였다.

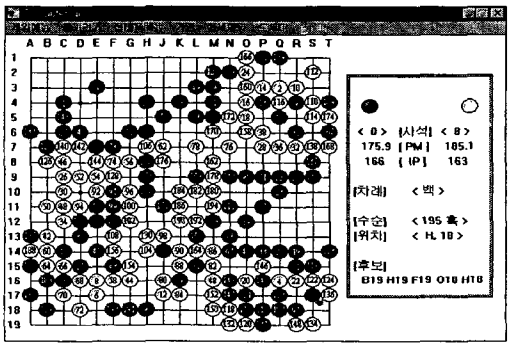

그림 8. CGS의 종반 형세
Fig. 8. The end game of CGS.

결론적으로 CGS는 귀에 집착을 많이 하는 실리를 챙겼고, MFGO는 바둑판의 중앙을 모두 차지함으로서 접근전에 강한 면모를 보여주었다.

V. 결 론

컴퓨터 바둑을 설계할 때 가장 힘든 문제중의 하나는 국면의 형세를 평가하기 위한 함수를 어떻게 개발할 것인가 하는 것이다. 바둑은 규칙의 간단함에 비해서 수 이동에 대한 변화가 많고 양자간에 점령해야 할 중립적인 지역을 두고 서로의 이해 대립이 불가피한 전면전의 성격을 가지기 때문에 국면의 변화에 따른 19×19 자리의 생사 여부를 분명히 할 필요가 있다.

본 논문에서는 바둑이 진행되는 임의의 국면에 대해서 확률 행렬을 이용한 후보를 생성하는 시스템인 CGS를 설계하고 구현하였다. CGS는 임의의 국면에 대해서 돌이 놓여 있지 않은 빈자리에 대해서 확률적인 생사 여부를 판단하여 현 국면에 적절한 후보를 결정하는 시스템이다. 돌이 놓여지면 CGS는 IP와 PV, PM 안정도의 계산에 근거하여 빈자리 중에서 PV가

큰 순서대로 다섯 개의 후보를 생성한다. CGS는 상대의 돌에 대해 직접 인접하여 착점하기를 꺼리고, 상대 세력이 강한 지역보다는 자신의 강한 세력을 확장하는데 역점을 둔다. 상대의 목표를 고려하지 않기 때문에 집의 확보에는 취약한 편이며, 간혹 축이나 패와 같은 전술을 필요로 하는 중반 이후의 상황에는 자신의 덩어리를 포기해 버리는 경향이 있다. 결론적으로 CGS는 초반에는 정석 사례를 사용하는 상용 프로그램과 비슷한 능력을 보이나 국면이 복잡해질수록 형세 평가가 정확하지 못하였다.

그러나 컴퓨터 바둑에서 후보 생성 알고리즘의 설계가 현실적으로 매우 힘든 지능 추론 문제중의 하나라는 점을 고려할 때, 정석 사례를 사용하는 NEMESIS에 비해서 사례를 전혀 구축하지 않은 CGS가 초반 단계에 있어서 다소 우세하게 영역을 확보했다는 사실은 향후 개발되는 컴퓨터 바둑의 주요한 개념으로 자리 매김할 수 있을 것이다. 앞으로의 작업은 제기된 몇가지 단점을 지속적으로 보완하여 후보 생성 시스템의 정확성을 높이는 일이다.

참고 문헌

[1] Ander Kierulf, Ken Chen, and Jurg Nievergelt, "Smart game board and Go explorer: A Study in Software and Knowledge Engineering," *Communication of the ACM*, Edmonton Canada, 1990.

[2] Stoutamire, D., 'Machine learning applied to Go,' *Master's thesis*, Case Wastern Reverse University, Manuscript available by Internet anonymous ftp from bsdserver. ucsf.edu, 1991.

[3] 곽민호, 김항준, "Situation Judgement in the *Game of Go Based on Influence*," Game Programming Workshop in Japan 95, 1995.

[4] 김영상, 유기영, "A Candidate Generation System Using Probability Matrix in Computer Go," IC-AI '99 *International Conference on Artificial Intelligence*(Las Vegas), Vol II, pp. 623-628, 1999.

[5] 김영상, "컴퓨터 바둑 프로그래밍 기법," *한국정보처리학회 논문지*, 제 3권, 제 3호, 1996

[6] 임중권, 김영상, 이종철, "연결정도에 따른 반면의 안정도 평가 기법," 한국정보과학회 봄 학술발표논문집, Vol. 20, No. 1, pp.193-196, 1994

[7] Walter Reitman and Bruce Wilcox, "The Structure and Performance of the INTERIM 2 Go Program", In *Proceedings of the 6th International Joint Conference on Artifical Intelligence*(Tokyo, August 20-23 '79), pp 711-719, 1979.

[8] D. Fotland, "Knowledge Representation in the Many Faces of Go", Manuscript available by *Internet anonymous ftp* from bsdserver.ucsf.edu, 1993.

저 자 소 개

金 永 祥(正會員)

1964년 7월 8일생. 1990년 2월 울산대학교 전자계산학과 졸업(공학사). 1990년~1991년 현대전자(주) 산업전자연구소에서 근무. 1993년 2월 경북대학교 컴퓨터공학과 졸업(공학석사). 1996년 2월 경북대학교 컴퓨터공학과 박사과정 수료. 1993년 3월~현재 제주한라대학 컴퓨터정보과에 재직. 주관심분야는 Graph theory, 게임이론, 정보보안

柳 基 永(正會員)

1976년 경북대학교 수학교육과 졸업(이학사). 1978년 한국과학기술원 전산학과 졸업(공학석사). 1992년 미국 Rensselaer Polytechnic Institute 졸업(이학박사). 1978년~현재 경북대학교 컴퓨터공학과에 재직. 주관심분야는 병렬처리, array processor 설계, 정보보안, 암호학