한국 CAD/CAM 학회 논문집
제 3 권 제 1 호 1998년 3월 pp. 57-67

학술논문

가중 선형 연상기억을 채용한 유전적 프로그래밍과 그 공학적 응용

연 윤 석*

Genetic Programming with Weighted Linear Associative Memories and its Application to Engineering Problems

Yun-Seog, Yeun*

ABSTRACT

Genetic programming (GP) is an extension of a genetic algoriths paradigm, deals with tree structures representing computer programs as individuals. In recent, there have been many research activities on applications of GP to various engineering problems including system identification, data mining, function approximation, and so forth. However, standard GP suffers from the lack of the estimation techniques for numerical parameters of the GP tree that is an essential element in treating various engineering applications involving real-valued function approximations. Unlike the other research activities, where nonlinear optimization methods are employed, I adopt the use of a weighted linear associative memory for estimation of these parameters under GP algorithm. This approach can significantly reduce computational cost while the reasonable accurate value for parameters can be obtained. Due to the fact that the GP algorithm is likely to fall into a local minimum, the GP algorithm often fails to generate the tree with the desired accuracy. This motivates to devise a group of additive genetic programming trees (GAGPT) which consists of a primary tree and a set of auxiliary trees. The output of the GAGPT is the summation of outputs of the primary tree and all auxiliary trees. The addition of auxiliary trees makes it possible to improve both the learning and generalization capability of the GAGPT, since the auxiliary tree evolves toward refining the quality of the GAGPT by optimizing its fitness function. The effectiveness of this approach is verified by applying the GAGPT to the estimation of the principal dimensions of bulk cargo ships and engine torque of the passenger car.

Key words : Genetic programming, Weighted linear associative memory, Primary tree, Auxiliary tree, Group of additive GP trees

1. 서 론

전형적인 함수 근사화(Function approximation) 방법은 학습집합(Learning set)을 이용하여 입력과 출력 관계를 적절히 매핑(Mapping)할 수 있는 함수를 찾는 것이라 할 수 있다. 가장 일반적인 방법이 기저함수(Basis function)를 다양한 방법으로 조합하는 것인데, Polynomial, Harmonic, Hyperbolic 등이 기저함수들로 사용될 수 있다[1]. 이때 SA(Simulated annealing), 유전적 알고리즘(Genetic algorithm), 그 외 다양한 최적화 기법을 사용하여 기저함수에 대한 계수의 값을 추정한다. 예를 들면, Feed-forward neural network은 Hyperbolic기반의 기저함수(Sigmoids)들의 결합체이며, 그 계수(Weight)는 오류역전파(Back propagation)[2]나 Conjugate gradient[3-5] 알고리즘을 이용하여 최적화된다.

그 반면에 함수 근사화의 도구로써 유전적 프로그래밍(Genetic programming, 이하 GP)[6, 7]은 앞에서 언

*정회원, 대진대학교 기계설계공학과

급한 방식과 구별된다. 유전적 프로그래밍은 유전적 알고리즘의 확장으로써 그 개체(Individual)가 트리(Tree) 형태의 컴퓨터 프로그램이 된다. 본 논문에서 컴퓨터 프로그램은 터미널 집합(Terminal set)과 함수 집합(Function set)의 조합으로 생성된 문법적으로 올바른 GP tree를 뜻한다. 진화과정(Evolving process)을 통하여 GP tree는 적합도(Fitness)를 최적화하기 위해서 그 구조 자체가 동적으로 변화하는데, 적합도 계산을 위해서 트리의 학습오차(Learning error)를 계산할 수 있는 함수가 사용된다. 기저함수 바탕의 근사화 기법은 그 함수의 형태가 이미 결정되어 있는 반면, 유전적 프로그래밍에서는 함수 즉 GP tree의 구조 자체가 적합도를 최적화하기 위하여 변화된다. 이러한 특징을 고려할 때, 유전적 프로그래밍은 함수 근사화의 유용한 도구로 활용될 가능성이 크다. 그러나 이것을 위해서는 다음과 같은 두 가지 근본적인 문제의 해결이 선행되어야 한다.

첫째, 유전적 프로그래밍에서는 다양한 공학적 응용이나, 함수 근사화에 필수적인 사항인 수치적 파라메터(Parameter) 혹은 가중치(Weight)를 추정하기 위한 방법이 결여되어 있다. 최근에 가중치 추정을 위한 많은 연구들이 진행되고 있는데, 이 모든 연구들은 비선형 최적화 기법에 의존하고 있다[8-14]. 이러한 접근 방법의 명백한 단점은 방대한 계산량을 요구한다는 점이다. 경우에 따라서는 한 개체군(Population) 내의 모든 개체에 대한 적합도를 계산하기 위해서 몇 대의 웍스테이션을 병렬로 사용해도 하루 이상의 시간이 소요될 수 있다. 따라서 본 연구에서는 비선형 최적화 기법의 사용을 지양하고, 가중 선형 연상기억(Weighted linear associative memory, 이하 WLAM)[15]의 사용을 모색하였다. 만일 연상행렬(Association matrix)의 크기가 커지면, 이 또한 많은 계산량이 요구되므로, 몇 개의 크기가 작은 연상 행렬들로 구성된 하나의 집합을 구성하여 사용하는 방안을 고안하였다. 이 연상 행렬은 GP tree의 전체 가중치 및 학습 집합의 일부만을 사용하여 구축된다. 이 접근 방식의 장점은 계산량의 대폭적인 감축과 아울러 비교적 정확한 가중치의 값을 산정할 수 있다는 점이다.

둘째, 유전적 프로그래밍의 알고리즘은 종종 지역최적점(Local minimum point)에서 벗어나지 못하기 때문에 만족할 만한 성능을 갖는GP tree의 생성이 어려운 경우가 있다. 유전적 알고리즘은 전역적 최적화의 기법으로 매우 많은 영역에서 활용되고 있는데, 지역최적점을 탈출하기 위하여 돌연변이 연산자(Mutation operator)가 사용된다. 그 반면 유전적 프로그래밍도 유사한 돌연변이 연산자가 존재하지만 일반적으로 유전적 알고리즘과는 달리 그 유용성이 크지 않다고 인식되고 있다[6, 7]. 이런 문제점을 해결하기 위해서, 본 연구에서는 Group of additive genetic programming trees(GAGPT)를 제시하였다. GAGPT는 하나의 주 트리(Primary tree)와 보 트리(Auxiliary tree)들의 집합으로 구성된다. 주 트리는 학습 집합을 사용하여 입력-출력 관계를 근사 시키는 일반적인 GP tree이다. 이 주 트리를 생성한 후 보 트리들이 첨가되는데, 보 트리는 원하는 출력과 주 트리 및 GAGPT에 이미 존재하는 보 트리들의 모든 출력을 합친 값에 기초한 적합도 함수를 최적화하는 방향으로 진화가 진행된다. 따라서 보 트리들의 첨가를 통하여 GAGPT에 대한 학습 성능(Learning capability) 및 일반화 성능(Generalization capability)의 향상을 꾀할 수 있다.

본 연구의 유용성을 검증하기 위하여 살물선(Bulk cargo ship)의 주요 치수(Principal dimensions) 및 승용차의 엔진 구동력(Torque) 추정에 GAGPT 접근법을 도입하고, 그 결과를 검토하였다.

2. 유전적 프로그래밍에 대한 소개

유전적 프로그래밍의 알고리즘은 초기에 Koza에 의해서LISP 언어로 구현되었기 때문에, 일반적으로 프로그램 코드들은 S-expression으로 표현된다[6]. 예를 들면, f=3.14×(sin(x)-abs(y)/z)를 S-expression으로 변환하면 (*3.14(-(sin x) (/(abs y) z))로 표현된다. 그리고 이것을 파스 트리(Parse tree) 혹은 GP tree로 표현한 것이 Fig. 1이다. GP tree의 노드(Node)는 사칙

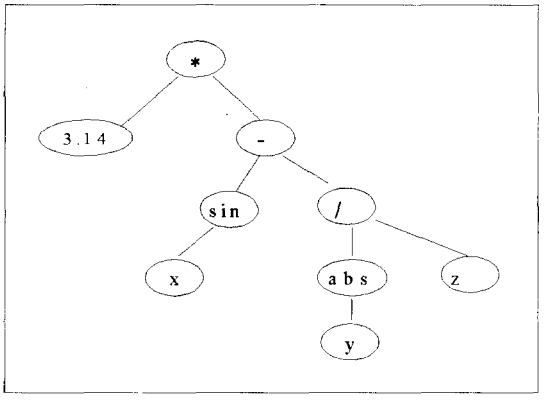


Fig. 1. An example of the GP tree.

연사자, 수학 함수 그리고 기타 프로그램 코드를 나타낸 함수(Func- tion) 노드와 변수 그리고 난수 발생을 나타낸 터미날(Terminal) 노드 등으로 구분된다. 3.14와 같이 고정된 상수는 변수를 나타낸 노드이되 항상 3.14를 그 값으로 갖는 터미날 노드로 표현하면 된다.

유전적 프로그래밍에서는 이러한 GP tree가 개체(Individual)에 해당되고, 이 개체가 모여서 개체군(Population)을 형성한다. 이때 개체군에 다음과 같은 유전적 연산자(Genetic operator)를 적용하여 적합도(Fitness)가 향상되도록 트리의 구조를 변화 시키게 된다.

1) 재생산(Reproduction)

재생산을 위하여 모(母) 트리(Parent tree)들이 적합도를 기준으로 선정되고, 이들은 다음 세대에 그대로 자(子) 트리(Offspring tree)가 된다.

2) 교배(Crossover)

선정된 모 트리들의 노드를 임의로 선정하여 그 하위 트리(Subtree)를 서로 바꾼다.

3) 돌연변이(Mutation)

트리의 노드를 임의로 선정하고 그 하위 트리를 무작위로 생성된 트리와 교환한다. 일반적으로 돌연변이 연산자는 대부분의 응용 영역에서 트리의 적합도를 크게 향상 시키지 못하는 것으로 알려져 있다[6].

유전적 프로그래밍은 종종 회귀 분석과 비교가 되는데, 회귀 분석은 식의 형태를 미리 가정하고 그에 따른 계수 값을 산정하는 반면, 유전적 프로그래밍은 GP tree의 구조를 직접 변화시키는, 즉 식의 형태 자체를 변화시킨다는 차이점이 있다. 그리고 유전적 프로그래밍에서는 앞의 예에서 보인 단순한 산술 식을 표현한 트리 뿐만 아니라, 프로그램 언어의 기본 요소인 조건분기, 루프(Loop) 등 응용 영역에 따라서 다양한 노드를 트리에 사용할 수 있다는 장점이 있다.

표준적인 유전적 프로그래밍의 문제점은 GP tree에서 정확한 수치 파라메터의 값을 결정하기가 어렵다는 점인데, 예를 들면 수학 함수들의 선형조합(Linear combination)에 대한 트리를 생성할 경우, 함수 노드와 난수(Random number)가 포함된 터미널 노드만을 사용하여 선형조합에 사용된 계수를 정확히 산출하기가 매우 어렵다. 그래서 이런 문제점을 해결하기 위한 대안으로 가중치를 도입하고, 이 값을 적절한 방법을 사용하여 추정해야 한다. Fig. 2는 가중치가 도입된 경우에 $f=a+b$를 GP tree와 S-expression으로 표현한 것이다.

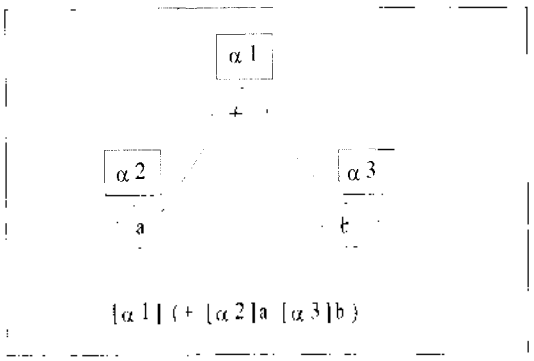


Fig. 2. An example of the GP tree with numerical weights and its S-expression.

가중치는 모든 트리 노드에 부과된 계수로써 트리 노드의 출력 값에 항상 이 값을 곱하여 전체 트리의 출력 값을 산정하게 된다. 일반적으로 GP tree의 가중치는 다양한 최적화 기법[8-14]들을 사용하여 추정하고 있다. GP tree는 단순한 수학 함수 만으로 구성되지 않고, 다양한 종류의 프로그램 코드가 사용될 수 있기 때문에 GP tree의 구배(Gradient) 정보를 사용한 최적화 기법은 적용되기가 어렵다. 본 논문에서는 비선형 최적화 기법의 도입에 따른 방대한 계산량을 회피하기 위하여 WLAM[15]를 사용하였다.

3. 가중 선형 연상기억

최근에 선형 연상기억(Linear associative memory)[16]을 시스템 파라메터 식별(System parameter identification)에 적용하려는 연구가 수행되었는데[17, 18], 만일 시스템 파라메터를 GP tree의 가중치와 대응하여 생각한다면 선형 연상기억은 GP tree의 가중치 산정에 충분히 사용될 수 있음을 짐작할 수 있다.

선형 연상기억을 이용하여 GP tree의 가중치를 산정하는 과정을 요약하면 다음과 같다.

학습집합 $L\{(X_1, y_1), (X_2, y_2), ..., (X_p, y_p)\}$과 GP tree의 노드에 대하여 서로 다른 m개의 가중치 벡터들의 집합 $W\{w_1, w_2, ..., w_m\}$가 주어질 경우, 이들을 사용하여 GP tree의 출력 벡터들의 집합 $z\{z_1, z_2, ..., z_m\}$를 구성할 수 있다. 여기서 X_i는 입력 벡터이고, y_i는 GP tree에 요구되는 출력 값이다. 또한 w_i와 z_i는 각각 n차원, p 차원의 벡터이다. z_i는 입력 벡터들의 집합 $X\{X_1, X_2, ..., X_p\}$에 대하여 가중치 벡터가 w_i일 때 GP tree의 출력 값들로 구성된 벡터이다. 이 가중치 벡터와 출력 벡터를 사용하여 행렬 $W(nxm)$와 Z

(*pxm*)를 구성할 수 있고, 식 (1)을 사용하여 연상기억 행렬(Association matrix) *M*을 구한다.

$$W = MZ \tag{1}$$

여기서 *M*(*nxp*)은 Singular value decomposition[19] 기법을 사용하여 구할 수 있다. 이때, 구하고자 하는 가중치 벡터 *w*는 식 (2)에 의하여 계산할 수 있다.

$$w^* = My \tag{2}$$

여기서 *y*는 학습집합 L에 포함된 요구되는 출력 벡터($y\{y_1, y_2, ..., y_p\}$)이다.

*w**와 *y*의 실제 관계는 비선형인데, 식 (2)에서는 이 관계를 근사적으로 선형화 한 것으로 볼 수 있다. 따라서 정확한 가중치의 산정이 어렵다고 판단할 수 있지만, 선형 연상기억을 사용하여 매우 정확한 시스템 파라메터를 계산하는 것이 가능하다는 연구 결과가 보고되어 있다[17, 18].

보다 정확한 가중치의 추정을 도모하기 위해서, 가중함수(Weight function)를 도입하여 *M*을 변경 시키고 이로부터 다시 가중치를 추정하는 방식이 있는데, 이러한 경우가 WLAM에 해당한다. 본 논문에서는 WLAM에 관련된 사항 중GP 알고리즘에 사용될 것만 간략히 기술하고자 하며, 자세한 내용은 참고문헌[15]을 참조하기 바란다. 우선, 초기 연상 행렬은 다음 식에 의하여 계산된다.

$$M = aWFZ^T\,[aZFZ^T + (1-a)I]^{-1} = aWFZ^TA \tag{3}$$

여기서, $A=[aZFZ^T+(1-a)I]^{-1}$, *a*는 Ridge parameter.

$$F = \left(1 \Big/ \sum_{k=1}^{m} f_k\right) \begin{bmatrix} f_1 & & \\ & f_2 & \\ & \cdots & f_m \end{bmatrix} \tag{4}$$

여기서, $f_i=f(|z_i-y|)$

*I*는 *pxp*단위 행렬이고, *f*는 그 값이 항상 양수인 단조감수함수(Monotonic decreasing function)인데, 다음과 같은 함수를 사용한다.

$$f(x) = 1/(D_o + x^2) \tag{5}$$

여기서, D_o는 상수

식 (3)-(5)를 사용하여 *M*(*nxp*)을 구축한 후, 식 (2)에 의하여 *w**를 계산한다. 이 추정치는 다음 식에 의하여 좀더 개선될 수 있다.

$$\hat{w}^* = w^* - M(z - y) \tag{6}$$

*z*는 입력이 *X*이고 가중치가 *w**일 때, GP 트리의 출력 값으로 구성된 벡터를 뜻한다. $\hat{w}^*$를 사용하여 새로운 가중치 벡터 *w*는 식 (7)을 사용하여 구하게 된다.

$$w = \hat{w}^* + (w^* - \hat{w}^*)\xi \tag{7}$$

여기서, $\xi = \begin{bmatrix} \xi_1 & & \\ & \xi_2 & \\ & \cdots & \xi_m \end{bmatrix}$

ξ_i는 -0.5에서 0.5 사이의 값을 갖는 난수이다. *M*은 다음과 같은 형태의 식에 의해서 개정된다.

$$M_{j+1} = \{M_j + \{a_{j+1}(1-\gamma_{j+1})h_{j+1}\}w_{j+1}z_{j+1}^TA_j\} \times [1 - z_{j+1}z_{j+1}^TA_j / \{1/\{a_{j+1}(1-\gamma_{j+1})\} + z_{j+1}^TA_jz_{j+1}\}] \tag{8}$$

$$A_{j+1} = h_{j+1}A_j[I - z_{j+1}z_{j+1}^TA_j / \{a_{j+1}(1-\gamma_{j+1})h_{j+1}\} + z_{j+1}^TA_jz_{j+1}\}] \tag{9}$$

$$a_{j+1} = a_j / [a_j + \gamma_{j+1}(1-a_j)] \tag{10}$$

$$\gamma_{j+1} = \sum_{l=1}^{j} f_l \Big/ \sum_{l=1}^{j+1} f_l \tag{11}$$

$$h_{j+1} = [a_j + \gamma_{j+1}(1-a_j)]/\gamma_{j+1} \tag{12}$$

a_i는 Ridge parameter로써, 초기 값은 보통 1 이하의 값을 사용하게 된다. GP tree의 가중치를 산정하는 과정을 요약하면 다음과 같다.

1) D_o및 Ridge parameter 초기 값 a_o를 결정 한다. *m*의 최소 값은 *n*+1이 되도록 한다.

2) 서로 다른 *m*개의 가중치 행렬 *W*(*nxm*)를 무작위로 생성하고, 이에 대응하는 GP tree의 출력 행렬 *Z*(*pxm*)를 입력 벡터들의 집합 *x*를 집합을 사용하여 구한다. 이로부터, 식 (3)을 사용하여 초기의 연상기억 행렬 *M*을 구축한다.

3) 식(2), (6), (7)을 사용하여 GP tree의 추정된 가중치 벡터 *w*를 계산한다.

4) *w*를 사용하여 GP tree의 새로운 출력 벡터 *z*를 구한다. *M*, *w* 그리고 *z*를 사용하여 식(8)-(12)로부터 새로운 연상기억 행렬 *M*을 계산한다.

5) 3)과 4)과정을 원하는 정도의 *w*가 구해 질 때까지 또는 주어진 최대 반복 횟수에 도달할 때까지 반복한다.

4. 가중 선형 연상기억을 채용한 유전적 프로그래밍

3절의 식(3)과 (8)를 살펴 보면, *M*(*nxp*)의 크기가 가중치 추정시 필요한 계산량에 가장 중요한 영향을

미친다. 만일GP tree의 노드 개수(n)가 많고, 학습집합의 양(p)이 많다면 M의 크기가 증가하고 이에 따라서 M을 구축하는데 필요한 계산량이 대폭 증가한다. 따라서 이러한 문제점을 해결하기 위한 방안으로 크기가 작은 몇 개의 M을 구축하여 크기가 큰 원래의 M대신에 사용하는 방법을 다음과 같이 모색하였다.

첫째, GP tree의 모든 가중치를 한꺼번에 처리하는 대신, 그 일부 만을 선정하여 값을 추정하고, 모든 가중치의 값이 추정될 때까지 이 과정을 반복 한다.

둘째, 전체 학습집합을 몇 개의 작은 집합으로 나누어 사용한다. 본 연구 수행의 경험에 의하면 보통 10-20개 정도의 원소들을 갖는 집합들로 구성하는 것이 적절하다고 판단된다.

위의 방안을 기초로 유전적 프로그래밍 알고리즘 하에서 WLAM을 이용한 가중치 산정 과정을 기술하면 다음과 같다.

1) 터미널 집합과 함수 집합을 정의하고, WLAM에 필요한 상수들(D_s, a_0)의 초기 값을 설정한다.

2) GP tree를 무작위로 생성하여 초기 개체군을 생성한다. GP tree의 모든 가중치는 1로 초기화 한다.

3) 개체군 내의 모든 GP tree에 대하여 다음 작업을 수행한다.

i. 입력 벡터들 $X\{X_1, X_2, ..., X_n\}$를 사용하여 GP tree의 출력을 계산하고, 이로부터 출력 벡터 z를 만든다. 이때, z와 실제 출력 벡터 사이의 학습오류(여기서는 Squared-error)를 계산한다.

ii. GP tree의 전체 가중치 중 그 일부 $\overline{w}$를 무작위 또는 정해진 방법에 따라서 선정하여, 가중치 벡터 $\overline{\omega}\{\overline{\omega}_1, \overline{\omega}_2, ..., \overline{\omega}_d\}$를 구성한다. 여기서 d는 n 보다 작거나 같아야 한다.

iii. 학습집합 L에서 일부의 데이터를 무작위 또는 정해진 방법에 따라서 추출하여, 크기가 작은 학습집합 $\overline{L}\{(\overline{X}_1, \overline{y}_1), (\overline{X}_2, \overline{y}_2), ..., (\overline{X}_g, \overline{y}_g)\}$을 구성한다. 여기서 g는 p보다 작거나 같아야 한다.

iv. $\overline{\omega}$에 난수를 더하는 방법을 사용하여, h개의 서로 다른 가중치 벡터를 구하고 이로부터 가중치 행렬 $\tilde{W}(dxh)$를 구성한다. $\tilde{W}$에 대응하는 GP tree의 출력 행렬 $\tilde{Z}(gxh)$를 입력 벡터 $\overline{X}\{\overline{X}_1, \overline{X}_2, ..., \overline{X}_g\}$를 사용하여 구한다. $\tilde{W}$와 $\tilde{Z}$를 사용하여, 식(3)으로부터 연상기억 행렬 $\tilde{M}(dxg)$를 구한다.

v. 식(2), (6) 그리고 (7)을 사용하여 추정된 가중치 벡터 를 계산한다. 이때, GP tree의 $\overline{W}$대신에 $\overline{\omega}$를 사용하여, i 과정에서와 같이 GP tree의 학습오류를 계산한다. 만일 계산된 학습오류가 전의 것 보다 작으면 $\overline{\omega}$를 ω'에 복사한다.

vi. $\overline{\omega}$에 대한 GP tree의 새로운 출력 벡터 $\tilde{z}$를 $\overline{X}$를 사용하여 구한다. $\tilde{M}$, $\overline{\omega}$ 그리고 $\tilde{z}$를 사용하여 식(8)로부터 새로운 연상기억 행렬 $\tilde{M}$을 계산한다.

vii. v-vi과정을 원하는 정밀도의 $\overline{\omega}$가 얻어지거나 최대 반복 횟수에 도달할 때까지 반복한다. ω'를 $\overline{\omega}$에 복사한다.

viii. iii-vii과정을 학습집합의 모든 데이터가 가중치 추정에 사용 때까지 반복한다.

ix. ii-viii과정을 모든 가중치가 추정될 때까지 반복한다.

4) 유전적 연산자들을 개체군에 적용하여 다음 세대의 개체군을 생성한다.

추정된 가중치는 지역최적 해일 가능성이 매우 크지만, GP tree 구조가 자체가 그 학습능력을 최적화하기 위해서 동적으로 변하기 때문에 추정된 가중치가 전역최적 해가 되어야 할 필요는 없다. 더욱이, 전역최적 해로 찾아진 가중치를 소유한 GP tree의 거동은 종종 Overfitting 현상을 보일 가능성이 높다. GP tree의 가중치 산정시 최적화 기법을 적용할 경우와 WLAM을 사용한 경우 차이점을 보이게 된다. 그러나 세대가 지나감에 따라서 그 성능에 있어서 큰 차이점을 보이지 않게 됨을 관찰할 수 있는데, 이것은 유전적 연산자들에 의해서 적합도 함수를 최적화하기 위해서 트리의 구조 자체가 변화되기 때문이라고 판단된다. 이때, 각기 다른 방법에 의거하여 추정된 가중치를 갖는 최적 트리들이 비록 동일한 성능을 보이는 경우라도 이들의 구조와 크기는 서로 차이가 있게 되는 것이 일반적인 현상이다.

5. Group of Additive Genetic Programming Trees(GAGPT)

유전적 프로그래밍 알고리즘이 지역최적점에 도달할 때, 이 지역을 탈출하지 못하는 경우를 자주 관찰 할 수 있는데, 이런 경우에 많은 세대가 지나더라도 좋은 성능을 갖는 GP tree가 생성되기 어렵다. 따라서, 본 논문에서는 이런 문제를 극복하기 위하여, GAGPT를 도입하였다.

GAGPT의 기본 개념은 주 트리(T_p)에 첨가될 보 트리(T_a)들을 도입한 것인데, 식 (13)에 기술되어 있다.

$$GAGPT = T_p + \sum_{i=1}^{q} \tilde{T}^i a \tag{13}$$

여기서, q는 보 트리의 개수이고, 는 $\overset{i}{Ta}$는 i 번째 보 트리를 의미한다.

즉 GAGPT의 출력은 주 트리와 모든 보 트리들의 출력을 누적함으로써 얻어진다. T_p와 $\overset{k}{Ta}$의 적합도 함수는 식 (14)과 (15)에 나타나 있다.

$$ft_p = \sum_{i=1}^{p} |y_i - z(X_i)|^2 \tag{14}$$

$$\overset{k}{fta} = \sum_{i=1}^{p} \left| y_i - z(X_i) - \sum_{l=1}^{k} \overset{l}{Za}(X_i) \right|^2 \tag{15}$$

여기서, $\overset{l}{Za}(X_i)$는 입력이 X_i일 때, l 번째 보 트리의 출력이다.

GAGPT를 구축하는 과정을 요약하면 다음과 같다.

1) ft_p를 적합도 함수로 사용하여 유전적 프로그래밍의 알고리즘을 통하여 최적의 주 트리 T_p를 먼저 생성한다.

2) $\overset{k}{fta}$를 적합도 함수로 사용하여 최적의 GP tree를 생성하는데, 이것이 보 트리 $\overset{k}{Ta}$가 된다.

3) 만일 GAGPT의 학습 오류를 좀더 줄이기 위해서는 또 다른 보 트리를 2)과정을 통하여 생성한다. 이 과정을 원하는 GAGPT의 성능이 얻어 질 때까지 또는 어떤 종료 조건이 만족될 때까지 반복한다.

다음절에서 알 수 있듯이 보 트리의 개수가 증가함에 따라서 GAGPT의 학습 성능은 꾸준히 개선되지만 보다 중요한 일반화 성능은 다른 경향을 보인다. 즉 어떤 특정한 보 트리의 숫자에서 최대의 일반화 성능을 보인 후, 또 다른 보 트리를 추가함에 따라서 일반화 성능은 개선되지 않거나 저하되는 현상을 관찰할 수 있다. 보 트리의 증가에 따른 GAGPT의 성능은 GP tree의 크기, 문제 영역의 특성에 크게 지배를 받는다. 그러나 무엇보다도 중요한 요인은 보 트리의 적합도 함수의 특징에 기인한다. 식 (16)에서 알 수 있듯이 보 트리의 적합도 함수는 주 트리와 GAGPT에 포함된 모든 보 트리 해의 합과 요구되는 출력 값과의 차이에 의존하고 있다.

$$\Delta_i = y_i - \{z(X_i) + \sum_{l=1}^{k-1} \overset{l}{Za}(X_i)\} \tag{16}$$

현재의 보 트리는 Δ_i를 기반으로 한 적합도 함수를 최적화하려는 방향으로 진화한다. 이때, Δ_i가 매우 복잡한 패턴을 보여 준다면 좋은 성능을 소유한 보 트리를 생성하기 어렵고, 따라서 보 트리의 추가에 따른 GAGPT의 성능 향상을 기대하기 어렵다.

6. Group of Additive Genetic Programming Trees의 응용

본 절에서는 살물선(Bulk cargo ship) 의 주요 치수와 승용차 엔진의 구동력 추정에 GAGPT를 적용하고 그 결과를 고찰하였다.

6.1 살물선의 주요 치수 추정

선박의 개념설계(Conceptual design) 단계에서는 선주의 요구 조건에 부응할 수 있는 선박의 길이(Length between perpendiculars), 폭(Breadth) 그리고 높이(Depth)등을 추정하게 된다. 이때 설계자는 재화중량(Dead weight)만을 고려한 회귀분석식과 과거의 실적선 자료를 토대로 설계를 진행하게 된다. 본 논절에서는 주요치수를 추정할 수 있는 GAGPT를 생성하고자 한다.

최근까지 선박설계에 관련된 지식베이스 시스템(Knowledge-based system) 개발의 사례들이 자주 발표되고 있으나[20, 21], 설계지식의 추출에 관련된 연구는 찾아보기 어려운 편이다. 이러한 지식베이스 시스템의 성능은 축적된 지식의 양과 질에 좌우되는데 공학문제에서 전문 지식의 추출이 매우 어렵다는 것은 주지의 사실이다. GAGPT는 설계지식을 자동 추출할 수 있는 도구로 사용될 수 있는 가능성이 있다. 즉 실적선 자료로부터, 주요 치수를 추정할 수 있는 GAGPT를 생성할 수 있는데, 이것은 신경망과 달리 수학식 형태로 전환하기 매우 유리하기 때문에, 추출된 설계지식으로 활용하기가 보다 용이하다.

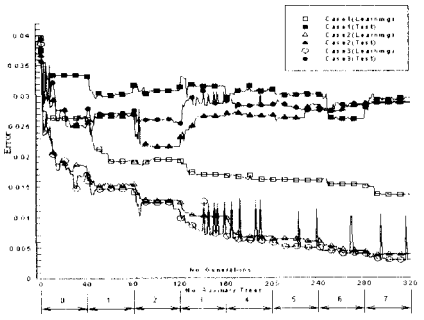
100척의 실적선 자료를 준비하였는데, 대부분 주요치수들은 선주의 요구 조건에 따라서 일정한 경향을 띄고 있지만, 선주의 특별한 요구 조건에 따라서 설계된 선박은 다른 특성을 보인다. 따라서 이것은 데이터에 마치 노이즈(Noise)가 존재할 때와 같은 효과를 발휘하기 때문에, 학습과 테스트 오차가 커지게 된다. 이 데이터를 무작위 방식으로 분할하여 절반은 학습집합으로 사용하였고, 나머지는 일반화 성능 즉 테스트 오차를 검토하기 위해서 사용하였다.

GP tree의 일반화 성능은 트리의 크기에 크게 좌우되기 때문에[22], 적절한 트리의 크기를 판정하기 위해서 몇 번의 실험을 실시하였다. 이 실험에서는 최대 허용 가능한 트리의 노드를 각각 세가지 다른 경우로 나누어, 어떤 경우가 GAGPT의 최적의 일반

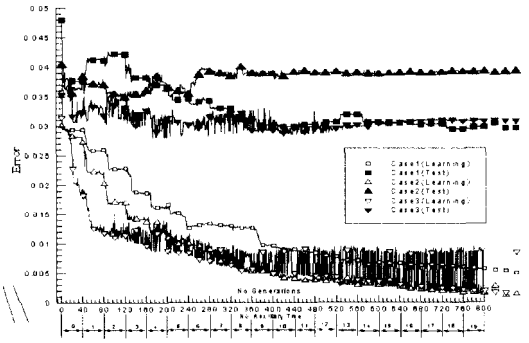
Table 1. Parameters used in GP algorithm for the principal dimensions of bulk cargo ships

population size	2000
initial depth of trees	2-3
allowable maximum nodes of trees	Case1 10 Case2 30 Case3 60
maximum generation for primary trees	40
maximum generation for auxiliary tree	40
selection method	tournament with 20 trees
reproduction probability	0.3
crossover probability	0.65
mutation probability	0.05
d (See section 4.)	15
g (See section 4.)	15
D_0 in equation (5)	0.001
a_0 in equation (10)	0.9
maximum allowable iteration number for recursively modifying matrix M(See equation (8).)	4

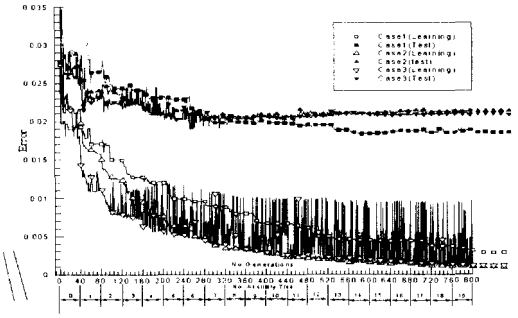
화 성능을 부여할 수 있는지 검토하였다(Table 1 참조). 유전적 프로그래밍 알고리즘에 사용된 파라메터들은 Table 1에 정리되어 있다. 사용된 터미날 집합은 $T\{DWT, C\}$이고, 함수 집합은 $F\{*, +, -, sqrt, \sin, \cos\}$이다. 여기서 DWT는 재화중량 이고 C는 난수이다. Fig. 3은 각각의 세대에서 찾아진 최적의 GAGPT에 대한 학습 및 테스트 오차를 보여 주고 있다. Fig. 3에 나타나 있듯이 40 세대가 지날 때마다 새로운 보 트리가 추가되고 있으며, Case 1에서는 최대 허용 가능한 트리의 노드 수를 10, Case 2에서는 30 그리고 Case 3에서는 60으로 제한하였다. 예상할 수 있듯이 Fig. 3에서 관찰되는 공통적인 특징은 보 트리의 추가에 따라서 GAGPT의 학습성능은 계속 개선되는 반면, 일반화의 성능은 계속해서 개선되기 보다는 다소 복잡한 특성을 보여 주고 있다. Fig. 3(a)에서 최적의 일반화 성능을 갖는 GAGPT는 Case 2로 보 트리의 개수는 2이다. 이때 더 이상 보 트리의 추가는 일반화 성능을 개선시키지 못하고 오히려 그 성능을 저하시키고 있다. Fig. 3(b)에서는 Case 3에서 보 트리의 개수가 4, 그리고 Fig. 3(c)에서는 Case 1에서 보 트리의 개수가 14일 때 최적의 일반화 성능을 보여 주고 있다. Table 2는 최적의 일반화 성능을 갖는 GAGPT의 학습 및 일반화 성능을 간략히 정리한 것이다. 이상의 관찰 결과에서 보 트리의 추가는 GAGPT의 학습과 일반화 성능을 개선



(a) Error landscapes of the GAGPT for the length between perpendiculars L_{pp} of bulk cargo



(b) Error landscapes of the GAGPT for breadth B of bulk cargo ships.



(C) Error landscapes of the GAGPT for depth D of bulk cargo ships.

Fig. 3. Error landscapes of the GAGPT for the principal dimensions of bulk cargo ships. Here, the measure of error is $\frac{1}{N}\sum_{i=1}^{N}\left|\frac{y_i - z_i}{y_i}\right|$, where y_i and z_i are the actual and estimated output, and N is the number of learning samples.

시킬 수 있음을 확인 할 수 있다. 그러나 보 트리의 개수가 증가한다고 하여 항상 GAGPT의 일반화 성능을 개선 시킬 수 있는 것이 아니다. 즉 최적의 일반화 성능을 줄 수 있는 보 트리의 개수가 존재하며 그 이상으로 보 트리를 추가하여도 일반화 성능이

Table 2. The results of GAGPTs that are found as the best ones in the perspective of the generalization capability

Principal Dimensions	The Primary Tree in the GAGPT		The GAGPT with Auxiliary Trees		
	Learning Error	Test Error	#Auxiliary Trees	Learning Error	Test Error
L_{hp}(Case2)	0.017539	0.024897	2	0.012836	0.021502
B(Case3)	0.018843	0.032048	4	0.011786	0.028071
D(Case1)	0.019775	0.024044	14	0.004481	0.018340

향상되기 어렵다.

6.2 승용차 엔진의 구동력 추정

자동차 엔진의 특성 파악은 자동변속기, 서스펜션(Suspension) 그리고 동력 전달계를 설계하는데 있어서 필수적인 사항이므로, 정확히 엔진의 거동을 기술할 수 있는 엔진 모델이 필요하다. 그런데, 최근에 전자화된 제어 장치에 의하여 연료가 분사되는 것이 일반적이므로, 간략하고 유용한 엔진 모델을 개발하는 것이 용이하지 않게 되었다. 엔진 모델을 개발하는 대신에, 트로틀 개도(Throttle opening)와 엔진의 회전속도에 따른 엔진 구동력을 실험적으로 측정하고, 이 실험 자료를 활용하는 방법을 고려할 수 있다[23]. 이 측정된 엔진 정특성 자료는 Fig. 4에 나타나 있다.

본 절에서는 이 자료를 사용하여 엔진의 구동력을 추정할 수 있는 GAGPT를 구축하고자 한다. 그런데 이용할 수 있는 자료의 양이 많지 않기 때문에, 모든 자료는 학습 자료로 이용되었다.

보 트리를 포함하고 있지 않은 GAGPT의 학습 오차는 평균3.4489% 이고, 두개의 보 트리를 사용함으로써 개선된 학습 오차는 2.7551%이다. 여기서 사용된 오차는 6.1절에 사용된 상대 오차이다. Fig. 5는 GAGPT의 학습 결과를 가시화한 것이다. 더 많은 보 트리를 추가하거나 GP tree의 크기를 증가 시키면 보다 개선된 학습 오차를 얻을 수 있지만, GAGPT 전체의 크기가 증가하여, 이로부터 비교적 간단한 수학식을 추출할 수 없다는 단점이 있다. GAGPT에 포함된 3개의 GP tree들을 수학식 형태로 전환 시키고, 이를 Mathmatica를 사용하여 간결히 정리하면 식(17)로 표현된다. Fig. 6(a)는 보 트리가 포함되지 않은 즉 주 트리에 대한 해를 가시화한 것이고, Fig. 6(b)는 두 개의 보 트리를 소유한 GAGPT의 해를 가시화한 것이다. 테스트 데이타가 없기 때문에 식의 일반화 성능을 계측할 수 없지만, 문헌[23]에서는 실험 데이타를 스플라인(Spline)과 선형보간법을 결합하여 사용한다는 사실을 고려한다면, Fig. 6(b)에 나타나 있듯이 식 (17)의 출력 면(Surface)이 연속적(Continuous)이고 유연(Smooth)하기 때문에, 이 식은 충분히 실용적으로 사용될 수 있음을 판단할 수 있다.

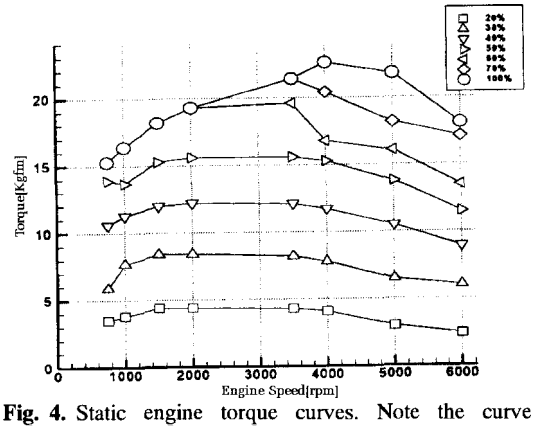


Fig. 4. Static engine torque curves. Note the curve represents torque at the same rate of throttle opening with respect to engine speed. For instance, the lower most curve is at 20% throttle opening. All curves are plotted by simply drawing a line between adjacent two data points.

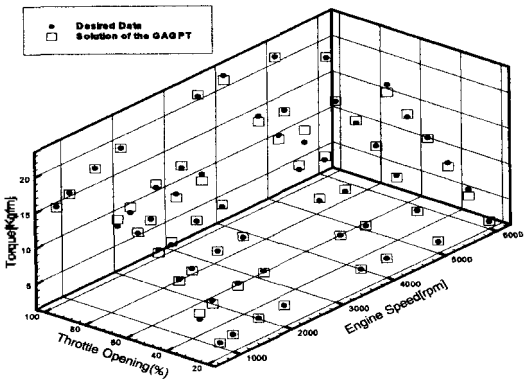
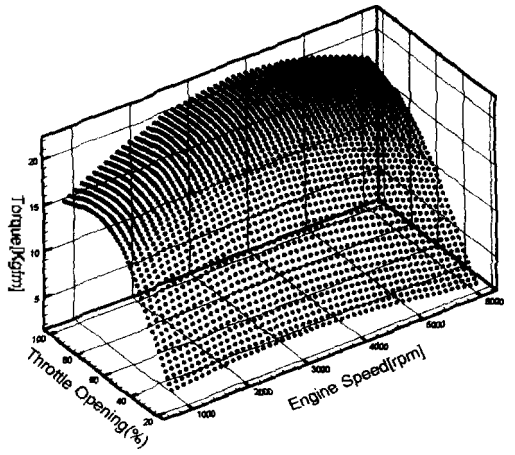


Fig. 5. Learning results of the GAGPT with two auxiliary trees for the estimation of engine torque.

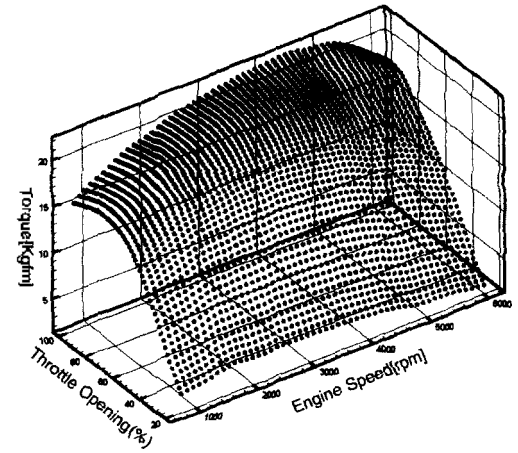
$$Torque = T_p + \overset{1}{Ta} + \overset{2}{Ta} \tag{17}$$

여기서,

$$T_p = 21.51261[e^{-0.60543(-1.11086+0.1066287\times 10^{-1}x_2)^4} - 0.34675\{-e^{(-0.84556e^{-0.519453\times 10^{-4}x_2^2})}$$



(a) The solution space of the GAGPT without an auxiliary tree



(b) The solution space of the GAGPT with two auxiliary trees

Fig. 6. The solution space of the GAGPT for the estimation of engine torque.

$$+0.160503_{x1}\}^{-2}\}^{4}$$

$$Ta^{1} = 1.59479\{\sin\{0.84979(0.107514\times10^{-3}{}_{x1} - 0.6337(0.66758\times10^{-3}{}_{x1} - 0.724706\times10^{-2}{}_{x2})^{4} + 0.97509\times10^{-4}{}_{x2})^{5}\}^{5},$$

$$Ta^{2} = 0.48086\{-0.6437Se^{-0.933026\times10^{-12}{}_{x1}{}^{4}} + 0.42955\{-0.13454\times10^{-17}{}_{x1}{}^{4} + 1.03246 \sin(0.97262\times10^{-3}{}_{x1})\}^{5}\}^{5}$$

x_1는 엔진의 속도(rpm)이고 x_2는 트로틀 개도(%)이다.

7. 결 론

본 논문에서는 *GP tree*의 가중치 산정을 위해 WLAM의 사용을 제안했고, GP tree의 성능을 향상시키기 위한 방안으로 GAGPT를 제시하였다. WLAM은 가중치 추정에 효율적으로 활용될 수 있지만, 연상기억 행렬의 크기가 증가함에 따라 계산량이 크게 증가한다는 단점이 있다. 그래서, 작은 크기의 연상행렬들의 집합을 구축하여, 이것을 가중치 추정에 활용함으로써 계산량을 감축의 효과를 도모하였다. 유전적 프로그래밍의 알고리즘은 지역최적점에 도달하여 탈출할 수 없는 가능성이 항상 존재하기 때문에, 생성된 GP tree가 최적의 성능을 갖는다고 보장할 수 없다. 이 문제를 해결하기 위해 제시된 GAGPT는 주 트리와 보 트리들의 집합으로 구성되며, 그 출력은 주 트리와 모든 보 트리의 출력을 누적하여 산출하게 된다. 보 트리는 GAGPT의 성능을 향상 시키는 방향으로 진화가 되기 때문에 GAGPT의 학습 및 일반화 성능을 개선시킬 수 있다.

본 논문에서 제시된 접근 방식의 유용성을 검증하기 위해서 살물선의 주요 치수 및 승용차의 구동력을 추정할 수 있는 GAGPT를 구축하였다. 이 결과로부터 보 트리의 증가는 학습 능력을 계속 향상시킬 수 있는 반면, 일반화 능력은 보 트리의 수가 특정한 수에 도달하면, 보 트리가 증가하더라도 더 이상 일반화 능력은 향상되지 않는 것을 관찰할 수 있었다. 또한, 최적의 보 트리의 수나 주 트리와 보 트리의 크기를 결정할 수 있는 일반적인 경향은 관찰되지 않았고, 이것은 주로 다루고 있는 문제의 특성에 의존되어 있는 것으로 판단되어 진다.

엔진 구동력을 추정할 수 있는 주 트리로부터 비교적 간단한 수학식을 도출함과 아울러, 보 트리로부터 얻어진 두 개의 수학식을 함께 사용함으로써 구동력 추정 시 그 정확성을 향상 시켰다. 인공 신경망과 비교하여 볼 때, 이러한 수학식들의 용이한 도출은 GAGPT를 사용함으로써 획득되는 장점 중에 하나라고 할 수 있다.

일반적으로 두개의 GP tree가 동일한 학습성능을 보여도 그 예측성능 즉 일반화성능에 차이를 보이게 되는데, 이견이 있지만 대체적으로 크기가 작은 트리가 보다 우수한 일반화 성능을 보인다고 여겨지고 있다(Occam's Razor)[23, 24]. 이와 관련하여 볼 때, 보 트리가 증가에 따라서 전체적으로 GAGPT의 크기도 증가하게 되고, 6절에 보였듯이 일반화 성능이 계속해서 개선되지는 않는다. 즉 Overfitting 현상을 보이게 되는데, 이 현상을 줄일 수 있는 방안이 강구되어야 한다. 현재, 선형보간법이나 *Lazy learning* 기법을 이용하여 Overfitting을 줄이기 위한 연구를 수행하

고 있다.

본 논문의 가장 큰 의의는 함수의 근사적 추정에 흔히 사용되는 회귀분석이나 신경망 등 이외에 새로운 방법으로 Symbolic regression의 성격을 갖는 GAGPT를 제시했다는 것이며, GP tree의 가중치 산정에 최적화 기법의 사용이 아닌 시스템 파라메터 추정의 관점에서 WLAM을 채용하여 계산시간을 대폭 단축했다는 점이다.

감사의 글

본 논문이 완성되기까지 많은 조언을 해 주신 서울대학교의 양영순 교수님과 본 논문에 사용된 데이타를 제공해 주신 기계연구원의 이경호 선임연구원 그리고 대진대학교의 정규홍 교수님께 깊은 감사를 드린다. 아울러, 본 논문을 세심히 검토해주신 익명의 심사위원들께도 깊은 사의를 표한다.

참고문헌

1. Beyer, U. and Smieja, F., "Learning from Examples, Agent Teams and the Concept of Reflection", *Int. J. of Pattern Recognition and AI*, Vol. 10, No. 3, pp. 251-272, 1996.
2. McClelland, J. and Rumelhart, D., *Parallel Distributed Processing*, Vol. 1, 2, MIT Press, Cambridge, MA, 1986.
3. Adeli, H. and Hung, S.L., *Machine Learning-Neural Networks, Genetic Algorithms, and Fuzzy Systems*, John Wiley and Sons, New York, 1995.
4. Hung, S.L. and Adeli, H., "An Adaptive Conjugate Gradient Learning Algorithm for Effective Training of Multilayer Neural networks", *Applied Mathematics and Computation*, Vol. 62, No. 1, 1994.
5. 양영순 외, 지능형 선상 가열 시스템 개발, 서울대학교 조선해양공학과, 기술보고서, 1996.
6. Koza, J.R., *Genetic Programming : On the Programming of Computers by Means of Natural Selection*, The MIT Press, 1992.
7. Koza, J.R., *Genetic Programming II: Automatic Discovery of Reusable Programs, The MIT Press*, 1994.
8. Sharman, K.C., Esparcia-Alcazar, A.I. and Li, Y. "Evolving Signal Processing Algorithms by Genetic Programming", in *Proc. First IEE/IEEE Int. Conf. on GA in Eng. Syst.: Innovations and Appl.*, Sheffield, Sept. pp. 473-480, 1995.
9. Gray, G.J., Li, Y., Murray-Smith, D.J. and Sharman, K.C., "Structural System Identification using Genetic Programming and a Block Diagram Oriented Simulation Tool", *Electronics Letters*, Vol. 32, pp. 1422-1424, 1996.
10. Bettenhausen, K.D., Marenbach, P., Freyer, S., Rettenmaier, H. and Nieken, U., "Self-Organizing Structured Modeling of a Biotechnological Fed-batch Fermentation by Means of Genetic Programming", in *Proc. of the IEE Conf. GALESIA*, Sheffield, UK, No 414, pp. 481-486, 1995.
11. McKay, B., Willis, M.J. Hiden, H.G. Montague, G. A. and Barton, G.W., "Identification of Industrial processes using genetic programming", *Identification in Engineering Systems* Vol. 1. pp. 510-519, Swansea, UK, 1996.
12. Weinbrenner, T., *Genetic Programming Techniques Applied to Measurement Data*, Diploma Thesis, Dept. of Electronics and Electrical Engineering Centre for Systems and Control, University of Glasgow, 1997.
13. 양영순, 연윤석, "유전적 프로그래밍을 이용한 중앙단면 설계지식의 추출", 대한조선학회 '96 추계학술회의 논문집, pp. 86-89, 1996.
14. 이경호, 연윤석, "다중 인공신경망과 유전적 프로그래밍의 복합적 접근에 의한 공학설계 시스템의 개발", 한국전문가시스템학회 추계학술대회 논문집, pp. 396-405, 1997.
15. Lin, J.C. and Durand, D.M., "Weight Linear Associative Memory Approach to Nonlinear Parameter Estimation", *J. of Optimization Theory and Applications*, Vol. 90, No. 1, pp. 139-159, 1996.
16. Kohonen, T., *Self-Organization and Associative Memory*, Springer-Verlag, New York, 1988.
17. Kalaba, R.E., Lichtenstein, Z., Simchony, T. and Tesfatsion, L., "Linear and Nonlinear Associative Memories for Parameter Estimation", *Information Sciences*, Vol. 61, pp. 45-66, 1992.
18. Kalaba, R.E. and Udwadia, F.E., "Associative Memory Approach to the Identification of Structural and Mechanical Systems", *J. of Optimization Theory and Applications*, Vol. 76, No. 2, pp. 207-223, 1993.
19. Gruber, M.H.J., *Regression Estimator: A Comparative Study*, Academic Press, New York, 1990.
20. Yeun, Y.S. and Yang, Y.S., "Design Knowledge Representation and Control for the Structural Design of Ships", *Knowledge-Based Systems*, Vol. 10, No. 2, pp. 121-133, 1997.
21. Lee, K.H., Lee, D.K. and Han, S.H., "Object-oriented Approach to a Knowledge-based Structural Design System", *Expert Systems with Applications*, Vol. 10, No. 2, pp. 223-231, 1996.
22. Zhang, B.T. and Muehlenbein, H., "Balancing Accuracy and Parsimony in Genetic Programming",

Evolutionary Computation, Vol. 3, No. 1, pp. 17-38, 1995.

23. Lee, K.Y., Cho, B.H, Huh, J.W. and Jung, G.H., *Development of the Hydraulic Control System for Automatic Transmission using Proportional Solenoid Valve*, Technical Report, Institute of Precision Machinery Design, Seoul National University, 1997.
24. Webb, G.I., "Further Experimental Evidence against the Utility of Occam's Razor", *J. of Artificial Intelligence Research*, Vol. 4, pp. 397-417, 1996.

연 윤 석

1989년 서울대학교 조선해양공학과 학사
1991년 서울대학교 조선해양공학과 석사
1995년 서울대학교 조선해양공학과 박사
1993년 대진대학교 기계설계공학과 전임강사
1995년 ~ 현재 대진대학교 기계설계공학과 조교수
관심분야 : Integrated Computer-Aided Engineering Artificial Intelligence in Engineering Design, Inductive program Discovery, Evolutionary Computations for Engineering Design, Multi-Agent Systems