經營科學
第17卷 第2號
2000年 11月

진화 알고리듬을 위한 객체지향 모델링과 클래스 라이브러리 구현*

정호연** · 이수연*** · 곽재승**** · 김용주***** · 박기태****** · 현철주*******

Object-Oriented Modeling and Implementation of a Class Library for Evolutionary Algorithms*

Ho-Yeon Chung** · Soo-Yeon Lee*** · Jae-Seung Kwak**** ·
Yong-Ju Kim***** · Ki-Tae Park****** · Chul-Ju Hyun*******

■ Abstract ■

In evolutionary algorithm, there exist various models for the evolution of the population with respect to schemes and strategies for reproduction. In the application of the algorithm to a specific problem, one model suitable to the problem is to be properly chosen and a program expert or a software is needed to help implement and test a designed algorithm. In this study, object oriented modeling and the class library for simple evolutionary algorithms(SEA) with one population is developed. The library proposed here can be used as a generalized tool for solving problems in a wide range of domains.

1. 서 론

진화알고리듬은 일종의 인공지능기법으로 유전알고리듬(genetic algorithm : GA), 진화전략(evolution strategy : ES), 진화프로그래밍(evolutionary programming : EP) 등이 이에 속한다. 진화알고리듬은 하나의 種(species)이 고정된 환경에 적응하여 진화하는 모형과 여러 종들이 동시에 상호작용

* 본 연구는 한국과학재단 특정기초 연구(과제번호 : 98-0200-09-01-3)지원으로 수행되었음.
** 전주대학교 산업공학과 부교수
*** 산업기술시험원
**** 연구개발정보센터
***** (주)도울정보기술, 동신대학교 산업공학과 교수
****** 전남대학교 대학원 산업공학과
******* 정인대학 품질관리과

하고 상호적응하면서 공진화(coevolution)하는 모형으로 크게 분류할 수 있다. 본 연구에서는 전자를 단순진화 알고리듬이라 부르기로 하고, 후자를 공진화 알고리듬이라 부르기로 한다.

특정 문제에 대해 진화알고리듬을 적용하기 위해서는 모집단의 운영방법이나 진화방식에 대한 선별방법, 유전연산자 등 다양한 요소별 방법과 전략 가운데 적용문제에 적당한 방법을 선택하거나 개발하여, 이를 구현해야 한다. 그러나 이러한 과정은 상당히 복잡하여 진화알고리듬과 프로그래밍에 익숙하지 않은 의사결정자가 진화알고리듬을 의사결정의 기본도구로써 사용하기에는 많은 어려움이 따른다. 따라서 진화알고리듬을 보다 쉽게 활용하기 위해서는 사용자의 편의성, 다양한 응용프로그램으로의 이식성 및 강건성을 갖춘 라이브러리의 구축이 필요하다.

지금까지 개발된 진화알고리듬에 관한 라이브러리는 GAGS[13], GAME[7], GAlib[12], libga[5], LEAP[11] 등이 있다. 그러나 이들 라이브러리는 모두 유전알고리듬에 대한 라이브러리이며, Galib [12]를 제외한 나머지는 모두 단순 유전알고리듬(simple GA) 또는 안정상태 유전알고리듬(steady-state GA)만을 지원한다. 또한 유전알고리듬을 구성하는 각 요소별로 다양한 방법이나 전략이 개발되어 있으나, 이들 라이브러리들은 일부 대표적인 방법만을 제공하고 있다. 한편 국내의 경우 진화알고리듬에 관한 연구가 활발히 이루어 지고 있으나, EAF[3]를 제외하고는 다양한 문제에 적용할 수 있는 진화알고리듬을 위한 소프트웨어나 라이브러리의 개발이 이루어지지 않고 있다.

EAF는 단순진화 알고리듬에 대한 라이브러리와 함께 사용자가 작성하고자 하는 응용프로그램의 골격코드를 생성해 주는 소프트웨어로 단순진화 알고리듬에 적합하게 개발되었다. 본 연구에서는 개발된 EAF를 바탕으로 다양한 문제에 적용할 수 있고 각 요소별로 기존에 개발된 다양한 방법이나 전략을 포함할 수 있도록 진화알고리듬에 대한 모델링을 수행한다. 그리고 이를 기반으로 새로 개발된 전략이나 기법의 추가와 공진화 알고리듬으로의 확장이 보다 용이한 단순진화 알고리듬을 위한 라이브러리를 개발하고자 한다. 이때 코드의 재사용성과 확장성을 위해 객체지향 개발 방법론(object-oriented software development methodology)을 사용한다. 이를 위하여 본 연구에서는UML (unified modeling language)을 사용하여 객체지향 기법의 일반적인 절차에 따라 사용자 요구사항 파악 및 객체지향 분석, 객체지향 설계, 객체지향 프로그래밍의 반복작업을 통해 진화알고리듬에 대한 모델링과 함께 클래스 라이브러리를 개발한다.

본 연구의 구성은 다음과 같다. 제2장에서는 진화알고리듬에 대한 객체지향 모델링을 다루고, 제3장에서는 클래스 라이브러리의 설계 및 구현방법을 제시한다. 제4장에서는 개발된 클래스 라이브러리의 적용예제를 통해 라이브러리의 유용성을 보인다. 제5장은 결론으로 구성되어 있다.

2. 단순진화 알고리듬에 대한 객체지향 모델링

객체지향 개발 방법론은 전반적인 영역의 정보체계에 대해 데이터 중심으로 주어진 문제를 모델링하는 방법, 즉 객체지향 분석 및 설계 방법을 적용하여 이를 프로그램으로 구현하는 일련의 기술들을 말한다. 객체지향 방법론은 다양한 방법들이 제시되어 왔으며, 이러한 방법론은 대체적으로 모델링 언어와 프로세스로 구성되는데, 모델링 언어는 그래픽을 위주로 한 표기법을 의미하고, 프로세스는 모델링 언어를 사용하여 분석, 설계하는 과정을 의미한다[15]. 본 연구에서는 모델링 언어로 객체지향 기술의 권위자인 Grady Booch, James Rumbaugh, Ivar Jacobson 등이 자신들이 제시한 방법(booch방법론, OMT, OOSE)과 기타 다른 전문가들이 제안한 방법을 통합하여 만든 UML(unified modeling language)을 사용한다. 이를 사용하여 객체지향 기법의 일반적인 절차를 가장 잘 적용시킨 OMT의 절차에 따라 라이브러리를 구현하고자

한다. OMT의 기본 절차는 크게 다음 4가지로 구성된다[15].

(1) 사용자 요구사항 파악

사용자 요구사항 파악은 시스템이 만족시켜야 할 요구사항을 찾고 이를 명세화 하는 과정으로, 시스템이 '무엇'을 수행할 것인가를 정의하는 것이다.

(2) 객체지향 분석(OOA : Object Oriented Analysis)

분석단계는 문제를 정의하고 정의로 부터 모형들을 제작하여 실세계의 중요한 특성들을 보여주는 단계이다. 일반적으로 이 단계에서 객체모형(object model), 동적모형(dynamic model), 기능모형(function model)을 정의한다.

(3) 객체지향 설계(OOD : Object Oriented Design)

객체지향 설계에서는 시스템을 단순히 데이터를 처리하는 절차들의 집합으로 간주하기보다는 하나의 객체로 부터 다른 객체까지 전달되는 메시지를 갖는 객체들의 집합으로 간주한다. 정보은폐, 추상화 등을 기초개념으로 하는 객체지향 설계는 프로그램 단위를 객체로 간주하고 그들 사이의 관계를 설정하는 과정이다.

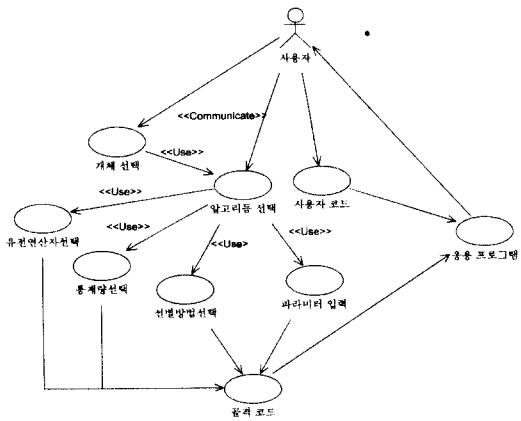
(4) 객체지향 프로그래밍(OOP : Object Oriented Programming)

객체지향 프로그래밍의 기본개념으로 객체, 클래스, 메시지, 메소드 등이 있다. 기존의 절차적 프로그래밍에서는 호출자가 데이터와 절차의 모든 제어를 가지고 절차를 호출하는데 비해 객체지향 프로그래밍에서는 동작 수행 자체가 메시지를 받은 객체에서 이루어지기 때문에 자료 추상화라는 중요한 특성을 지니게 된다.

2.1 사용자 요구사항 파악

UML에서는 사용자 요구사항을 파악하기 위해 유스케이스 다이어그램을 작성하는데, 이는 사용자와 시스템 간의 전형적인 상호작용을 표현한 것이다. 즉, 유스케이스 다이어그램은 사용자의 관점에서 모델링하는 것으로, 사용자의 시점을 정확히 이해함으로써 사용자가 간편하게 사용할 수 있으면서 유용한 시스템의 설계가 가능하도록 해준다. 유스케이스 다이어그램은 액터와 유스케이스로 구성되는데, 액터는 시스템과 상호작용하는 사람이나 사물을 나타내고, 유스케이스란 시스템과 사용자 간의 상호작용을 표현한 것으로, 시스템이 갖는 기능 중에서 중요한 기능들이 유스케이스가 된다[9].

진화알고리듬을 위한 라이브러리가 가져야 할 중요한 기능에는 적용알고리듬, 개체형태, 선별방법, 유전연산자, 통계량 등이 포함되어야 하므로 이들 요소들이 유스케이스가 되고, 사용자가 액터가 된다. 이때 적용알고리듬과 개체의 표현방법은 기본적으로 사용자가 선택해야 할 요소이며, 선별이나 유전연산자, 통계량과 같은 요소들은 적용 알고리듬에 따라 종속될 수 있다. 이러한 개념에 입각하여 유스케이스 다이어그램을 작성하면 <그림 1>과 같이 나타낼 수 있다. 즉, <그림 1>은 사용자가 라이브러리 사용에 있어 진화알고리듬의 구성요소에 대해 어떤 방식으로 전략이나 방법, 또는 파라메터 등을 지정할 것인가를 설계한 것이다.



〈그림 1〉 단순진화 알고리듬을 위한 라이브러리의 유스케이스 다이어그램

<그림 1>에서 보면 사용자는 라이브러리 사용에 있어 개체의 형태와 적용알고리듬, 그리고 문제

의 특성에 따른 사용자 코드를 직접 선택하고, 나머지 요소들은 '알고리듬'을 통해서 간접적으로 선택할 수 있다. 따라서 '사용자'라는 액터와 '알고리듬 선택'이라는 유스케이스 간에는 의사소통 관계인 'communication' 관계로 두고, '알고리듬' 선택이라는 유스케이스는 '개체선택' 유스케이스를 비롯하여 다른 유스케이스들을 포함하는 관계인 'use'관계(또는 'include'관계라고도 한다)로 설정된다. 이와 같은 방법으로 사용자 입력을 완료하고, 입력 내용이 적용된 골격코드와 사용자 정의 코드를 결합하면 응용 프로그램이 작성되어 사용자는 자신이 원하는 프로그램을 얻게 된다.

2.2 객체지향 분석

사용자의 요구사항이 파악되면 시스템에 대한 객체지향적 분석이 이루어 진다. 객체지향 분석은 문제를 정의하고 정의로부터 모형들을 제작하여 시스템의 특성을 보여주는 단계이다[3, 7]. 일반적으로 객체지향 분석단계에서는 객체들과 그 특성을 식별하여 객체들의 정적 구조와 객체들 간의 관계를 보여주는 객체 모델링, 시간의 흐름에 따른 객체의 상태 변화를 보여주는 동적 모델링을 하게 된다.

2.2.1 객체 모델링

객체 모델링은 시스템에 요구되는 객체들을 보여줌으로써 주로 시스템의 정적인 구조를 포착하는데 사용된다. 우선 유스케이스 다이어그램을 기반으로 하여 시스템에 필요한 객체들은 다음과 같이 추출한다.

먼저 자료의 저장단위는 아니지만 모집단, 선별, 통계들을 총괄적으로 관리하는 알고리듬이 객체가 될 수 있다. 이때 단순 유전알고리듬, 안정상태 유전알고리듬, 생태 유전알고리듬 등을 각각 알고리듬 객체에서 상속된 객체로 구분한다. 다음으로 유연성을 높이기 위해 모집단과 모집단을 구성하는 개체를 각각 독립적인 자료구조를 갖는 객체로 추출하였다. 선별방법에 있어서도 확장성을 위해 한 방법에 대해 각각 독립적인 객체로 두고, 각 객체의 공통적인 부분을 모아 상위객체인 선별객체로 추출하였다. 개체의 적응도를 변환시키는 스케일링 방법도 선별방법과 마찬가지로 상위객체에 스케일링을 두고 각각의 방법들을 객체로 구분하였다. 진화의 과정 동안 생기는 통계 수치를 저장하는 통계 또한 객체가 된다. 유전연산(돌연변이와 교차)은 객체가 될 수 있으나 개체의 표현방법에 종속되어 비가능 해를 생성할 수 있으므로 객체로 추출하지 않았다. 클래스 라이브러리 구축에 필요한 객체들을 추출하여 정리한 것이 다음 <표 1>에 나타나 있다.

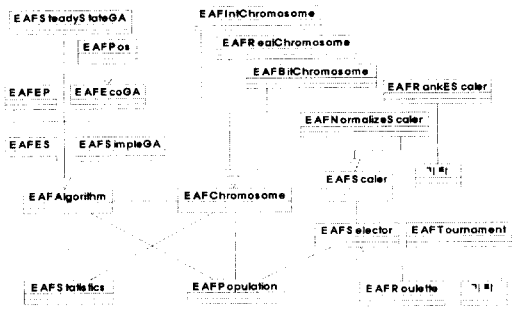
〈표 1〉 진화알고리듬 라이브러리 객체

객 체	객 체 이 름	주 요 기 능
알고리듬	EAFAlgorithm EcoGA, SimpleGA SteadyStateGA, ES, EP	전체 진화과정을 총괄한다. 진화 방법들
개 체	EAFChromosome Bit, Int, Real	하나의 잠재해 개체의 형태들
모 집 단	EAFPopulation	개체의 집합
스케일링	EAFScaling Scaling Method	적응도 값 변환을 위한 스케일링 스케일링 방법들
선 별	EAFSelector Selection Method	다음 세대를 구성하기 위한 선별 선별 방법들
통 계 치	EAFStatistics	통계치를 저장 또는 계산

객체를 추출한 다음에는 각 객체들 사이의 연관성을 규명해야 한다. UML에서는 객체들 사이에 일반(association), 상속(generalization), 집합(aggregation), 종속(dependency)의 4가지 관계가 존재한다[2, 16]. 일반관계는 '사용한다(use)'의 의미로 해석될 수 있는데, 대개 설계 초기에 클래스 간의 관계는 '일반'으로 표현되며, 개발이 진행됨에 따라 상속, 집합, 종속으로 구체화 된다. '상속'관계는 말 그대로 상속관계를 나타낸다. 즉, 상속받은 개체는

상위 개체의 모든 멤버 데이터와 멤버함수를 상속받게 된다. 집합관계는 객체간의 포함관계를 의미하는데, 예를 들어 자동차 객체의 경우, 엔진이라는 객체를 포함한다. 종속관계는 '일반'관계 중에서 상속, 집합이 아닌 관계에 해당한다. 구현관점에서 볼 때, 멤버함수 안에서 상대 클래스의 객체를 생성하여 사용하는 경우, 멤버함수의 인자로 상대 클래스의 객체가 넘어오는 경우 등이 이에 해당한다.

<그림 2>는 객체를 추출하여 객체 간에 상호관계를 표시한 객체 다이어그램이다. 알고리듬을 나타내는 EAFSimpleGA, EAFEcoGA, EAFSteady StateGA, EAFES, EAFEP 객체들은 공통적인 과정을 추상화시켜 만든 EAFAlgorithm이라는 객체로부터 상속된다. 문제의 특성에 따라 인자의 형태가 다른 개체들을 나타내는 객체들인 EAFIntChromosome, EAFBitChromosome, EAFRealChromosome들도 공통적인 부분을 추상화시켜 만든 EAFChromosome이라는 객체로 부터 모든 속성과 연산자가 상속된다. 선별방법 또한 공통의 자료와 연산자를 상위 객체인 EAFSelector에서 상속받는다. 이웃단위로 진화하는 생태 유전 알고리듬의 경우 이웃 모집단의 어느 위치에 어떠한 개체가 있는지를 알아야 할 필요가 있으므로 그 위치를 나타내는 EAFPos 객체는 EAFEcoGA 객체에 포함되는 관계를 갖는다. 그리고, 진화알고리듬의 핵심이 되는 알고리듬, 개체, 모집단들도 서로 복잡하게 연관 관계를 가지고 있다. 다음 세대에 생존할 개체를 선택하는 선별과정 또한 알고리듬, 개체, 모집단과 각각 종속 관계를 갖는다. 선별의 방법은 확률적일 수도 있고 확정적일 수도 있다. 확률적일 경우 모집단의 정보를 이용해 선별을 하고, 확정적일 경우 모집단의 가장 좋은 개체의 정보를 이용해야 하므로 선별 객체는 모집단 객체와는 물론 개체 객체와도 종속의 관계를 가져야 한다. 또한 알고리듬에 적용될 선별 방법의 정보가 필요하고, 선별 방법에 적용되는 알고리듬의 정보 또한 필요하므로 알고리듬 객체와 선별 객체는 상호 종속관계를 갖는다.



〈그림 2〉 객체 다이어그램

2.2.2 동적 모델링

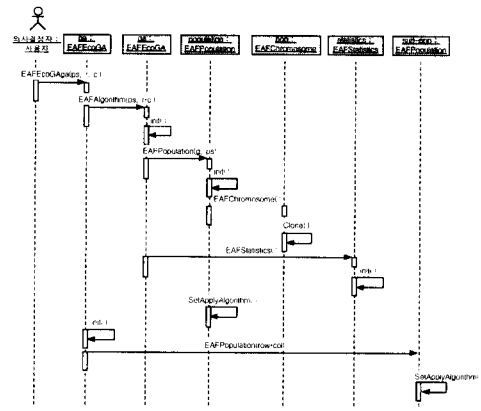
객체 다이어그램을 통해 각각의 객체와 그들 사이의 관계를 아무리 잘 정의했다고 해도 실제 실행되는 시스템 안에서 상호작용이 원활하지 않으면 결코 좋은 시스템이 될 수 없다. 시스템은 시간이 경과됨에 따라 변화되기 마련이다. 동적 모델링은 '시간의 흐름'에 따른 객체들과 객체들 사이의 상호작용을 모델링하는 것이다. UML에서 동적 모델링은 교류 다이어그램(interaction diagram)과 상태 다이어그램(state diagram)을 통해 구현된다. 교류 다이어그램은 객체들이 시스템 내에서 어떻게 상호 작용하는 지를 표현한 것으로, 순차 다이어그램(sequence diagram)과 협력 다이어그램(collaboration diagram)으로 나눌 수 있다. 순차 다이어그램은 객체들의 상호작용을 시간의 흐름에 따라 표현한 것이고, 협력 다이어그램은 여러 객체들의 정적인 관계를 중심으로 상호작용을 표현한 것이다. 순차 다이어그램은 시간의 흐름에 따라 순차적으로 객체간의 상호작용을 표현하므로 이해하기 쉽고 작성이 용이하다. 따라서 본 연구에서는 순차 다이어그램을 통해 동적모델링을 수행하였다.

진화알고리듬에 대한 객체모델링을 통해 추출된 객체들의 변화는 사용자가 적용하고자 하는 알고리듬으로 부터 시작되고, 알고리듬의 실행과정에서 발생하는 객체들의 상호작용으로 발생된다. 즉, 사용자가 적용 알고리듬을 선택하면, 이에 따른 다른 객체들의 초기화가 시작되고, 다음으로 알고리

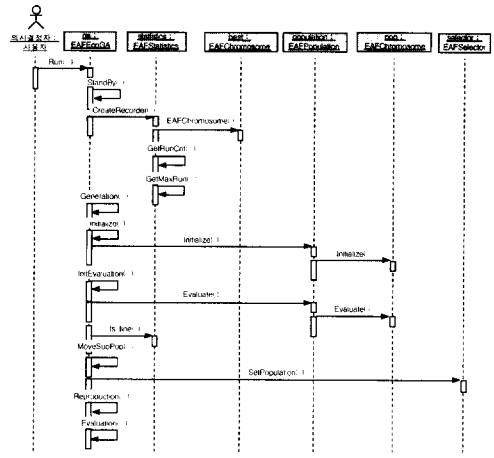
듬이 실행되면서 객체들의 변화가 일어나는 것이다. 따라서 알고리듬 선택에 따른 객체들의 변화와 알고리듬 실행에 따른 객체들의 변화에 따른 동적 모델링이 필요하다고 할 수 있다. 이를 위하여 본 연구에서는 적용 알고리듬으로 생태 유전알고리듬을 선택하였을 때 초기화가 일어나는 순서에 대한 시나리오와 프로그램의 실행에 대한 시나리오를 작성하여 각각에 대한 순차 다이어그램을 통해 동적 모델링을 수행하였다.

먼저, 사용자가 알고리듬을 EAFEcoGA로 정의하면 이 객체는 모집단을 생성하여 초기화 시키고, 모집단은 개체를 생성하여 초기화한다. 또한 알고리듬 객체는 문제에 필요한 통계치를 계산하는 EAFStatistics 객체를 생성하여 초기화하고, 다시 이웃 모집단을 생성하여 초기화 시킨다. 이러한 시나리오를 바탕으로 순차 다이어그램을 작성한 그림이 <그림 3>에 나타나 있다. 다음으로, 알고리듬이 실행되면 EAFEcoGA 객체는 통계치를 저장하기 위해 EAFStatistics 객체의 통계치 생성 함수를 호출하고 모집단을 초기화한다. 이 때 모집단 객체는 개체를 초기화 하고, 평가를 하게 되며, EAFStatistics 객체는 필요한 정보저장을 위해 EAFChromosome 객체를 생성한다. 통계치 객체에 저장된 종료조건을 검사하고, 종료조건을 만족하지 않으면, 진화과정을 반복한다. <그림 4>는 이러한 실행에 대한 시나리오를 순차 다이어그램으로 작성한 것이다.

UML에서 하나의 상태 다이어그램은 하나의 클래스에 대해 그려지는 것이 보통으로, 그 클래스의 객체가 가질 수 있는 상태와 상태의 전이를 유발한 이벤트, 그리고 상태가 전이할 때의 행동을 다이어그램에 묘사한다. <그림 5>는 생태 유전알고리듬에서 한 개체의 상태 변화를 보여주는 것이다. 개체는 초기화, 평가 과정을 거치고 임의의 선택에 의해 이웃 모집단 또는 전체 모집단에 포함된다. 이웃 모집단에 포함된 개체는 임의 선택에 의해 유전 연산을 하고 다시 이웃 모집단에 포함된다. 이 때 생성된 자손 대신에 대체되는 개체는 사라지게 되며, 계속 살아남은 개체는 전체 모집단에 포함되게 된다. 상태 다이어그램은 이러한 과정 속에서 개체가 갖는 상태의 변화를 표시한 것이다.



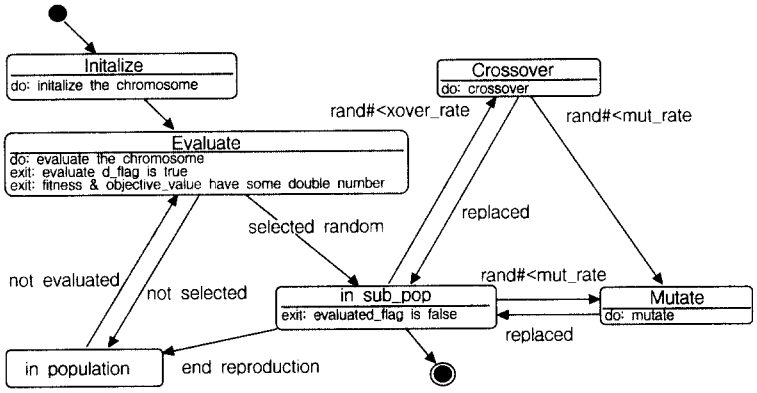
〈그림 3〉 알고리듬 입력 순차 다이어그램



〈그림 4〉 프로그램 실행 순차 다이어그램

3. 클래스 라이브러리의 설계 및 구현

객체지향 개발 방법론에서 가장 초점을 두는 관점은 객체 모델링 관점이다. 이는 정보에 관한 표현이 시스템의 객체지향 관점을 잘 나타내 주기 때문이다 그러나 객체 모델링 하나만으로는 각 객체들의 동작과 기능들이 나타나지 않으므로 완벽



〈그림 5〉 개체의 상태 다이어그램

하다고 할 수 없다. 객체에 대한 완벽한 모델링을 하기 위해서는 객체에 대한 동적인 정보가 나머지 두 모델에서 구해져야 한다. 결국 동적인 모델과 기능 모델이 객체 모델에 통합되어야만 좋은 정보가 모아질 수 있다. 그러므로 세 모델을 통합하는 작업이 이루어져야 하며, 이 단계가 일반적으로 객체지향 설계단계에 해당된다. 세 모델의 통합은 객체의 정적인 구조와 오퍼레이션을 포함하여 객체를 정의하는 것을 의미한다.

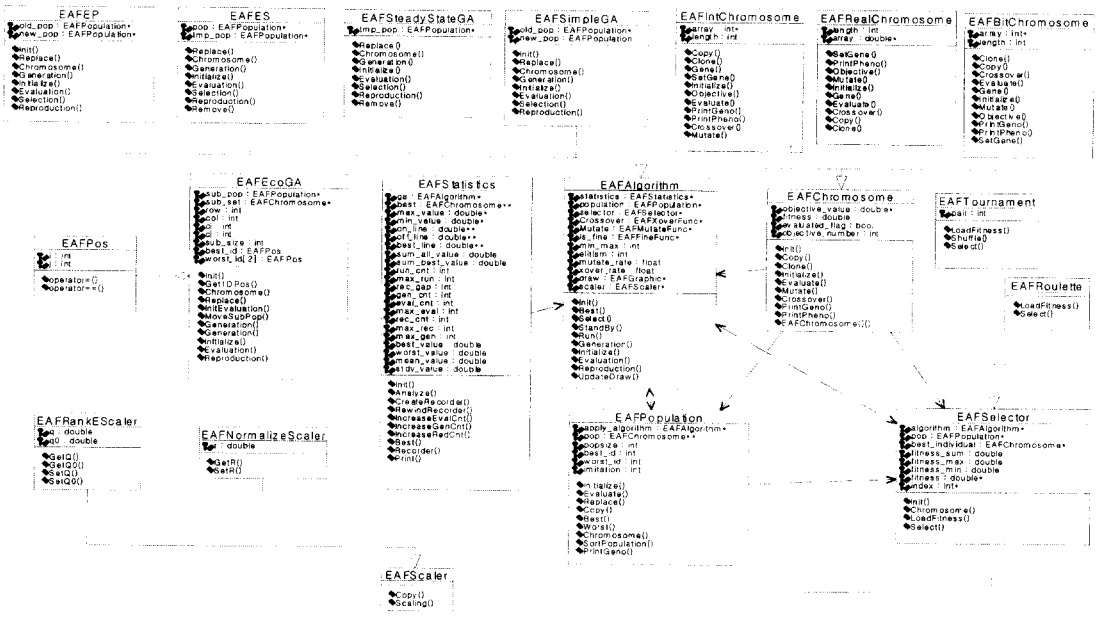
본 연구에서는 앞 절에서 작성된 여러 다이어그램을 토대로 시스템에 필요한 객체를 클래스로 추상화하고, 객체 간의 연관성을 토대로 클래스와 클래스 사이의 연관성을 구분한 다음, 각 클래스에 필요한 속성과 연산 기능을 정의하였다. 다음 <표 2>는 라이브러리의 주요 상위 클래스들과 상위 클래스의 주요 속성과 동작을 표시한 것이다.

설계한 클래스의 계층적 구조와 각 클래스들 사이의 연관성을 나타낸 것이 <그림 6>과 같은 클래스 다이어그램이다. 여기서 보는 바와 같이 진화 알고리듬 라이브러리는 한 클래스가 다른 클래스의 객체를 데이터 멤버로 갖는 경우가 많으므로 상당히 복잡한 연관관계를 갖는다.

이와 같은 과정을 통해 클래스들의 속성과 오퍼레이션이 모두 규명되면 순차 다이어그램, 상태 다이어그램 그리고 클래스 다이어그램을 토대로 구체적인 연산기능을 설계하고, 프로그래밍을 통해 라이브러리를 구현하게 된다. 객체지향 프로그래밍은 지금까지 해 온 분석과 설계를 컴퓨터가 이해할 수 있도록 코드로 구현하는 단계이다. 본 연구에서는 가장 범용적인 객체지향 언어인 C++ 언어를 사용하여 구현하였다. 이 때 ANSI C++에 표준화되어 있는 문법을 사용함으로써 특정 컴파일러에 종속되지 않도록 구현하였다.

〈표 2〉 클래스 중요 속성과 동작

클래스	속성, 동작	기 능
EAF-Algorithm	Statistics	통계량 객체
	Population	모집단 객체
	Selector	선별방법 객체
	Init()	초기화
	Run()	최대 반복실험회수까지 실행
	Generation()	실험 1회 반복
EAF-Chromo-some	Objective_value	목적함수 값
	Fitness	적응도 값
	Evaluate()	개체 평가
	Mutate()	돌연변이
	Crossover()	교차
EAF-Population	Apply_algorithm	알고리듬 객체
	Pop	이차원 개체 객체
	Initialize()	모집단 초기화
	Replace()	대체
EAF-Selector	Algorithm	알고리듬 객체
	Pop	모집단 객체
	Best_individual	가장 좋은 개체
EAF-Statistics	Ga	알고리듬 객체
	On_line	Online performance
	Off_line	Offline performance
	Max_eval	최대 평가 개체 수



〈그림 6〉 클래스 다이어그램

한편 기존의 모든 라이브러리는 DOS화면에서 텍스트 형태나 파일 형태로 실행 결과를 얻을 수 있도록 구성하였다. 그러나 윈도우즈 환경이 가장 보편적인 운영체계이므로, 본 연구에서 개발한 라이브러리는 응용 프로그램의 인터페이스를 하나의 모듈로 구성함으로써 사용자가 실험의 결과를 윈도우즈 환경에서 그래프를 통하여 볼 수 있도록 구현하였다. 즉, 사용자는 DOS 형태의 텍스트로 통계치의 결과를 받아볼 수 있고, 이를 그래프로 표현한 윈도우즈 창으로 받아볼 수 있도록 구현하였다.

4. 적용 시나리오

이제까지 진화알고리듬에 대한 사용자 요구사항 파악 및 객체지향 분석과 객체지향 설계단계를 거쳐 이를 바탕으로 라이브러리를 개발하였다. 본 장에서는 구현된 라이브러리를 사용하여 응용프로그램에 적용하는 시나리오를 다음의 예제 프로그램을 이용하여 설명한다.

```
EAFBitArrayChromosome chromosome(20) ;   // (a)
EAFEcoGA ga(&chromosome, 10, 10) ;       // (b)

EAFRoulette selector ;                   // (c)
ga.SetSelector(&selector) ;              // (c)
ga.SetMinMax(MINIMIZE) ;                 // (d)

ga.SetCrossOver(::OnePointCrossover) ;   // (e)
ga.SetXoverRate(0.8) ;                   // (f)
ga.SetRecGap(100) ;                      // (g)
ga.Run() ;                               // (h)
```

(1) 개체형태와 적용알고리듬 선택

사용자는 먼저 적용하고자 하는 문제에 적합한 개체의 표현방법과 적용 알고리듬을 결정해야 한다. 예제 프로그램의 (a)는 개체의 표현방법을, (b)는 적용 알고리듬을 지정한 것이다. 즉, (a)의 경우, 개체의 길이가 20인 이진표현을 사용한다는 것을 의미하고, (b)의 경우는 적용 알고리듬으로 10×10 의 2차원 격자구조인 생태적 유전알고리듬을 사용한다는 의미를 갖는다. 사용자는 적용하고자 하는 표현방법과 알고리듬을 다음 <표 3>과 같은

〈표 3〉 개체의 표현형태와 적용알고리듬

	정 의 코 드	설 명
개 체 형 태	EAFBitChromosome *개체변수*(개체 길이) EAFIntArrayChromosome *개체변수*(개체 길이) EAFRealArrayChromosome *개체변수*(개체 길이)	이진표현 1차원 정수값 배열 표현 1차원 실수값 배열표현
알고리듬 형 태	EAFSimpleGA *알고리듬변수*(&*개체변수*, 모집단 크기) EAFSteadyStateGA *알고리듬변수*(&*개체변수*,모집단 크기) EAFEcoGA *알고리듬변수*(&*개체변수*,가로크기,세로크기) EAFES *알고리듬변수*(&*개체변수*,모집단 크기) EAFEP *알고리듬변수*(&*개체변수*,모집단 크기)	표준 유전알고리듬 안정상태 유전알고리듬 생태 유전알고리듬 진화전략 진화프로그래밍

〈표 4〉 선별방법과 스케일링 방법의 선택

	방 법	사용방법 및 사용예
선별방법	EAFRoulette (확률바퀴) EAFRanking (순위) EAFTournament (토너먼트) EAFSUniversal (확률적 균등) EAFDTTruncation (확정적 절단) EAFDBlock (확정적 블록) EAFDeterministic (확정적 샘플링) EAFRStochastic (확률적 잔여샘플링)	선별방법 *선별변수* ; *알고리듬변수*.Set Selector(&*선별변수*) (ex) EAFRoulette selector ; ga.SetSelector(&selector) ;
스케일링 방 법	EAFRankLScaler (선형 순위) EAFRankEScaler (지수적 순위) EAFNormalizeScaler (정규화) EAFLinearScaler (선형) EAFPowerLScaler (지수법칙) EAFSigmaTScaler (시그마 절단)	스케일링방법 *스케일링변수* ; *알고리듬변수*.SetScaling(&*스케일링변수*) (ex) EAFLinearScaler scaler ; ga. SetScaling (&scaler) ;

형태로 코드를 작성하면 된다. 여기서 이태릭체는 사용자가 정의한 변수를 의미한다.

개체의 표현방법에 따른 초기 모집단의 생성방법과 각 개체의 적응도 평가방법은 적용하고자 하는 문제에 따라 다르게 된다. 따라서 사용자는 반드시 개체의 초기화 방법과 적응도 함수에 대한 코드를 작성해야 하며, 이 때 각 코드의 형태는 다음과 같다.

```
void 개체형태::Initialize( ){
    // 개체의 초기화
}
double 개체형태::Objective(개체형태& 변수명) {
    // 적응도 함수
}
```

(2) 진화알고리듬 요소선택

개체의 형태와 알고리듬을 결정한 다음, 적용문제에 적합한 선별방법과 스케일링 방법, 유전 연산자 등을 결정해야 한다. 먼저, 사용자가 선별과 스케일링 방법을 선택하기 위해서는 예제 프로그램의 (c)와 같은 형태로 지정한다. 즉, 사용자는 선별방법과 스케일링 방법에 대해 각각 변수를 선언한 다음 이를 알고리듬 변수에 연결하는 코드를 작성해야 한다. 라이브러리에 구현된 선별과 스케일링에 대한 방법과 일반적인 표현방식은 <표 4>와 같다.

이 때 적용할 문제가 최소화 문제인가 또는 최대화 문제인가에 따라 각 방법이 달라질 수 있다. 또한 가장 좋은 해를 다음 세대에 생존시킬 것인가(elitism사용 여부)의 여부도 결정해야 한다. 최

대화/최소화 여부와 elitism을 적용하고자 할 때 예제 프로그램의 (d)와 같은 형태로 지정하면 되고, 이를 일반적인 표현방식으로 각각 나타내면 다음과 같다.

알고리듬변수.SetMinMax(MINIMIZE 또는 MAXIMIZE) ;
알고리듬변수.SetElitism() ;

교차와 돌연변이와 같은 유전연산자는 적용하는 문제에 따라 매우 다양한 방법이 있으므로 이들을 모두 라이브러리에 포함할 수 없다. 따라서 본 연구에서는 2.2.2절에서 제시한 대표적인 표현방법에 따른 유전연산자만을 라이브러리 함수로 포함하였으며, 사용자가 적용문제에 맞게 개발한 유전연산자들도 쉽게 라이브러리에 추가시킬 수 있도록 하였다. 즉, 미리 정의된 유전연산자의 사용은 예제 프로그램의 (e)와 같은 형태로 작성하고, 사용자가 정의한 유전연산자의 경우에는 다음과 같이 정의하고, 이를 알고리듬에 연결시키면 된다.

교차의 경우

```
EAFChromosome* 교차함수명(EAFChromosome*
p1, EAFChromosome* p2) {
   개체형태& parent1 = *(개체형태 *)p1 ;
   개체형태& parent2 = *(개체형태 *)p2 ;
   …       // parent1과 parent2에 의한 교차방법
             정의
}
알고리듬변수.SetCrossover(교차함수명) ;
```

돌연변이의 경우

```
EAFChromosome*   돌연변이함수명(EAFCh-
romosome* p) {
   개체형태& ind = *(개체형태 *)p ;
   …       // ind에 의한 돌연변이 방법 정의
}
알고리듬변수.SetMutation(돌연변이함수명) ;
```

적 용

```
EAFChromosome* UDF_CX(EAFChromosome*
p1, EAFChromosome* p2) {
   …       //교차방법 정의
}
EAFChromosome*UDF_MX(EAFChromosome*
p) {
   …       // 돌연변이 방법 정의
}
ga.SetCrossover(UDF_CX) ;
ga.SetMutation (UDF_MX) ;
```

(3) 파라메타 입력

유전파라메타로는 모집단의 크기, 교차율, 돌연변이율, 종료조건 등이 있다. 이 때 모집단의 크기는 예제 프로그램의 (b)와 같이 입력되며, 교차율과 돌연변이율 등은 (f)와 (g)와 같은 형태로 지정한다. 여기서 (g)는 통계량의 기록간격을 지정한 것이다. 통계량은 각 세대별로 가장 좋은해와 세대평균, 그리고 최종적으로 online, offline성능을 기록하게 된다. 다음 <표 5>는 필요한 파라메터를 지정하는 방법을 나타낸 것이다.

〈표 5〉 파라메터의 입력방법

파라메타	입 력 방 법	사 용 예
교차율	*알고리듬변수*.SetXoverRate(교차율) ;	ga..SetXoverRate(0.8) ;
돌연변이 율	*알고리듬변수*.SetMutateRate(돌연변이율) ;	ga.SetMutateRate(0.05) ;
종 료 조 건	*알고리듬변수*.SetMaxEval(개체평가횟수) ;	ga.SetMaxEval(1000)
통계량 기 록	*알고리듬변수*.SetRecGap(기록간격)	ga.SetRecGap(100)

마지막으로 본 연구에서는 응용 프로그램의 인터페이스를 하나의 모듈로 구성함으로써 사용자가 실험의 결과를 윈도우즈 환경에서 그래프를 통하여 볼 수 있도록 구현하였다. 사용자가 DOS환경에서 응용프로그램을 실행하고자 할 경우에는 main( )함수에 앞에서 설명한 방법만을 지정하면 된다. 만약 윈도우 환경에서 응용프로그램을 실행하고자 할 경우에는 main( )함수 앞에 다음과 같

은 코드만 추가하면 된다.

```
int WINAPI WinMain(HINSTANCE hInstance,
HINSTANCE hPrevInstance,
                PSTR szCmdLine, int iCmdShow)  {
    int x = EAFWinInit(hInstance, hPrevInstance,
            szCmdLine, iCmdShow)
    return x ;
}
```

5. 결 론

본 연구에서는 단순진화 알고리듬에 대해 객체지향 개발 방법론을 이용하여 안정적이고 확장이 용이한 라이브러리를 모델링하고 구축하였다. 이때 사용자 편의성을 중시한 설계로 사용자로 하여금 최소한의 작업으로 진화알고리듬을 통한 문제해결을 가능케 하고, 새로 개발된 요소나 방법의 추가가 가능하면서 공진화 알고리듬으로의 확장이 용이한 라이브러리의 개발에 중점을 두었다.

개발된 라이브러리를 이용하여 작성된 사용자 응용프로그램은 코드가 매우 간결하고 코드의 높은 가독성(readability)으로 인하여 쉽게 다른 문제로의 응용이 가능하다. 또한 라이브러리의 확장성을 통하여 향후 개발될 여러 진화알고리듬의 기법들을 효과적으로 라이브러리에 추가할 수 있음으로써 확장성과 활용도가 높은 진화알고리듬 라이브러리가 될 것이다. 향후 단순진화 알고리듬과 최근 진화계산 분야에서 활발히 연구되고 있는 공진화 알고리듬을 모두 지원할 수 있는 라이브러리를 개발할 예정이다.

참 고 문 헌

[1] 김여근, 윤복식, 이상복 공저, 「메타 휴리스틱」, 영지문화사, 1997.

[2] 윤청, 「성공적인 소프트웨어 개발 방법론」, 생능출판사, 1996

[3] 이수연, 정호연, 서광언, 김여근, 단순진화 알고리듬을 위한 애플리케이션 프레임워크 개발, 「산업공학」 제12권, 제4호(1990.12), pp.540-550.

[4] Coad P. and E. Yourdon, *OOAObject-Oriented Analysis*, 2nd Edition, Prentice-Hall, Englewood Cliffs, New Jersey, 1990.

[5] Corcoran, A.L., Wainwright. R.L., *LibGA : A user-friendly workbench for order-based genetic algorithm research*, 1993.

[6] Fogel, L.J., A.J. Owens and M.J. Walsh, *Artificial Intelligence through Simulated Evolution*, NY : John Wiley, 1966.

[7] *GAME*, University College London

[8] Grady Booch, *Object-Oriented Analysis And Design with Applications*, The Benjamin/Cummings Publishing Company Inc., 1994.

[9] Holland J.H., *Adaptation in Neural and Artificial Systems*, Univ. of Michigan Press, Ann Arbor, 1975.

[10] Jacobson I., *Object-Oriented Software Engineering a Use Case Driven Approach*, The Addison-esley Publishing Company Inc., 1992.

[11] Van Hemert J.I., http://www.wi.leidenuniv.nl/~jvhemert/leap/,1996

[12] Matthew Wall, *Mattew's GALib*, http://lancet.mit.edu/ga/,1995.

[13] Merelo, J. J., GeNeura. G., Granada GNAT, "Genetic and Neural Auxiliary Toolbox Reference Manual and User's guide," Available by anonymous ftp from kal-dir. ugr.es:/pub/GAGS.tar.gz., 1997.

[14] Michalewicz, *Genetic algorithms + Data structures = Evolution programs 2nd Ed.*, Springer - Verlag, Berlin, 1994.

[15] Rechenberg, I., "Cybernetic Solution Path of an Experimental Problem," *Royal Aircraft Es-*

tablishment, Library Translation 1122, Farnborough, Hants, U.K, 1965.

[16] Rumbaugh. J, Blaha M., Premerlani W., Eddy E., Lorensen W., *Object-Oriented Modeling and Design*, Prentice-Hall, Englewood Cliffs, New Jersey, 1991.

[17] Terry Q., *Visual Modeling with Rational Rose and UML*, Addison Wesley, 1998.

[18] Yuval Davidor, "A naturally occurring niche & species phenomenon : The model and first results," *4th International Conference on Genetic Algorithms*, 1991.