단일 실행 시뮬레이션 최적화를 위한 Reverse-Simulation 기법

Reverse-Simulation Method for Single Run Simulation Optimization

이 영해*, 박 경종*
Young-Hae Lee, Kyoung-Jong Park

Abstract

Simulation is commonly used to find the best values of decision variables for problems which defy analytical solutions. This objective is similar to that of optimization problems and thus, mathematical programming techniques may be applied to simulation. However, the application of mathematical programming techniques, e. g., the gradient methods, to simulation is compounded by the random nature of simulation responses and by the complexity of the statistical issues involved.

In this paper, therefore, we explain the Reverse-Simulation method to optimize a simulation model in a single simulation run. First, we point the problem of the previous Reverse-Simulation method. Secondly, we propose the new algorithm to solve the previous method and show the efficiency of the proposed algorithm.

1. 서 론

현대의 많은 복잡한 실제 시스템은 이산형 시스템(Discrete Event System)으로 모델링된다. 이들 시스템들은 일반적으로 이산 사건의 발생과 시간에 따른 상태변화에 의해 유도된다. 이와 같은 이산 사건의 복잡한 관계는 확정적 추론 기법(deterministic approximation techniques)이나 확률적 시뮬레이션(stochastic simulation)에 의해 연구되어야 한다.

그러나, 도착 시간과 서비스 시간의 랜덤 변수들이 임의의 분포, 제한된 버퍼 크기, 우선순위 등을 가정하는 모델들은 분석적인 방법으로 해결할 수 없다. 따라서 이러한 모델들은 시뮬레이션 기법으로 해결해야 한다[12].

시뮬레이션 모델링은 제조 설비와 같은 복잡한 확률적 시스템을 분석하고, 수행 척도 등을 계산하기 위해 광범위하게 사용되고 있다. 때때로 우리는 제어가능한 파라메터들에 대한 시스템의 수행 척도를 최적화하는데 관심이 있다. 시뮬레이션 모델의 최적값을 찾기 위한 전통적인 방법들은 계산량이 많은 시뮬레이션의 반복을 필요로 한다. 지금까지의 시뮬레이션 최적화 방법들은 시뮬레이션의 반복 수를 줄이면서 최적값을 찾는 것으로 가고 있다고 해도 과언은 아니다.

최적화 문제는 시스템 요소의 일부 혹은 전부가 확률적이고 복잡하므로, 목적함수가 분석적으로 표현될 수 없으

* 한양대학교 산업공학과

며, 여러 분야의 복잡한 시스템 모델링 과정을 고찰해 볼 때, 정확한 분석적 표현이 가능한 시스템은 찾기 힘들다. 비록, 분석적인 표현이 가능한 경우에도 대부분은 비선형이거나 확률적이다. 이러한 분석적 표현에 따르는 어려움으로 인하여, 컴퓨터 시뮬레이션에 의한 모델링이 복잡한 시스템을 연구하는 가장 효과적인 수단으로 인식되고 있다.

시뮬레이션 최적화란 시뮬레이션 방법을 사용하는 확률적 시스템에 대한 최적화 연구이며, 확률적 최적화의 일부로서 설명될 수 있다. 시뮬레이션을 사용한 평가는 지금까지 What if 문제에 대응하는 형태를 나타내었다. 최근 수년 동안 컴퓨터 시뮬레이션의 성공이 How to 질문에 답을 할 수 있는 정도로 확대되었다. What if 질문은 시스템의 결정 변수들에 대한 주어진 일련의 값의 수행척도에 대한 대답을 준다. 한편, How to 질문은 시스템의 결정변수에 대한 최적값을 찾아주므로 주어진 응답치나 응답 벡터가 최대화되거나 최소화된다. 지금까지는 시스템의 정량적인 결정변수의 최적화를 다루는 시뮬레이션 최적화 절차에 행해졌으며, 복잡한 시스템 구조의 최적화를 다루는 절차에 대한 연구가 계속되고 있다. 시뮬레이션 최적화에 대한 연구는 Ho와 Cao[4], Rubinstein 과 Shapiro [12], Glynn[2], Meketon[9], Jacobson과 Schruben[5], Safizadeh[8] 등에 의해서 발전되었다.

그러나, 기존의 시뮬레이션 최적화 방법은 다음과 같은 문제점이 존재한다. 첫째, 목적식이나 제약식의 분석적 표현이 존재하지 않으며, 시스템에 대한 정량적인 분석이 거의 불가능하다. 이것은 지역 경사도(local gradients)의 정확한 계산이나 미분 가능성을 어렵게 한다. 둘째, 목적식과 제약식은 결정 변수의 확률적 함수이며, 이를 원하는 수리적 표현으로 구성하기가 어렵다. 그러므로 구하고자 하는 최적점을 찾는 절차가 복잡하며 많은 노력이 요구된다. 셋째, 컴퓨터 시뮬레이션은 분석적인 함수를 평가하는 것 보다 훨씬 더 많은 실행을 하게 되므로 비용이 많이 소비된다. 따라서 최적화 알고리즘의 효율이 절대적으로 요구된다. 넷째, 시스템을 모델링하는 데 있어서 대부분의 시뮬레이션 실행자는 시뮬레이션 언어를 사용하는 반면, 최적화 부분은 범용 프로그래밍 언어 사용을 요구하고 있다. 그러나 시뮬레이션 모델을 일반적인 최적화 루틴에 인터페이스하는 문제는 결코 단순한 작업이 아니다. 다섯째, 사용자의 요구가 갈수록 구체화되고 중요시되고 있어서 시뮬레이션 최적화를 수행할 때, 구체적인 목표치를 제시하는 사용자의 요구에 부응하는 시스템이 개발되어야 한다.

따라서 본 연구에서는 이러한 문제점들을 감소시키고자 시뮬레이션 최적화를 위해 새로운 기법인 단일 실행 Reverse-Simulation 최적화 기법을 소개하고자 한다. 본 연구의 2장에서는 새로운 최적화 기법인 Reverse-Simulation 최적화 기법을 설명하고, Reverse-Simulation 최적화 기법의 성공을 위해 전문가 시스템과 연결하는 방법을 3장에서 보여준다. 4장에서 Reverse-Simulation 최적화 기법의 문제점을 제시하고 해결 방안에 대해서 5장에서 설명한다. 마지막으로 6장에서 결론 및 추후 연구 과제를 제시한다.

2. Reverse-Simulation 기법

Reverse-Simulation 기법은 Wild 와 Pignatiello[14]에 의해 1991년에 처음으로 제안되었으며, Kwanjai 와 Wild[6], Wild 와 Jackson[13]에 의해 부분적인 발전이 있었고, 1994년에 Wild 와 Pignatiello[15]에 의해 개념적인 틀이 제공되었다. Reverse-Simulation은 바람직스러운 수행도 목표 값(target values)이나 일정한 값의 범위를 결정하고 이러한 사용자 정의(user-defined) 수행도 목표를 만족시키기 위해 동적으로 시스템 설계를 조정하는 것이다. Reverse-Simulation은 시스템 수행도 평가와 최적화에 대한 시작점을 찾기 위해 시스템 설계 변수들에 대한 초기의 가능해 값들을 결정하는 데 유용한 정보를 제공한다. 즉, Reverse-Simulation은 시스템 수행도 기준을 만족시키는 시스템 설계 변수들에 대한 초기 환경을 결정하는 과정을 구축한다.

Reverse-Simulation을 위해서 분석가는 출력 데이터를 미리 예측할 필요가 없다. 그러나, 분석가는 좋은 시스템 수행도가 무엇이고, 수행 척도의 바람직스러운 목표 값이나 범위를 미리 규정하는 것이 필요하다. 이 때, 시스템 목표들은 바람직스러운 시스템 수행도를 표현하는 시스템 제약식이 된다. Reverse-Simulation 기법은 분석가의 수행도 목표들을 시뮬레이션 프로그램의 수행 동안 동적으로 시스템 설계 변수 값들을 조작 함으로서 설정된 제약식 만족을 통해서 이러한 목표들이 성취될 수 있는 시스템 환경으로 연결하려고 시도한다.

Reverse-Simulation의 수행 단계는 다음과 같다.

단계 1 : 수행 척도의 목표 값이나 범위에 따라 시스템 목표를 입력한다.

단계 2 : 시스템 설계 변수 값들이 목표 수행도를 만족하도록 동적으로 시뮬레이션을 실행한다.

단계 3 : 조합 과정을 수행해 안정된(stable) 시스템 설계를 만들어 내는 시스템 설계 변수들의 가능해 값들을 출력한다.

단계 4 : 시스템의 목표치를 만족하는 최적 또는 시스템의 목표치에 가장 근접한 최상의 시스템 설계를 찾기 위해 계속적인 시뮬레이션으로 시스템 설계 변수들의 가능해를 설정한다.

지금까지 설명한 바와 같이 Reverse-Simulation은 현장 관리자가 필요로 하는 결과를 얻기 위해 시뮬레이션 시작 시의 시스템 상태를 설정하자는 이론이다. 따라서 Reverse-Simulation은 단계 1에서 설명한 것 처럼 먼저 시스템 목표 값이나 범위가 주어져야 한다. 이러한 시스템 목표 값은 최소한 1개이며, 일반적으로는 2개 이상이 될 것이다. 두 번째는 단계 2에서 설명한 것 처럼 단계 1에서 주어진 시스템 목표치를 가지고 실제로 시뮬레이션을 수행하여 그 결과를 얻는다. 단계 2의 시뮬레이션 실행 단계에서 시스템 목표를 만족하는 시스템 설계 변수들이 여러 쌍이 얻어질 수 있다. 이러한 시스템 목표 값을 만족하는 시스템 설계 변수들의 가능해 값들을 얻어내는 과정이 단계 3이다. 단계 4는 이렇게 얻어진 결과에서 최적 또는 최상의 시스템 설계 변수 값들을 결정한다.

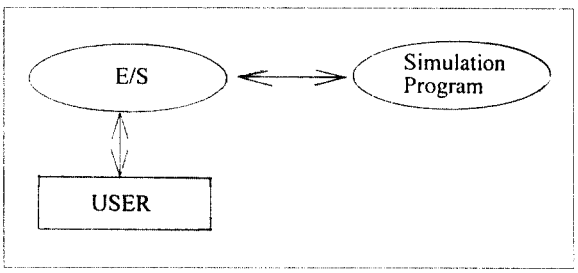
단계 2에서 우리가 시뮬레이션을 수행하면서 시스템 설계 변수 값들이 목표 수행도를 만족하도록 동적으로 조작되도록 한다고 설명하였는데 실제로 우리는 기존의 시뮬레이션 수행 방법처럼 한 번 시뮬레이션을 수행하여 오프-라인(off-line)으로 결과를 판단하고, 만족스럽지 못하면 다시 시뮬레이션을 반복하는 방법이 아닌 시뮬레이션을 수행하면서 동적으로 결정하는 즉, 온-라인(on-line) 방법을 연구 대상으로 한다.

그러나, Wild 와 Pignatiello에 의해서 제안된 Reverse-Simulation은 최적의 초기 상태를 선택하기 위해서 단계 4의 가정을 수행해야 한다. 따라서 본 연구에서는 안정 상태(steady state) 와 최적 상태(optimal state)라는 개념을 추가하여 단계 4가 필요없고, 단계 3에서 종료되는 Reverse-Simulation 최적화 방법을 제시하고자 한다.

Reverse-Simulation 최적화를 수행하는 동안 온-라인으로 시뮬레이션 결과가 시스템 목표치를 만족하는 가를 확인하는 방법은 그렇게 간단하지가 않다. Wild 와 Pignatiello는 Reverse-Simulation을 수행하면서 목표로 하는 시스템 수행도를 온-라인으로 만족하는 방법으로 시뮬레이션과 전문가 시스템(Expert Systems)을 연결하였으며, 본 연구에서도 시뮬레이션과 전문가 시스템을 연결하는 방법으로 위의 문제를 해결한다.

3. 시뮬레이션 프로그램과 전문가 시스템의 연결

시뮬레이션과 전문가 시스템은 매우 유사한 점이 많다. 두 방법은 모두 다 불확실한 세계를 모델링하기 위해 다양한 표현 방법을 사용한다. 시뮬레이션과 전문가 시스템이 추구하는 목적은 의사 결정을 돕는 computer-based 모델을 제공하는 것이다. 시뮬레이션과 전문가 시스템을 결합하기 위한 분류법은 O'Keefe[10]에 의해서 잘 설명되어 있다.



〈Figure 3.1〉 Association of Expert System and Simulation Program

Wild와 Pignatiello에 의해서 사용된 방법은 사용자가 전문가 시스템에게 시스템 목표치와 우선 순위에 관한 정보를 주면, 전문가 시스템이 시뮬레이션 프로그램과 정보를 주고 받으면서 온-라인으로 시스템 상태를 파악하는 방법이며 이것을 그림으로 나타내면 다음의 〈Figure 3.1〉과 같다. 이 방법은 지식 전문가들에게 많은 흥미를 끌고 있으며, 사용자나 실제 환경에 의해 전문가 시스템이 테스트되는 것이 아니라 시뮬레이션에 의해 전문가 시스템이 실험된다. 이 방법은 개발 시간이 단축될 뿐만 아니라 테스트가 매우 합리적으로 수행될 수 있다. 이러한 접근 방법이 이용 가능한 응용 영역은 실-시간 공정 제어 분야이며,

앞으로도 많은 연구가 수행되어야 할 부분이다.

시뮬레이션 프로그램과 전문가 시스템을 연결하기 위해 M/M/s 대기 모형을 가지고 설명한다. Reverse-Simulation에서 다루는 대기 행렬 문제는 주어진 시스템 목표치를 만족하는 시스템의 시작 상태를 찾기 위해서 서버의 수가 계속해서 변하기 때문에 M/M/s 문제가 된다. 본 연구에서 사용되는 전문가 시스템은 일반적인 전문가 시스템과 약간 다르다. 전문가 시스템에서 나온 결론은 사용자와의 직접적인 상담에 의해서 나온 것이 아니라 시뮬레이션 프로그램과의 상담에서 나온 결론이다. 즉, 전문가 시스템은 시뮬레이션이 수행되므로서 시뮬레이션 프로그램에 의해 불려지고, 전문가 시스템에 의해 조언이 시뮬레이션이 실행됨에 따라 동적으로 시뮬레이션 프로그램에 주어진다.

지식 표현에 있어서도 이용가능한 전문가 시스템과 대화를 할 수 있는 상업적인 시뮬레이션 소프트웨어는 현재 나와 있지 않다. 따라서, 이러한 문제점을 해결하기 위해 시뮬레이션 소프트웨어와 일반 범용 언어가 쉽게 인터페이스로 연결되는 시뮬레이션 언어를 사용한다. 실제로 이러한 조건을 만족시키는 시뮬레이션 언어가 많이 나와 있지만 본 연구에서는 SLAMSYSTEM을 사용하며, SLAMSYSTEM은 FORTRAN과 인터페이스를 할 수 있다[11].

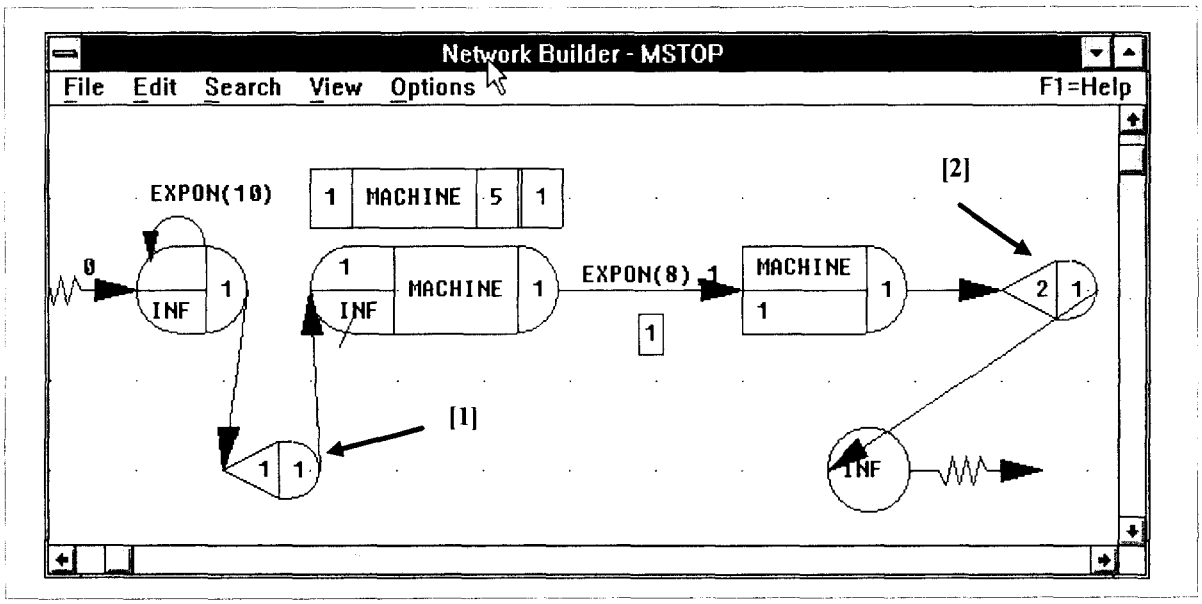
전문가 시스템의 생성 규칙들은 FORTRAN에서 쉽게 *If Then Else* 구조로 표현될 수 있다. 따라서 시뮬레이션 프로그램이 SLAMSYSTEM으로 구축되기 때문에 Reverse-Simulation 기법을 표현하는 프로토타입 시스템에서 사용되는 전문가 시스템은 FORTRAN으로 구성된다.

M/M/s 시스템에서 우리의 시스템 목표치를 서버의 이용률이 78 ~ 85% 가 되면 만족한다고 가정하고, 이 때의 초기 조건을 결정한다. 아래의 〈Figure 3.2〉는 SLAMSYSTEM에서 구성된 모델을 나타낸다.

〈Figure 3.2〉는 SLAMSYSTEM 네트워크 모델링이며, 전문가 시스템과 인터페이스 될 곳을 [1] 과 [2]로 표시하였다. 본 연구에서는 전문가 시스템이 정보를 주고 받는 장소를 개체가 시스템에 들어올 때와 서비스를 받고 시스템을 빠져나가는 시간으로 한정하였다. 이렇게 규정한 이유는 시스템에서 개체의 증감이 발생할 때가 시스템의 변화 상태를 알기가 가장 좋다는 가정하에 정하였다.

다음 단계는 전문가 시스템 영역에 들어갈 룰 베이스(rule base)를 구축하는 단계이다. 위에서 시스템 목표치를 서버 이용률이 78 ~ 85%로 정했기 때문에 여기에 맞는 룰 베이스를 구축한다.

룰 베이스를 구성할 때는 개체가 시스템에 들어올 때의



〈Figure 3.2〉 SLAMSYSTEM Network Model

전문가 시스템 영역 〈Figure 3.2〉의 [1]과 시스템을 빠져 나갈 때의 전문가 시스템 영역 [2]로 나누어 룰을 구성한다. 먼저 [1]번 영역의 룰을 설명하면 다음의 〈Table 3.1〉과 같다.

〈Table 3.1〉 The front rule sets of Expert system

RULE 1	IF	There is an idle resource associated with queue Q, and server utilization S is less than 0.78,
	THEN	Remove a part from queue Q to begin processing with resource R.
	ELSE	Go to Rule 2.
RULE 2	IF	There is not an available resource R, and server utilization S is greater than 0.85,
	THEN	Server S = S + 1 and continue simulation.
	ELSE	Go to Rule 3.
RULE 3	IF	There are the number of idle resources R that are grater than 2, and server utilization S is less than 0.78.
	THEN	Server S = S -1 and continue simulation.
	ELSE	Go to Rule 4.
RULE 4	IF	Server utilization S is 0.78 〈= S 〈= 0.85,
	THEN	Stop simulation(MSTOP = -1) and report output file.
	ELSE	Inform to user about problems

개체가 시스템을 빠져 나갈 때인 영역 [2]의 룰을 설명하면 다음의 〈Table 3.2〉와 같다. 〈Table 3.2〉는 〈Table 3.1〉과 기본적인 룰 베이스는 같으며 RULE 2만 수정된 것이다.

지금까지 시스템 목표치에 대해서 〈Table 3.1〉과 〈Table 3.2〉에서 룰 베이스를 구축했지만, 시스템 목표치가 많아지면, 여기에 대한 룰 베이스도 추가되어야 한다. 룰 베이스를 구축할 때는 우리가 얻고자 하는 정보가 특정한 영역의 정보이면, 여기에 따라 룰 베이스를 자세하게 구성해야 한다. 위에서 룰 베이스를 설명하였으나, 본 연구의 초점은 좋은 룰 베이스를 구축하는 것이 아니고 Reverse-Simulation에서의 안정 상태를 결정하는 알고리즘이므로 좋은 룰 베이스 구축 방법은 인공 지능 전문가들에게 남겨두고자 한다.

〈Table 3.2〉 The end rule sets of Expert system

RULE 2	IF	There is not an available resource R, and server utilization S is greater than 0.85, and the number of parts in queue Q is greater than 1.
	THEN	Server S = S + 1 and continue simulation.
	ELSE	Go to Rule 3.

4. Reverse-Simulation 최적화 기법의 문제점

시뮬레이션을 실행하기 전에 먼저 임의로 초기 환경을 설정한다. M/M/s 대기 모형에서 도착 시간 간격이 10분이고, 서비스 시간이 8분이라고 정하고 초기의 작업자 수를 15로 정한다. 여기서 정하는 변수들은 사용자가 임의로 설정하면 된다. 따라서 도착률(λ)은 시간당 6이고, 서비스률(μ)은 7.5로 계산된다. λ와 μ는 안정 상태 결정 알고리즘을 사용한 경우와 안한 경우를 비교하기 위한 자료로서 나중에 이론치를 구하기 위해 사용한다.

지금까지 정의된 환경을 가지고 M/M/s 대기 모형의 Reverse-Simulation 결과를 살펴보자. 아래의 〈Table 4.1〉은 안정 상태 결정 알고리즘을 사용하지 않고 시뮬레이션 프로그램과 전문가 시스템을 온-라인으로 사용한 결과이며, 전체 실행 회수는 100회이며 평균치를 보여준다.

〈Table 4.1〉 The results of Reverse-Simulation

The # of selected machines	2
Average waiting times in queue	2.216
Average numbers in queue	0.216
Average utilization	0.845
Entity count	10

〈Table 4.1〉의 결과를 보면, 시스템 목표치에서 준 이용률 0.78 ~ 0.85를 만족한다. Hillier 등[3]에 나타난 식을 이용하여 이론치를 구해보면 대기열의 평균 대기 길이가 0.8295이고 평균 대기 시간은 8.295가 된다. 그러나 실제로 구해진 0.216과 2.216은 매우 큰 차이를 보여주고 있으

며, 단지 10개의 개체가 시스템을 빠져 나가면 종료한다. Reverse-Simulation은 온-라인으로 개체의 상태가 변할 때마다 전문가 시스템에서 비교를 하기 때문에 변위 상태를 제거하지 않는 경우에는 매우 빠른 시간에 시스템 목표치를 만족한다. 이런 결과가 나타나는 이유는 시뮬레이션의 확률적(stochastic) 특성에 의해 시스템이 변동이 심하기 때문에 시뮬레이션을 수행하는 초기에도 시스템을 만족하면 즉시 시뮬레이션을 종료하기 때문이다.

그러므로, Reverse-Simulation은 결정 변수들의 값을 고정하여 시뮬레이션을 수행한 후 안정 상태를 결정하는 기존의 시뮬레이션과는 다르게 시뮬레이션을 수행하여 결정 변수들의 값을 변화시키면서, 시스템 목표치의 만족 조건에 따라서 종료되기 때문에 기존의 안정 상태 결정 알고리즘을 사용할 수 없고, 새로운 안정 상태 결정 알고리즘이 필요하다.

Reverse-Simulation에서는 온-라인 즉, 동적으로 전문가 시스템이 시스템의 상태를 관찰하기 때문에 사용되는 알고리즘의 속도(계산량)가 매우 큰 고려 요소가 된다. Kyoung[7]의 논문에서 기존의 안정 상태 결정 알고리즘을 속성에 따라 분류하였다. 다음의 〈Table 4.2〉가 그 결과를 보여준다. Kyoung의 분류에 의하면 온-라인으로 알고리즘을 적용할 수 있는 방법은 MCR 과 CMR 이다. 그러나 MCR과 CMR 방법은 안정 상태를 결정할 때 효율이 좋지 않다는 실험 결과가 발표되었다[7] 그러므로 본 연구에

〈Table 4.2〉 Properties of the algorithms

Method / Property	CR	MCR	CMR	CMR2	WM	LKM	WP
On-line	N	Y	Y	N	N	N	N
Pilot run	Y	N	N	Y	Y	Y	Y
Complexity	Low	Low	Middle	Middle	High	High	Middle

CR: 콘웨이 방법, MCR: 수정 콘웨이 방법, CMR: 평균 교차 방법, CMR2: 누적 평균 방법, WM: Welch 방법, LKM: Law-Kelton 방법, WP: Welch 절차.

서는 안정 상태와 최적 상태의 개념을 도입하여 Wild 와 Pignatiello 의해 제안된 Reverse-Simulation 개념을 Reverse-Simulation 최적화 개념으로 향상키는 방법을 제안한다.

5. 안정 상태와 최적 상태 검출 알고리즘

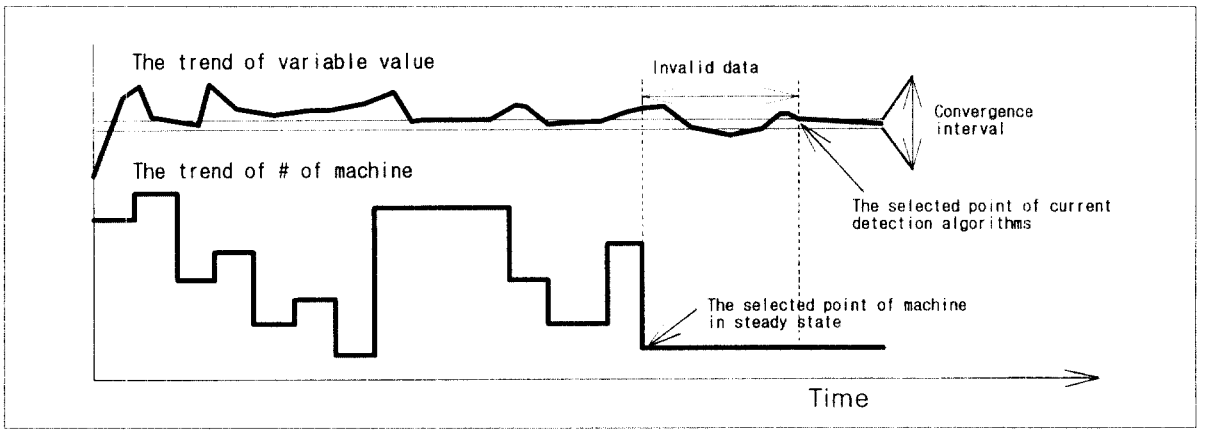
기본 알고리즘을 진행하기 전에 Reverse-Simulation 최적화를 위한 안정 상태 결정 방법과 기존의 시뮬레이션에서의 안정 상태 사이의 차이점을 구분하는 것이 필요하다. 기존의 시뮬레이션에서의 안정 상태는 데이타를 분석하기 위해서 노이즈(noise)가 많은 데이타를 제거하고 나머지 데이타를 분석할 때 사용된다.

Reverse-Simulation 최적화에서의 안정 상태는 안정 상태에서 결정된 변수의 데이타 값을 분석하기 위해서 사용되는 것이 아니고, 목표치들을 만족하는 기계 대수를 찾는 문제이다. 따라서 Reverse-Simulation 최적화에서의 안정 상태를 찾는 알고리즘은 기존의 시뮬레이션에서의 강력한(strong) 알고리즘 뿐만 아니라 노이즈를 어느 정도 가지고 있는 약한(weak) 알고리즘도 사용할 수 있다. 이러한 결과를 그림으로 나타내면 다음의 〈Figure 5.1〉 과 같이 나타낼 수 있다.

〈Figure 5.1〉을 가지고 설명하면, 기존의 시뮬레이션 방법에서의 안정 상태를 검출하는 방법은 변하는 데이타들의 값을 조사하여 이 값이 일정한 영역이나 한 점(point)에 수렴하면 여기서 부터 수집된 데이타를 가지고 분석한다. 데이타 값을 가지고 분석하기 때문에 안정 상태를 검출하는 알고리즘은 매우 강력한 알고리즘을 사용했다.

그러나, Reverse-Simulation 최적화에서는 찾고자 하는 값은 변수들의 결과 데이타 값이 아니고 목표치를 만족하는 시작 상태의 변수 값이다. 〈Figure 5.1〉을 다시 살펴보면, 원하는 기계의 대수가 수렴 구간을 만족하는 변수 값의 상태보다도 빠른 시간에 안정 상태에 들어갈 수가 있기 때문에 기존의 안정 상태 검출 알고리즘을 사용하면 데이타의 손실이 많고, 알고리즘의 효율도 향상되지 않는다. Reverse-Simulation 최적화에서는 데이타들이 어느 정도의 노이즈를 가지고 있어도, 선택하는 기계의 대수가 안정 상태에 들어오는 시작점을 찾으면 된다.

본 논문에서 제안하는 알고리즘의 기본 아이디어는 사용자가 만족하는 수렴 값을 설정한 후에 시스템의 상황에 민감한 결정 변수를 선택(예를 들면, 대기열의 대기 길이나 대기 시간)하여 시스템 목표치를 만족하는 경우에 연속적으로 얻어지는 두 결정 변수의 차이가 수렴 한계치 안에 들어오면 안정 상태로 간주하여, 시뮬레이션을 종료하고 Reverse-Simulation의 입력 값을 결정하는 방법 이다.



〈Figure 5.1〉 The steady state of Reverse-Simulation optimization

이러한 과정을 식으로 나타내면 다음과 같다.

$$|X_i - X_{i-1}| \leq C$$

Where, X_i : *current value*

X_{i-1} : *previous value*

C : *convergence value*

위에서 우리가 고려해야 할 한가지 문제점은 수렴 한계치 C를 결정하는 적절한 방법이 제시되어야 한다. 그러나 변수 X_i의 값의 변위가 이론적으로 $0 \leq X_i \leq \infty$ 사이에서 변하기 때문에 수렴 한계치 C의 설정에는 많은 문제점이 존재한다. 따라서 수렴 한계치 C를 임의적으로 설정하는 문제를 해결하기 위해서 오[1]의 논문에서는 카오스 이론의 지수 변화율을 사용하여 수렴 한계치 C를 자동으로 조절하였으며, 지수 변화율의 수식은 다음과 같이 구성된다.

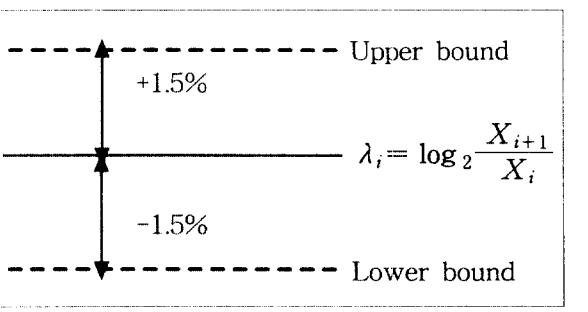
$$\lambda_i = \log_2 \frac{X_{i+1}}{X_i},\ i = 1,2,3,\ldots,\ n\text{-}1$$

X_i : *value of ith output data*

n : *run length*

오의 실험 결과에 의하면 지수 변화율 λ_i가 3%인 경우에 가장 좋은 결과를 얻었다. 따라서 본 논문에서 지수 변화율의 허용 오차를 3%로 정한다. 이것을 그림으로 나타내면 다음의 〈Figure 5.2〉와 같다.

또한 시스템의 상태가 안정 상태이면 시스템의 특성이 임의의 분포에 수렴하기 때문에 지수 변화율이 특정한 경향을 가져서는 안된다. 이런 특징을 배제하기 위해서 오의 논문에서 평균 변화율 3%를 기준 했을 때, 연속적으로 증가하여 현재의 출력 값의 몇 배 이내에 들어오면 좋은 결과를 보여주는 실험을 하였다. 그 결과 2배로 했을 때가 그 이상의 배수로 한 결과보다 우수하므로 본 논문에서는 허용차 범위를 평균 변화율 3%의 2배로 정한다. 따라서 2배가 되는 시점을 p라 하면 다음과 같은 결과를 얻을 수 있다.



〈Figure 5.2〉 The tolerance of λ_i

$$(1+0.015)^p = 2$$

$$p = \frac{\log 2}{\log(1+0.015)} \cong 48$$

위의 계산 결과에서 p의 값이 48이므로 연속적으로 양의 값이나 음의 값을 48번 갖지 않아야 된다는 새로운 제

약식이 추가된다. 또한 오의 논문에서 위의 두 가지 제약식을 사용할 경우에는 연속적으로 데이타가 최대 500번 만족하면 안정 상태로 들어가기 때문에 연속적으로 500번 까지 검사하여 만족하지 못하면 시뮬레이션을 종료하도록 한다. 지금까지의 결과를 기초로 알고리즘을 설명하면 다음과 같다.

단계 0 : λ_i의 허용 오차를 ALL_BET←$\log_2(1\pm0.015)$ 로 한다.
안정상태 결정기준 회수 SS_NUM←1를 정한다.
ALL_NUM←0
POS_NUM←0
NEG_NUM←0
OBJ_NUM←0

단계 1 : 연속적으로 ALL_BET를 만족하면 ALL_NUM = ALL_NUM + 1로 한다.
그렇지 않으면 ALL_NUM←0로 한다.

단계 2 : ALL_BET가 연속적으로 양의 값을 가지면 POS_NUM = POS_NUM+1로 한다.
그렇지 않으면 POS_NUM←0로 한다.
ALL_BET가 연속적으로 음의 값을 가지면 NEG_NUM = NEG_NUM+1로 한다.
그렇지 않으며 NEG_NUM←0로 한다.
POS_NUM == 48 이거나 NEG_NUM == 48 이면,
ALL_NUM←0
POS_NUM←0
NEG_NUM←0 로 한다.
[단계 1]로 이동한다.

단계 3 : ALL_NUM == SS_NUM 이면 안정 상태(steady state) 라고 결정한다.
단계 3-1 : 목표치를 연속적으로 만족하지 못하면, OBJ_NUM←0 로 한다.
SS_NUM = SS_NUM + 49로 한다.
[단계 1]로 간다.
단계 3-2 : 목표치를 연속적으로 만족하면, OBJ_NUM = OBJ_NUM + 1로 한다.

단계 4 : SS_NUM == 540 이면, 안정 상태에서도 목표치를 만족하지 못하므로 결과를 알려주고 종료한다.

단계 5 : OBJ_NUM == 49이면, 최적 상태(optimal state)이며 결과를 알려주고 종료한다.
그렇지 않으면 [단계 1]로 간다.

위의 알고리즘을 실제로 적용하여 10번의 시뮬레이션을 수행한 결과 다음의 〈Table 5.1〉의 결과를 얻었다. 결과를 살펴보면, 위에서 제시한 안정 상태와 최적 상태를 찾는 알고리즘에 따라 주어진 목표치를 만족하고 모든 기계가 1대로 수렴함을 알 수 있다.

〈Table 5.1〉 The results of Reverse-Simulation using the proposed algorithm

Run number	AWTIQ	AWLIQ	Objective value	Selected machine
1	24.827	2.523	0.8000	1
2	15.378	1.589	0.8142	1
3	37.689	3.837	0.8342	1
4	26.503	2.643	0.8000	1
5	19.316	2.021	0.8023	1
6	19.750	2.019	0.8242	1
7	15.696	1.616	0.8139	1
8	19.023	1.921	0.8304	1
9	21.442	2.258	0.8305	1
10	24.324	2.338	0.8241	1

AWTIQ: Average Waiting Time in Queue
AWLIQ: Average Waiting Line in Queue

6. 결론 및 추후 연구 과제

지금까지의 최적화 방법은 사용자의 요구 수준을 고려하지 않은 최적화였다. 그러나 본 연구에서는 사용자의 요구 조건을 충족할 만큼만 최적화를 수행하는 Reverse-Simulation 방법을 소개하였다. 따라서, 이 방법을 사용하면 제조 환경과 소비자의 요구 조건이 급격하게 변할지라도, 제조 환경에서 즉시 대처할 수 있다.

또한, Reverse-Simulation 기법은 단일 런 최적화가 가능하지만, 단일 런 최적화가 가능한 다른 방법을 사용할 때의 가정이 필요 없으며, 적용시 수리 모델 개발에 영향을 받지 않는다. 그러나, 이 방법은 위에서 지적한 것 처럼 정확한 안정 상태를 찾지 못하므로 이러한 안정 상태를

온-라인으로 찾아주는 새로운 기법의 개발이 필요하다.

따라서 본 연구에서는 Reverse-Simulation 개념을 Reverse-Simulation 최적화 개념으로 확장하여 안정 상태 와 최적 상태 개념을 도입하여 목표치가 1개인 M/M/s에 적용하여 좋은 결과를 얻었다. 추후 연구 과제는 목표치가 2개 이상인 경우의 문제에 적용하여 그 효율성을 증명하여 본 연구에서 제공한 개념과 알고리즘을 일반화하는 것이다.

참고문헌

[1] 오 형술, "카오스 이론을 응용한 안정상태 시뮬레이션의 출력분석", 한양대학교 박사학위 논문, 한양대학교, 1996.

[2] Glynn. P. W., "Optimization of Stochastic Systems", *Proceedings of 1986 W. S. C.*, pp. 52-59, 1986.

[3] Hillier, F. S. and Lieberman, G. J., Introduction to Operations Research, 5th ed. McGraw-Hill, 1990.

[4] Ho, Y. C. and X. R. Cao, "Perturbation Analysis of Discrete Event Dynamic Systems", Kluwer Academic Publishers, 1991.

[5] Jacobson, S. H. and L. W. Schruben, "Techniques for Simulation Response Optimization", *Operations Research Letters*, Vol. 8, pp. 1-9, 1988.

[6] Kwanjai, N. N., and Wild, R. H., "Knowledge-Based Simulation to Assist in System Design Identification", *Proceedings of the Winter Simulation Conference*, 1992.

[7] Kyoung, K. H., "On-Line Determination of Steady State in Simulation Output", Master Dissertation, Master of Science, Dept. of I. E., Hanyang Univ., 1996.

[8] M. Hossein Safizadeh, "Optimization in Simulation: Current Issues and the Future Outlook", *Naval Research Logistics*, Vol. 37, pp. 807-825, 1990.

[9] Meketon, M. S., "Optimization in Simulation: A Survey of Recent Results", *Proceedings of 1987 W. S. C.*, pp. 58-67, 1987.

[10] O'Keefe, R., "Simulation and Expert Systems- A Taxonomy and Some Examples", *Simulation*, Vol. 46, pp. 10-16, 1986.

[11] Pritsker, A. A. B., Introduction to Simulation and SLAM II, John Wiley & Sons, 1986.

[12] Rubinstein, R. Y. and Shapiro, A Discrete Event Systems, John Wiley & Sons, 1993.

[13] Wild, Rosemary H. and He, Jackson, "An Architecture to Support Reverse Simulation", *Proceedings of the 1994 S. C. S. C.*, 1994.

[14] Wild, Rosemary H. and Pignatiello, Joseph J. Jr., "An Expert System Based Reverse Simulation Techinque", *Proceedings of the 1991 S. C. S. C.*, 1991.

[15] Wild, Rosemary H. and Pignatiello, Joseph J. Jr., "Finding Stable System Designs: A Reverse Simulation Technique", *Communications of the ACM*, Vol. 35, No. 10, pp. 87-98, 1994.

● 저자소개 ●

이영해

1997년 고려대학교 산업공학 학사

1983년 미국 Univ. of Illinois, 산업공학 석사

1986년 미국 Univ. of Illinois, 산업공학 박사

일본 오사카대학 전자제어기계공학과 객원교수(1990년)

현재 한국시뮬레이션학회 부회장

한양대학교 산업공학과 교수

관심분야 : Simulation in Manufacturing, simulation Output Analysis, simulation Optimization

박경종
1992년 한양대학교 산업공학 학사
1994년 한양대학교 산업공학 석사
1996년 현재, 한양대학교 산업공학과 박사과정
관심분야 : Simulation Optimization, Simulation and Artificial Intelligence