이산형 변수 시스템의 설계를 위한 시뮬레이션 활용 기법 연구*

A Method for Design of Discrete Variable Stochastic Systems using Simulation

박경종**, 이영해***
Kyoung-Jong Park, Young Hae Lee

Abstract

This paper deals with a discrete simulation optimization method for designing a complex probabilistic discrete event system. The proposed algorithm in this paper searches the effective and reliable alternatives satisfying the target values of the system to be designed through a single run in a relatively short time period. It tries to estimate an autoregressive model, and construct mean and confidence interval for evaluating correctly the objective function obtained by small amount of output data. The experimental results using the proposed method are also shown.

* 이 논문은 1999년 한양대학교 교내연구비에 의하여 연구되었음.
** 프라이스워터하우스쿠퍼스
*** 한양대학교 산업공학과

1. 서론

제조업이 발달함에 따라 시스템이 대형화, 복잡화 되고 있어서 이러한 시스템들은 간단한 분석적인 방법이나 비현실적인 가정들을 가진 해석적인 방법으로 해결할 수 없는 경우가 많이 발생한다. 예를 들어, 대기행렬 네트워크로 모델링 될 수 있는 복잡한 유연제조시스템에서 부품들의 도착간 시간이나 기계들의 서비스 시간이 임의의 분포를 따르거나, 가공제품의 종류가 많고, 버퍼 크기가 유한하며, 복잡한 우선 순위 규칙 등을 가진 경우에는 기존에 개발되어 있는 해석적인 방법으로는 해결하기가 매우 어렵기 때문에, 시뮬레이션 기법을 사용해야 한다.

또한, 복잡한 확률적 이산 사건 시스템(예를 들면, 작업장에서의 적정 작업자 선정 문제와 (s, S) 재고 관리 시스템에서 발주량 및 재발주량 선정 문제 등)의 설계 문제는 시스템 요소의 일부 또는 전부가 확률적이고 복잡하므로, 목적함수를 분석적으로 알려진 함수로 표현하기가 어렵다. 여러 분야의 복잡한 시스템 모델링 과정을 고찰해 볼 때, 정확한 분석적 표현이 가능한 시스템은 찾기 힘들며, 분석적인 표현이 가능한 경우에도 대부분은 비선형이거나 확률적이기 때문에 이산 사건(discrete event) 시뮬레이션을 이용하여 복잡한 시스템을 연구하는 것이 가장 효과적인 수단으로 인식되고 있다.

시뮬레이션 모델링은 제조 설비와 같은 복잡한 확률적(stochastic) 이산 사건 시스템을 분석하고, 수행척도 등을 계산하기 위해 광범위하게 사용되고 있다. 그러나, 시뮬레이션 모델을 사용하여 시스템 설계자가 필요로 하는 대안(가능해 또는 최적해)을 찾기 위해 사용할 수 있는 대부분의 기존 방법들은 계산량이 많이 요구되는 시뮬레이션의 반복 수행을 필요로 한다.

복잡한 확률적 이산 사건 시스템의 설계 도구로서 이산 사건 시뮬레이션을 사용하기 위해서는 극복해야 할 몇 가지 문제가 있다. 첫째, 시스템의 성능을 평가할 목적함수와 일부 제약식의 값은 알려진 함수를 이용한 단순한 계산이 아니라 오직 시뮬레이션을 수행하여서만 얻을 수 있다. 둘째, 시뮬레이션을 수행하여 얻은 결과는 확률적 이산 사건 시스템의 경우 반드시 확률적 요소가 포함되어 있기 때문에 효과적인 통계적 분석이 요구되고, 최적화 문제도 결정적(deterministic) 최적화가 아닌 확률적 최적화 문제가 된다. 셋째, 수행하는 시뮬레이션의 형태가 안정상태(steady state) 시뮬레이션인 경우, 하나의 대안에 대하여 안정상태에서의 시스템의 평가척도를 구하는 데는 일반적으로 출력 데이터들이 서로 자동상관(auto-correlation)을 포함하여 이를 해결하는 데 많은 출력 데이터들이 필요하게 되어 실행시간이 많이 걸리게 된다. 안정 상태란 연속형 시뮬레이션에서 사용되는 용어로서 시뮬레이션에서 시간이 흘러도 데이터들이 같은 분포를 가지는 경우를 의미한다. 넷째, 시뮬레이션을 이용하여 하나의 대안을 평가하는 데도 많은 시간이 걸리는데, 해 발견을 위한 탐색공간이 클 경우에는 더욱 많은 시간이 걸리게 된다(Azadivar 1992).

단계 1:	시스템의 반응치를 정의하고, 각각의 시스템 설계 변수들에 대해서 값을 할당한 대안을 결정한다.
단계 2:	규정된 시스템 환경에 대해서 여러 번의 독립적인 반복 시뮬레이션을 수행하고, 출력 데이터를 수집한다.
단계 3:	얻은 출력 데이터를 분석한다. 만약, 출력 데이터가 만족스러우면, 단계 4로 가고, 출력 데이터가 만족스럽지 못하면, 시스템 설계 변수들의 값을 바꾸어서 단계 2로 다시 간다.
단계 4:	최상(the best)의 시스템 설계안을 선택하고 종료한다.

시뮬레이션을 이용하여 이산 사건 시스템을 설계할 때 일반적인 설계자가 따르는 순서는 위와 같다.

위에서 설명한 단계에서 가장 어려운 부분은 단계 2와 단계 3의 반복이다. 시스템의 설계 변수들이 많아지면, 각각의 설계 변수들의 값을 할당하고 시뮬레이션을 반복 수행하여, 출력 데이터를 분석한 후, 다시 설계 변수들의 값을 조정하고 시뮬레이션을 반복하는 과정은 매우 복잡하고 시간이 많이 소요된다. 탐색 가능한 대안의 수가 작을 경우에는 모든 대안을 검사하는 방법(enumeration)이 가능하지만, 그렇지 못할 경우에는 효과적인 대책이 없다.

그러므로, 본 연구에서는 위에서 설명한 문제점을 해결하기 위해 이산 사건 시뮬레이션 기법을 이용하여 이산형 변수를 가진 복잡한 확률적 이산 사건 시스템을 설계할 경우, 요구되는 시스템 평가척도와 제약식을 충족하며 신뢰할 수 있는 대안을 효율적으로 빠른 시간 내에 온-라인으로 발견해 내는 문제를 다룬다. 이를 위해 필요한 탐색 알고리즘을 개발하고, 짧은 시뮬레이션의 수행으로 얻은 데이터를 이용하여 안정상태의 반응치를 추정하는 기법을 개발한다.

2. 기존 연구 고찰

본 연구에서 다루는 문제는 넓게는 시뮬레이션 최적화(simulation optimization)의 범주에 포함되지만, 엄밀하게 분류하면 최적화 문제라기보다는 시뮬레이션 출력분석에 의해 평가할 수 있는 주어진 목적함수의 목표치(일정한 값이나 구간으로 주어짐)를 만족하는 적절한 대안을 발견하는 문제이다. 여기에서 개발된 방법은 필요하면 최적화를 수행하여 좋은 해(때로는 최적해)를 얻거나, 최적화를 위한 좋은 초기해를 발견하는 목적을 위해 사용될 수 있다. 따라서, 본 절에서는 일반적인 시뮬레이션 최적화 분야에서의 기존 연구들을 살펴보고자 한다.

시뮬레이션 최적화 문제는 Glynn(1986), Meketon(1987), Jacobson and Schruben(1989), Safizadeh(1990), Ho and Cao(1991), 그리고 Rubinstein and Shapiro(1992) 등에 의해 지속적인 연구와 발전이 이루어져 왔다. 일반적으로 많이 사용되는 Finite Differences를 이용한 방법에서는 변화율(gradient)를 추정하기 위해서 m개의 파라메터에 대해 최소한 m+1번의 시뮬레이션 반복이 필요하다는 단점을 가지고 있다(Heidergott 1995). 따라서 시뮬레이션 최적화를 수행할 때 발생하는 많은 반복 수행의 문제를 해결하기 위해서는 단일 실행(single run)으로 최적화가 가능한 방법들을 필요로 한다. 단일 실행으로 최적화를 수행하기 위해 Perturbation Analysis(PA) 방법과 Score Function (SF) 방법이 제안되었지만, 이 방법들은 모두 다 연속형 결정변수를 대상으로 하고 있으며, 진행 방향이 정수형으로 결정되어야 하는 이산형 결정변수에는 아직까지 적용되지 못하고 있다.

또한, 현재까지의 시뮬레이션 최적화에 대한 연구는 연속형 결정변수에 대하여 주로 발전되었고, 이산형 결정변수에 대한 시뮬레이션 최적화 연구는 상대적으로 연구가 많이 진행되지 못했다. 그러나, 시스템의 버퍼 최적화나 자원 최적화 등에서 많은 관심과 필요성이 증대되어 최근에 와서 관심이 높아지고 있다. 시뮬레이션 최적화 문제에서 이산형 결정변수를 가진 이산형 확률적 최적화(discrete stochastic optimization) 문제를 풀기 위해서 simulated annealing(Lee and Iwata 1991, Ahmed et al. 1997), stochastic ruler method(Yan and Mukai 1992), stochastic comparison method(Gong et al. 1992), random walk(Andradottir 1991, 1995, 1996), nested partitions method(Shi and Sigurdur 1997), evolutionary(genetic) algorithm(Pierreval and Tautou 1997), multi-armed bandit method(Barry and Fristedt 1985), learning automata(Yakowitz and Lugosi 1990) 등의 방법들이 사용되었으며, 이

방법들은 시뮬레이션 과정을 반복 수행하면서 최적화를 수행한다.

기존 연구들 중에서 특히 Chen(1994)은 $g(x)=\gamma$ 라고 표현되는 stochastic root finding 문제에 대해 Monte Carlo sampling을 이용한 알고리즘과 retrospective approximation(Fu and Healy 1997) 방법을 이용한 알고리즘을 제시하였다. Chen이 제시한 방법은 매우 많은 시뮬레이션의 반복 실행을 요구하므로 간단한 수치 시뮬레이션을 목적함수의 평가 도구로 이용하고, 목적함수가 $g(x)=\gamma$ 라고 표현되는 경우에는 이용이 가능하나, 복잡하고 한 번의 시뮬레이션 수행동안 긴 실행시간을 요구하는 안정상태 이산사건 시뮬레이션에는 적합하지 않다.

Wild and Pignatiello(1994)는 안정상태 이산사건 시뮬레이션을 이용하여 시스템을 설계할 때 주어진 목표치($g(x) \leq \gamma$의 형태로 표시)를 만족하는 대안을 발견하기 위해 시뮬레이션 도중에 결정변수 값을 변경해 가는 reverse-simulation이라고 부르는 방법을 제시하였다. 그러나, 이 방법은 결정변수 값의 변경에 대해 시스템 목적함수 값의 변화의 방향을 알 수 있는 M/M/s로 모델링 될 수 있는 아주 간단한 경우에만 적용이 가능하고, 각 개체가 시스템을 떠날 때마다 결정 변수 값을 변경하므로 안정되고 신뢰할 수 있는 대안을 얻기가 어렵고, 다양한 형태의 목표치가 주어진 경우에 사용하기가 불가능하다. 또한, 지금까지의 연구들과 개발된 방법들은 이산 사건 시뮬레이션 기법을 이용한 이산 사건 시스템의 최적 설계시 반드시 고려해야 할 긴 시뮬레이션 수행 시간을 간과한 경우가 많았고, 단지 시뮬레이션을 기초로 한 확률적 최적화 알고리즘에 국한하는 경우가 많았다.

본 연구에서는 이산 사건 시뮬레이션 기법을 이용하여 이산형 변수를 가진 복잡한 확률적 이산 사건 시스템을 설계할 경우, 단일 실행으로 빠른 시간내에 요구되는 시스템 평가척도와 주어진 시스템의 목표치를 충족하는 대안을 발견해내는 문제를 대상으로 한다.

본 연구의 제 1 장에서는 시뮬레이션 기법의 일반적인 문제점들을 다루었고 제 2 장에서는 시뮬레이션 최적화 기법의 기존 연구 고찰에 대하여 설명하였다.

제 3 장에서는 해 탐색을 위한 알고리즘을 설명하는데, 알고리즘에 대한 기본적인 설명과 알고리즘의 핵심이 되는 결정변수 값의 조정을 위한 세부 알고리즘 및 알고리즘의 종료를 위한 조건들을 기술하고 개발된 알고리즘의 구현과 구조에 관하여 논한다.

또한, 안정상태 시뮬레이션을 수행할 때 짧은 시뮬레이션 수행으로 얻은 적은 수의 출력 데이터로 주어진 목적함수의 평가를 정확하게 할 수 있는 방법에 대하여 설명한다. 이를 위해 적은 수의 출력 데이터로 모델링 할 수 있는 자동회귀 모델을 사용한다. 제 4 장에서는 지금까지 소개된 방법의 유용성과 효율성을 평가하기 (s, S) 재고 모델을 이용하여 실험 및 분석을 행한다. 마지막으로, 제 5 장에서는 본 연구의 결과를 설명하고 추후 연구 방향을 제시한다.

3. 해 탐색 알고리즘

제조 시스템 분야에서 제기되는 문제의 시스템 설계 대안을 단일 시뮬레이션 실행(single simulation run) 환경에서 찾기 위한 방법은 [알고리즘 3.1]과 같이 정리된다.

본 연구에서는 결정변수(x_i)가 이산형인 경우를 가정하므로, 각 변수에 대한 Δx_i는 각 변수의 최소 증감값(resolution)을 나타내는데, 정수값을 갖는 경우는 일반적으로 1을 가지나 탐색공간이 큰 경우에는 사용자가 크기를 적절하게 조정할 수 있다.

목적함수를 평가하고 결정변수 값을 조정할 시간 간격(Δt)의 단위는 바로 전 단계에서 결정변수 값의 조정 행위가 일어난 뒤로부터의 처리 회수, 또는 시뮬레이션에서 처리된 사건(event) 수 등으로 설정이 가능하다. 예를

들어, Δt 가 10 인 경우는 10 번의 트랜잭션(transaction)에 대한 정보를 수집하여 다음 단계로 진행할 방향을 결정한다는 의미이다. 수행하는 시뮬레이션의 형태가 종료(terminating) 시뮬레이션인 경우는 Δt 가 수행하는 시뮬레이션의 총 수행 길이가 될 수 있다.

[알고리즘 3.1]

단계 1:	목적함수($f_i(X)$), 결정변수(x_j, j=1, ..., n), 목표치(A_i), 결정변수 x_j, j=1, ..., n 에 대한 증감(Δx_j), 목적함수를 평가하고 결정변수 값을 조정할 시간 간격(Δt)을 설정하고, 시뮬레이션을 시작한다.
단계 2:	시뮬레이션 도중 Δt 마다 목적함수와 목표치를 비교하여 결과에 따라 바람직한 방향으로 Δx_j 만큼 결정변수 값을 조정하고, 시뮬레이션을 계속한다(알고리즘 3.2 참조).
단계 3:	알고리즘 종료 조건을 검사하고 만족하면, 최근 시점까지 가장 많이 방문했던 결정변수 값들의 조합을 최종해 x_j^*, j=1, ..., n 로 정하고 시뮬레이션을 종료한다(알고리즘 3.3 참조).
단계 4:	얻어진 최종해를 가지고 충분한 길이의 검증 시뮬레이션을 수행하고 필요한 자료를 출력한다

그러나, 안정상태 시뮬레이션인 경우는 주어진 변수 값들의 조합인 하나의 대안에 대하여 안정상태에서의 출력을 얻을 때까지는 상당히 긴 시뮬레이션의 수행 길이가 필요하므로, 요구되는 최종해를 얻을 때까지 필요한 여러 대안들의 평가에 걸리는 총시간은 컴퓨터로 주어진 시간 내에 처리할 수 없는 매우 긴 시간이 될 수 있다. 따라서, 이를 줄이기 위한 노력이 절대적으로 필요하다.

일반적으로 Δt 가 작을수록 자주 목적함수를 평가하고 결정변수 값을 조정하므로 빨리 목표치를 만족하는 결정변수 값으로 수렴할 확률이 높을 수도 있겠으나, Δt 가 작은 만큼 얻은 데이터의 수가 적어 목적함수 값을 평가하는 데 오차가 커질 수 있다. 그리고, 목적함수를 평가하고 결정변수 값을 조정하는데, 많은 계산 시간을 소비하게 된다. 이를 위해 짧은 시뮬레이션의 수행으로 안정상태에서의 목적함수 값을 효율적으로 추정할 수 있는 방법을 제안한다.

앞의 알고리즘 3.1 의 단계 2 에서, 시뮬레이션 도중 Δt 마다 목적함수와 목표치를 비교하여 결과에 따라 바람직한 방향으로 Δx_j 만큼 결정변수 x_j 의 값을 조정한다고 하였는데, 이에 대한 설명이 다음의 제 3.1 절에서 자세히 기술한다.

3.1 결정변수 조정

알고리즘 3.1 의 단계 2 에서, 시뮬레이션 도중 Δt 마다 목적함수와 목표치를 비교하여 결과에 따라 바람직한 방향으로 Δx_j 만큼 결정변수 x_j 의 값을 조정한다고 하였는데, x_j 의 값의 증감에 따라 목적함수 값의 변화가 쉽게 파악 되는 경우와 그렇지 않는 경우로 나눌 수 있다.

전자의 경우의 예로 단일 작업장 문제에서는 기계 대수(결정변수)의 증가(감소)는 작업물의 평균 대기시간(목적함수)을 감소(증가)시키며, Job Shop 문제에서는 작업장 i 에 있는 기계 대수의 증가(감소)는 버퍼에서 대기하는 부품의 수를 감소(증가)시키며, 기계의 평균 이용률의 감소(증가)를 초래한다. 또한, 작업장의 수리공 문제에서 가능한 기계 대수의 증가(감소)는 기계의 평균 고장상태 시간의 증가(감소)를 초래하고, 수리공 수의 증가(감소)는 기계의 평균 고장상태 시간의 감소(증가)를 초래한다.

후자의 경우와 같이 결정변수 값의 증감에 따라 목적함수 값의 변화가 바로 감지되지 않는 경우의 예로 (s,S) 재고관리시스템에서 결정변수인 (s,S) 값의 증감이 목적함수인 단

위 기간당 발생하는 평균 재고관리 비용의 변화가 예측되지 않는데, 이러한 경우도 알고리즘에서 고려하는 것이 필요하다.

시뮬레이션 도중 Δt 마다 목적함수와 목표치를 비교하여 결과에 따라 결정변수 x_j의 값을 조정할 방법의 기본적인 설명은 다음과 같다.

각 변수에 대하여 시뮬레이션 출력으로 얻는 목적함수의 값이 단조 증가 함수(monotonic increasing function) 또는 단조 감소 함수(monotonic decreasing function) 인지를 파악하여 다음의 조건에 따라 방향을 결정한다. 즉, 1) 주어진 A_i의 형태 [{ $f_i = c$ }, { $f_i > c$ }, 또는 { $f_i < c$ }], 2) t-1 시점과 t 시점의 결정변수 j의 값 $x_{j,t-1}$ 와 $x_{j,t}$, 3) t-1 시점과 t 시점에서 시뮬레이션 모델로부터 얻어진 목적함수 i의 값 $y_{i,t-1}$ 와 $y_{i,t}$, 4) t 시점에서 시뮬레이션 모델로부터 얻어진 목적함수 i의 값 y_i와 주어진 목표치의 값 c의 관계에 따라 결정한다. 각 변수 값의 증가, 감소, 그리고 불변의 횟수를 누적하여 가장 많은 횟수를 기록한 방향으로 각 변수 값을 Δx_j 만큼씩 변화를 시킨다. 이로 인해 변수 값의 변화 방향이 목적함수에 따라 충돌(conflict)이 발생하는 경우도 동시에 고려할 수 있다. 결정변수 값의 조정을 위해 사용되는 용어에 대한 설명과 알고리즘은 다음과 같다.

$x_{j,t}$: 시뮬레이션이 행해지는 시점 t에서의 결정변수 j의 값

$y_{i,t}$: 시뮬레이션이 행해지는 시점 t에서의 목적함수 i의 시뮬레이션 출력값

$count_j^o$: 결정변수 j에 대하여 $x_{j,t-1}$ 와 $x_{j,t}$ 의 변화를 관찰할 때, y 값이 변하지 않을 때의 횟수

$count_j^+$: 결정변수 j 에 대하여 $x_{j,t-1}$ 와 $x_{j,t}$ 의 변화를 관찰할 때, y 값이 증가할 때의 횟수

$count_j^-$: 결정변수 j 에 대하여 $x_{j,t-1}$ 에서 $x_{j,t}$ 의 변화를 관찰할 때, y 값이 감소할 때의 횟수

a_j 와 b_j: 결정변수 j의 하한치와 상한치

L_i^{++} : 변수 x_j 값을 증가(감소)시키면 목적함수 i 값이 증가(감소)하는 변수들의 집합

L_i^{+-} : 변수 x_j 값을 증가(감소)시키면 목적함수 i 값이 감소(증가)하는 변수들의 집합

L_i^o : 목적함수 i에 대하여 L_i^{++} 나 L_i^{+-} 에 포함되지 않는 변수들의 집합

시스템을 평가할 때는 본 연구에서 처럼 목적함수의 방향만 평가하는 경우와 방향과 크기를 동시에 고려할 수 있다. 그러나 확률적 시뮬레이션 모델에서 방향과 크기를 같이 고려하면 시스템이 확률적이기 때문에 변화 크기도 고려되는 시스템에 매우 의존적이기 때문에 방향과 크기를 동시에 고려하는 것은 무의미하다.

3.2 종료 알고리즘

목적식이나 대안을 만족할 때 임의의 시점에서 시뮬레이션을 종료해야 하는데, 각 시점 t에서 수행하게 될 알고리즘의 종료조건 검사를 위해 필요한 용어 및 알고리즘을 설명하면 다음과 같다.

x_j^* : 현시점 t에서 최근 K 번의 시점에서 결정된 변수의 값들 중에서 빈도수가 가장 높은 값

K : 종료 조건 검사를 위해 기준이 될 최근의 시점 수(K = 1, 2, ..., n)

η : 종료 조건을 위해 결정변수 x_j, j=1, ..., n 에 대하여 허용치($x_j^* \pm \eta \Delta x_j$) 를 산정하는데 사용되는 정수(η = 0, 1, 2, ..., m)

[알고리즘 3.3]

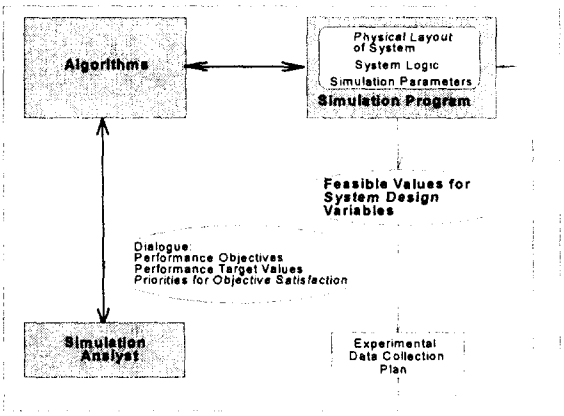
단계 0:	정수인 K, η 값을 부여한다.
단계 1:	결정변수 x_j, j=1, ..., n 에 대하여 다음 과정을 수행하고, 모든 결정변수에 대하여 만족하면 단계 2 로 간다. (알고리즘 3.1)에 의해 x_j^*를 구하고, 최근 K 시점들에서 결정변수 x_j, j=1, ..., n 의 모든 값이 $[x_j^* \pm \eta \Delta x_j]$ 범위에 들어오면, 단계 2 로 간다. 그렇지 않으면, 본 알고리즘을 중단하고 다음 시점 t+Δt 까지 시뮬레이션을 계속한다.
단계 2:	목표치 i 에 대하여 만족 여부를 확인하여, 만족하면 시뮬레이션을 종료한다. 그렇지 않은 경우, η = 0 이면, 본 알고리즘 및 시뮬레이션을 중단하고, '목표치를 만족하는 대안이 존재하지 않음'으로 결론을 내리고, η > 0 이면, η 를 η = max[0, η - 1]로 수정한 뒤 다음 시점 t+Δt 까지 시뮬레이션을 계속한다.

단계 0 에서의 K 값(K = 1, 2, ..., n)은 시뮬레이션을 통해 얻는 시스템 수행도의 확률성의 크기에 종속되며, 시스템 설계자에게 주어진 컴퓨터 종류와 할당된 시간을 고려하여 시스템 설계자가 결정해야 한다. η 값(η = 0, 1, ..., m)은 정의된 문제 및 변수의 성질에 따라 달라져야 하는데, 보통 0 또는 1 의 값이면 적당하다.

단계 2 에서는 모든 결정변수 값들이 종료를 위한 허용 범위 내에 포함되어도 목표치를 만족하지 못하면 허용 범위를 줄인다. 더 이상 허용치가 없는 경우(η = 0)는 '목표치를 만족하는 대안이 존재하지 않음'으로 결론을 내리고, 알고리즘 및 시뮬레이션을 중단하고, 허용치가 있는 경우(η > 0)는 시뮬레이션을 계속 수행함을 의미한다. 허용 범위가 넓으면 결정변수의 변화 폭이 그만큼 크다는 것을 내포한다. 따라서, 허용 범위가 클수록 빨리 수렴하지만 비교하는 목표치의 변화 폭이 커질 수가 있기 때문에 목표치가 만족되지 못하면 허용치를 줄여서 다시 검사한다.

3.3 알고리즘 및 시뮬레이션 모델의 연결

지금까지 제 3 장에서 설명한 알고리즘의 구현은 시뮬레이션 프로그램과 알고리즘을 연결함으로써 구성된다. <그림 1>은 본 연구에서 사용되는 시뮬레이션 프로그램과 제안된 알고리즘의 연결 방법이다(Wild and Pignatiello 1994).



<그림 1> Structure of the suggested method

<그림 1>에서 시뮬레이션 분석가가 분석할 때 필요한 목적함수, 목표 값, 그리고 목적함수를 만족시키기 위한 우선순위 등에 관한 정보를 제공하여 알고리즘을 구성한다. 시뮬레이션 프로그램에는 구성되는 시스템의 물리적인 배치, 시스템의 진행 로직, 시뮬레이션 파라메터 등이 포함된다.

단계 3:	목적함수 i 에 대하여, 다음의 case 중의 하나를 수행한다.	
	(case 1) $A_i = \{f_i = c\}$	
	단계 3.1.1:	$x_j \in L_i^{\circ}$ 이면, 단계 3.1.3 을 행하고, 아니면 단계 3.1.2 를 수행한다.
	단계 3.1.2:	$y_{i,t} = c$ 이면, $count_j^{\circ} = count_j^{\circ} + 1$ $y_{i,t} < c$ 이고, $x_j \in L_i^{++}$ 이면, $count_j^{+} = count_j^{+} + 1$ $y_{i,t} < c$ 이고, $x_j \in L_i^{+-}$ 이면, $count_j^{-} = count_j^{-} + 1$ $y_{i,t} > c$ 이고, $x_j \in L_i^{++}$ 이면, $count_j^{-} = count_j^{-} + 1$ $y_{i,t} > c$ 이고, $x_j \in L_i^{+-}$ 이면, $count_j^{+} = count_j^{+} + 1$
	단계 3.1.3:	$y_{i,t} = c$ 이면, $count_j^{\circ} = count_j^{\circ} + 1$ $y_{i,t-1} \le y_{i,t}$ 이고, $y_{i,t} < c$ 이면, $count_j^{+} = count_j^{+} + 1$ $y_{i,t-1} \le y_{i,t}$ 이고, $y_{i,t} > c$ 이면, $count_j^{-} = count_j^{-} + 1$ $y_{i,t-1} > y_{i,t}$ 이고, $y_{i,t} < c$ 이면, $count_j^{-} = count_j^{-} + 1$ $y_{i,t-1} > y_{i,t}$ 이고, $y_{i,t} > c$ 이면, $count_j^{+} = count_j^{+} + 1$
	(case 2) $A_i = \{f_i < c\}$	
	단계 3.2.1:	$x_j \in L_i^{\circ}$ 이면, 단계 3.2.3 을 행하고, 아니면 단계 3.2.2 를 수행한다.
	단계 3.2.2:	$y_{i,t} < c$ 이면, $count_j^{\circ} = count_j^{\circ} + 1$ $y_{i,t} \ge c$ 이고, $x_j \in L_i^{++}$ 이면, $count_j^{-} = count_j^{-} + 1$ $y_{i,t} \ge c$ 이고, $x_j \in L_i^{+-}$ 이면, $count_j^{+} = count_j^{+} + 1$
	단계 3.2.3:	$y_{i,t} < c$ 이면, $count_j^{\circ} = count_j^{\circ} + 1$ $y_{i,t-1} \le y_{i,t}$ 이고, $y_{i,t} \ge c$ 이면 $count_j^{+} = count_j^{+} + 1$ $y_{i,t-1} > y_{i,t}$ 이고, $y_{i,t} \ge c$ 이면 $count_j^{-} = count_j^{-} + 1$
	(case 3) $A_i = \{f_i > c\}$	
	단계 3.3.1:	$x_j \in L_i^{\circ}$ 이면, 단계 3.3.3 을 행하고, 아니면 단계 3.3.2 를 수행한다.
	단계 3.3.2:	$y_{i,t} > c$ 이면, $count_j^{\circ} = count_j^{\circ} + 1$ $y_{i,t} \le c$ 이고, $x_j \in L_i^{++}$ 이면, $count_j^{+} = count_j^{+} + 1$ $y_{i,t} \le c$ 이고, $x_j \in L_i^{+-}$ 이면, $count_j^{-} = count_j^{-} + 1$
	단계 3.3.3:	$y_{i,t} > c$ 이면, $count_j^{\circ} = count_j^{\circ} + 1$ $y_{i,t-1} \le y_{i,t}$ 이고, $y_{i,t} \le c$ 이면 $count_j^{-} = count_j^{-} + 1$ $y_{i,t-1} > y_{i,t}$ 이고, $y_{i,t} \le c$ 이면 $count_j^{+} = count_j^{+} + 1$

단계 4:	목적함수 i에 대하여, 다음의 case 중의 하나를 수행한다.	
	(case 1) $A_i = \{f_i = c\}$	
	단계 4.1.1:	$\mid y_{i,t} - c \mid \le \varepsilon$ 이면, $count_j^{\circ} = count_j^{\circ} + 1$ $\mid y_{i,t} - c \mid > \varepsilon$ 이고, $x_j \in L_i^{\circ}$ 이면, 단계 4.1.3 을 수행하고, 아니면 단계 4.1.2 를 수행한다.
	단계 4.1.2:	$x_j \in L_i^{++}$ 이면, $count_j^{-} = count_j^{-} + 1$ $x_j \in L_i^{+-}$ 이면, $count_j^{+} = count_j^{+} + 1$
	단계 4.1.3:	확률 0.5 로 $count_j^{+} = count_j^{+} + 1$, 또는 확률 0.5 로 $count_j^{-} = count_j^{-} + 1$
	(case 2) $A_i = \{f_i < c\}$	
	단계 4.2.1:	$x_j \in L_i^{\circ}$ 이면, 단계 4.2.3 을 행하고, 아니면 단계 4.2.2 를 수행한다.
	단계 4.2.2:	$y_{i,t} < c$ 이면, $count_j^{\circ} = count_j^{\circ} + 1$ $y_{i,t} \ge c$ 이고, $x_j \in L_i^{++}$ 이면, $count_j^{-} = count_j^{-} + 1$ $y_{i,t} \ge c$ 이고, $x_j \in L_i^{+-}$ 이면, $count_j^{+} = count_j^{+} + 1$
	단계 4.2.3:	$y_{i,t} < c$ 이면, $count_j^{\circ} = count_j^{\circ} + 1$ $y_{i,t-1} \le y_{i,t}$ 이고, $y_{i,t} \ge c$ 이면 $count_j^{+} = count_j^{+} + 1$ $y_{i,t-1} > y_{i,t}$ 이고, $y_{i,t} \ge c$ 이면 $count_j^{-} = count_j^{-} + 1$
	(case 3) $A_i = \{f_i > c\}$	
	단계 4.3.1:	$x_j \in L_i^{\circ}$ 이면, 단계 4.3.3 을 행하고, 아니면 단계 4.3.2 를 수행한다.
	단계 4.3.2:	$y_{i,t} > c$ 이면, $count_j^{\circ} = count_j^{\circ} + 1$ $y_{i,t} \le c$ 이고, $x_j \in L_i^{++}$ 이면, $count_j^{+} = count_j^{+} + 1$ $y_{i,t} \le c$ 이고, $x_j \in L_i^{+-}$ 이면, $count_j^{-} = count_j^{-} + 1$
	단계 4.3.3:	$y_{i,t} > c$ 이면, $count_j^{\circ} = count_j^{\circ} + 1$ $y_{i,t-1} \le y_{i,t}$ 이고, $y_{i,t} \le c$ 이면 $count_j^{-} = count_j^{-} + 1$ $y_{i,t-1} > y_{i,t}$ 이고, $y_{i,t} \le c$ 이면 $count_j^{+} = count_j^{+} + 1$
단계 5:	결정변수 x_j, j=1, ..., n 에 대하여 다음을 수행한다. $count^* = \max [count_j^{\circ}, count_j^{+}, count_j^{-}]$ (동률일 경우는 랜덤 선택) $count^* = count_j^{+}$ 이면, $x_{j,t+1} = \min[(x_{j,t} + \Delta x_j), b_j]$ $count^* = count_j^{-}$ 이면, $x_{j,t+1} = \max[a_j, (x_{j,t} - \Delta x_j)]$ $count^* = count_j^{\circ}$ 이면, $x_{j,t+1} = x_{j,t}$	

시뮬레이션 분석가와 알고리즘에 의해 제공된 정보를 가지고 시뮬레이션 프로그램에서 시뮬레이션을 수행하면서 시스템 설계 변수들에 대해서 가능한 해들을 찾아내서 실험 계획 및 분석을 수행하여 적절한 대안을 찾아내는 과정을 반복한다.

구성된 알고리즘은 시뮬레이션 모델과 분리하여 연결하는데, 이러한 이유는 시뮬레이

션 모델과 시뮬레이션 프로그램을 모르는 사용자가 알고리즘만을 구성하고 시뮬레이션 모델 구성자가 시뮬레이션 모델을 구성하여 인터페이스 표준만을 같게 만들면 빠른 시간에 많은 작업을 수행하기 때문에 추천되는 구성 방식이다.

위에서 제시된 모델은 펜티엄 60MHZ 의 환경과 Windows 95 를 기반으로 시뮬레이션 툴은 Awesim!을 사용하였고 알고리즘의 인터페이스를 위해 VC++ 4.0 을 이용해 코딩한다.

3.4 안정상태 목적함수의 평가

앞에서 기술한 알고리즘 3.1 에서 Δt 가 작을수록 자주 목적함수를 평가하고 결정변수 값을 조정하므로 빨리 목표치를 만족하는 결정변수 값으로 수렴할 확률이 높을 수도 있겠으나, Δt 가 작은 만큼 얻은 데이터의 수가 적어 목적함수 값을 평가하는 데 시뮬레이션 모델의 확률적 성질에 의해 오차가 커질 수 있다. 이를 위해서 비교적 짧은 시뮬레이션의 수행으로 얻은 적은 수량의 데이터로 안정상태에서의 목적함수 값을 효율적으로 추정할 수 있는 방법을 기술한다.

앞에서 지적한 바와 같이, 시뮬레이션 출력의 초기 편의(initial bias)의 영향을 받지 않는 안정상태에서 추정치를 얻기 위해서는 일반적으로 매우 긴 시간동안 시뮬레이션의 수행이 필요한 데, 초기 편의를 줄이기 위해 많은 시뮬레이션 연구자들이 연구를 해왔다(Law and Kelton 1995).

Voss et al.(1996)은 비교적 짧은 변이기간(transient period) 동안의 시뮬레이션 수행으로 안정상태에서의 목적함수 값을 효과적으로 얻을 수 있는 자동회귀 모델(Autoregressive Model)을 이용한 방법을 제시하였으며, 실험을 통하여 기존의 방법인 Unweighted Batch Means(Law and Kelton 1995), Weighted Batch Means(Bischak et al. 1993) 방법과 비교하여 Mean Error, Root Mean Square Error, Coverage, Mean Interval Length 등의 면에서 비슷하거나 우수함을 보였다.

따라서, 본 연구에서는 비교적 짧은 시간인 Δt 동안 시뮬레이션을 수행하여 안정상태시의 목적함수 값을 추정하기 위해 Voss et al.(1996)이 제시한 방법을 기초로 하여 사용하며, 자동회귀 모델 설명은 제 3.4.1 절 부터 제 3.4.3 절에서 정리된다.

3.4.1 자동회귀 모델

비교적 짧은 변이기간에서 얻은 시뮬레이션 출력 데이터는 데이터간에 자동상관이 강하게 존재하므로 그 출력 과정(output process)을 자동회귀 모델(Autoregressive Model)로 잘 표현할 수 있다(Fishman 1978).

자동회귀 모델은 새로운 관측치 z_t는 과거의 관측치들과 선형 관계를 가진다고 가정되며, 차수(order) p 를 가진 자동회귀 모델 AR(p)은 식 (3.1)과 같이 표현된다(Fuller 1996).

$$z_t = \phi_0 + \sum_{i=1}^{p} \phi_i z_{t-i} + \varepsilon_t, \qquad t = 1,2,... \quad (3.1)$$

여기에서 차수 p 는 유한(finite)하며 $\{\varepsilon_t, t = 1,2,...\}$는 평균 0 이고, 분산이 $\sigma_\varepsilon^2 < \infty$를 만족하는 IID 정규 확률 변수이다. $\Phi = \{\phi_0, \phi_1, ..., \phi_p\}$는 자동회귀 계수 (autoregressive coefficient) 들의 벡터를 의미한다.

시스템 반응치의 평균이 안정상태의 유일한 값으로 수렴한다고 가정하면 안정상태 평균 $\mu = \mu(\phi)$ 은 식 (3.2)와 같이 표현된다.

$$\mu(\Phi) = \lim_{t \to \infty} E[Z_t] = \phi_0 \{1 - \sum_{i=1}^{p} \phi_i\}^{-1} \quad (3.2)$$

$\hat{\Phi}$을 자동회귀 계수 벡터 Φ의 추정치라고 하면, $\hat{\Phi}$에서의 안정상태 평균의 추정치는 다음 식으로 표현된다.

$$\mu(\hat{\Phi}) = \hat{\phi}_0 \{1 - \sum_{i=1}^{p} \hat{\phi}_i\}^{-1}$$

그리고, $\hat{\Phi}$에 대한 조건부 최소자승 추정치(conditional least square estimate)는 다음 식 (3.3)에 의해 구할 수 있다(Fuller 1996).

$$\hat{\Phi} = A_n^{-1} v_n \tag{3.3}$$

여기에서,

$$A_n = \frac{1}{n-p}\sum_{t=p+1}^{n} Z_t' Z_t$$

$$Z_t = (1, z_{t-1}, \ldots, z_{t-p})$$

$$v_n = \frac{1}{n-p}\sum_{t=p+1}^{n} Z_t' z_t$$

그리고, 식 (3.1)이 성립되고, ε_t가 정규분포를 하면, $\hat{\Phi}$는 조건부 최우 추정치(conditional maximum likelihood estimate)가 되고, $\mu(\hat{\Phi})$는 μ의 조건부 최우 추정치가 된다(Voss et al. 1996).

3.4.2 자동회귀 모델의 차수 추정

상대적으로 적은 n 개의 데이터에 의한 AR 모델 식 (3.1)에 유효한 차수 p의 선택은 Broerson and Wensink(1993)에 의해 제시된 방법을 기초로 해서 구할 수 있으며, 절차는 다음의 알고리즘 3.4 와 같다.

[알고리즘 3.4]

단계 1:	p 의 최대값 p_{max}를 선택한다.
단계 2:	$0 \le p \le p_{max}$에 대하여 $S^2(p)$ 값을 계산한다.
단계 3:	$0 \le p \le p_{max}$에 대하여 FIC(p) 값을 계산한다.
단계 4:	FIC(p) 값을 최소로 하는 차수 p^*를 선택한다.

$$S^2(p) = \frac{1}{n}\sum_{t=1}^{n} z_t^2 \prod_{i=1}^{p}(1-v_i)$$

$$\ln\{S^2(p)\} = \ln\left(\frac{1}{n}\sum_{t=1}^{n} z_t^2\right) + \sum_{i=1}^{p}\ln(1-v_i) \tag{3.4}$$

$$v_i = \frac{n-i}{n(n+2)} \cong \frac{1}{n}$$

$S^2(p)$는 residual variance를 의미하고, v_i는 유한 표본 분산계수(finite sample variance coefficients)로서 여러 형태의 공식이 제시되어 있으나, 본 연구에서는 Broerson and Wensink(1993)에 의해 행해진 실험에서 추정치 $\hat{\Phi}$의 분산이 가장 작게 나온 Yule-Walker 방법을 사용하며, 공식은 식 (3.4)로 구성된다.

FIC(p)는 Finite Information Criterion을 의미하며 p 개의 계수를 사용한 모델에 의한 예측 오차(prediction error)를 나타내는 하나의 척도이며 여러 공식이 제시 되었으나, 본 연구에서는 사용되지 않은 미래의 데이터에 대한 적합성과 계산의 효율성을 고려하여 다음의 식 (3.5)를 사용한다.

$$FIC(p) = \ln\{S^2(p) + 2\sum_{i=1}^{p} v_i\} \tag{3.5}$$

FIC를 사용하는 이유는 FIC가 AIC (asymptotic information criterion)의 일반 형태인 GIC(generalized information criterion)와 근사적으로 동일하다는 이론적인 강점을 갖기 때문이다(Broersen and Wensink 1993, Voss et al. 1996).

p_{max}는 이론상으로는 Batch Means 방법을 사용할 때 구해지는 배치의 크기(batch size)보다 하나가 작은 값으로 하면 되지만, 본 연구에서는 Voss et al.(1996)에서 선택한 $p = 15$ ($n \ge 256$ 인 경우)와 Weighted Batch Means 연구에서 Bischak et al.(1993)이 사용한 p 값들을 고려하여 $p_{max} = 20$ 을 선택하였다.

3.4.3 안정상태 평균 및 신뢰구간 추정

단일 실행에 의해 얻은 상대적으로 적은 n 개의 데이터에 의한 안정상태 평균 및 신뢰구간의 추정을 위한 공식은 다음과 같이 유도된다.

$\overline{Z}_{n-p} = \frac{1}{n-p}\sum_{t=p+1}^{n} z_t$ 를 p-제거표본평균(p-runcated sample mean)이라고 하고, $\{Z_t\}$가 식 (3.1)을 만족하는 covariance stationary process 로

수렴한다면, 시뮬레이션을 수행하여 얻은 n 개의 데이터에 의한 안정상태 평균 $\mu(\Phi)$ 의 추정치 $\hat{\mu}_n$은 표준 평균과 편의 수정(bias correction)을 고려하면 다음의 식(3.6)과 같이 표현될 수 있다(Broersen and Wensink 1993, Voss et al. 1996).

$$\hat{\mu}_n = \overline{Z}_{n-p} + \frac{\sum_{i=1}^{p}\hat{\phi}_i\left(\sum_{t=p-i+1}^{p} z_t - \sum_{t=n-i+1}^{n} z_t\right)}{(n-p)\left(1-\sum_{i=1}^{p}\hat{\phi}_i\right)} \qquad (3.6)$$

지금까지 제 3.4 절에서는 목적함수의 추정치 $\mu(\Phi)$를 추정하기 위한 일련의 과정을 설명하였다. 이러한 진행과정을 정리하면 다음의 알고리즘 3.5 로 정리된다.

|알고리즘 3.5|

단계 1: 알고리즘 3.4 를 이용하여 p*를 계산한다.

단계 2: 식 (3.3)을 이용하여 $\hat{\Phi}$를 계산한다.

단계 3: 식 (3.6)을 이용하여 $\hat{\mu}_n$를 계산한다.

위의 알고리즘 3.5 에서 나온 $\hat{\mu}_n$를 알고리즘 3.2 의 단계 2 의 $y_{i,j}$에 대입하여 적은 수의 데이터를 가지고 최적의 시스템 대안을 설정하기 위한 방향을 결정한다. 본 연구에서 제시된 알고리즘은 자동회귀 모델을 사용하기 때문에 시뮬레이션 모델처럼 데이터들이 서로 상관되어 있는 경우에만 적용이 가능하다.

4. 실험 및 평가

제 3 장에서 알고리즘을 개발할 때 결정변수의 값을 변경하면 목적함수의 방향을 예측할 수 있는 경우와 결정변수의 값을 변경하면 목적함수의 방향을 예측할 수 없는 경우를 모두 고려하여 일반적인 알고리즘을 개발하였다. 본 장에서의 수치예제는 가장 복잡하고 모든 경우가 포함되는 문제를 대상으로 설명한다. 따라서, 이러한 경우를 모두 포함하는 (s, S) 재고관리시스템을 대상으로 하여 구현된 알고리즘의 효율을 파악하고 나타나는 결과를 분석한다.

특정 제품을 판매하는 한 회사가 각각의 다음 n 개월 동안 어느 정도의 재고량을 보유해야 하는지를 판단할 때 시뮬레이션 기법을 이용하여 적절한 설계 대안을 선택하고자 한다.

수요가 발생하는 시간 간격은 평균이 0.1 개월인 IID 지수 랜덤 변수를 따르며, 수요 D 의 크기는 다음과 같은 조건으로 IID 랜덤 변수를 따른다고 가정한다.

$$D = \begin{cases} 1 & w.\,p.\ 1/6 \\ 2 & w.\,p.\ 1/3 \\ 3 & w.\,p.\ 1/3 \\ 4 & w.\,p.\ 1/6 \end{cases}$$

매월 초에 회사는 재고 수준을 파악해서 공급자에게 얼마나 많은 재고를 주문해야 하는지를 결정한다. 만약 회사가 H 개의 항목을 주문하면 setup cost K=\$32, 주문되는 항목 수에 따라 incremental cost i=\$3 을 갖는 비용 K+iH 이 발생한다. 주문된 항목이 도착하는 기간은 0.5 와 1 사이의 일양분포를 따른다.

회사는 적정한 발주량을 결정하기 위해 다음과 같은 (s, S) 정책을 따르는데, I 는 매월 초의 재고 수준이다.

$$H = \begin{cases} S - I & if\ I < s \\ 0 & if\ I \geq s \end{cases}$$

(s, S) 재고모델에서 발생하는 비용은 주문비용(ordering cost), 유지 비용(holding cost), 결품 비용(shortage cost)이 발생하면 본 실험에서는 언급된 비용의 합인 평균 총 비용을 목적함수로 하며, 모델에 대한 보다 자세한 설명은 Law and Kelton(1995)을 참고한다.

본 실험에서는 (s, S)를 (40, 60)으로 설정하고 미리 (40, 60)의 경우를 길게 시뮬레이션한 결과치를 대상으로 이 값에 가까운 (s, S)값

을 발견할 수 있는지 Δt의 크기를 변경하면서 실험을 한다. 실험치를 결정하기 위해 (40, 60) 조건에 대하여 10,000 개월을 시뮬레이션하여 실험치로 평균 총 비용 125.74 를 얻었다.

설명한 (s, S) 재고관리시스템을 수리식 모형으로 표현하면 다음과 같다.

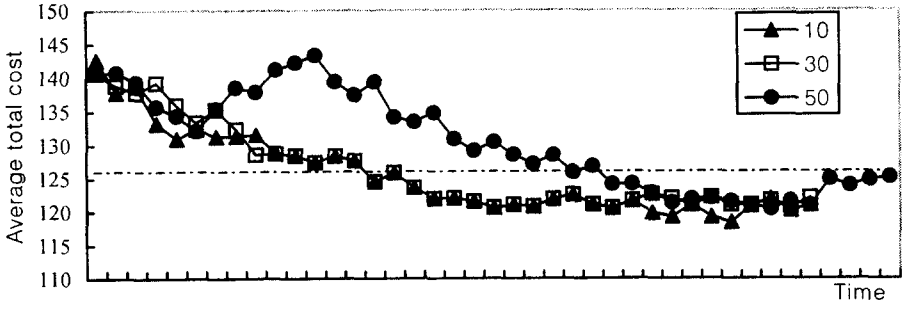
$\arg\{E[f(X)] = c\}$

여기에서,

$E[f(X)]$ = 단위 기간당 발생하는 평균 재고관리 비용

$A = \{125.74-\alpha \leq E[f(X)] \leq 125.74+\alpha\}$

$[x_1, x_2]$=[s: 재발주점, S: 발주량]



<그림 2> Average total cost at Δt in the (s, S) inventory system

제안된 알고리즘에서 필요한 실험치를 위해 재발주점 s 의 하한치와 상한치를 (10, 50), 발주량 S 의 하한치와 상한치를 (20, 70)으로 설정하고 시뮬레이션을 시작할 때의 초기 (s, S)값을 (10, 30)으로 한다. 종료 조건 K=10, A 의 허용오차 α를 ±5 로 하고, 결정변수의 증분 값 Δs 와 ΔS 를 1 로 한다.

본 실험에서 적용하는 자동회귀 모델은 데이터들이 상관(correlation)된 경우를 가정하기 때문에, 예제에서 적용되는 (s, S) 모델에서 얻어지는 $y_{i,i}$ 값들은 개체들의 누적평균(accumulate average)을 사용하여 데이터의 상관 관계를 유지한다.

다음의 <그림 2>는 Δt 가 10, 30, 그리고 50 일 때의 제안된 알고리즘을 수행한 결과이며, 시뮬레이션이 진행됨에 따라 평균총비용이 실험치로 결정된 값 125.74 에 근접해 가는 것을 알 수 있다.

다음의 <표 1>은 Δt 를 변경하는 동안 랜덤 변수를 임의로 선택해서, 각 경우를 50 회씩 시뮬레이션하여 종료되었을 때 얻어지는 (s, S)와 평균재고비용을 나타낸 도표이다.

<표 1> The results of simulation by the proposed algorithm in the (s, S) inventory system

Δt	(s, S)	Average total cost
10	(29, 49)	121.05
30	(32, 52)	122.12
50	(39, 59)	125.20

<표 1>에서 Δt 가 50 인 경우에는 실험치로 결정된 평균총비용 125.74 에 가까운 125.20 으로 평균재고비용이 수렴하고, 구해진 결정변수 (s, S)가 (39, 59)로 실험치로 결정된 (40, 60)에 매우 근사함을 관찰할 수 있다. 그러나, Δt 가 30 이하인 경우에는 시뮬레이션이 종료되는 시점은 빠르지만 알고리즘을 사용하여 계산된 목적함수 값과 결정변수 값이 실험치로 결정된 값들과 많은 차이가 발생한다는 것을 알 수 있다.

5. 결론 및 추후 연구 과제

본 연구에서는, 이산형 결정변수를 갖는 확률적 시스템을 대상으로 단일 실행으로 빠른 시간에 온-라인으로 최적의 시스템의 설계 대안을 발견하고, 얻어진 설계 대안을 안정상태에서 분석할 때 데이터의 손실을 방지하기 위해 짧은 시뮬레이션 수행으로 적은 수의 출력 데이터로 주어진 목적 함수를 평가하기 위해 자동회귀 모델링 방법을 사용하였다.

주어진 알고리즘을 사용하면 반복적인 시뮬레이션 실행을 하지 않고 단일 실행으로 적은 수의 데이터를 가지고 원하는 목표 수준을 만족하는 결정변수의 값을 얻어낼 수 있었다. 그러나 주어진 문제의 영역과 변수의 성격에 따라 목적함수를 만족하는 Δt 의 크기가 무한하게 변할 수 있기 때문에 적절한 Δt 의 선정도 중요한 사항으로 고려되어야 한다. 또한 본 연구에서 목적함수 값의 방향만 고려했는데 크기를 같이 고려하면 시스템 자체가 확률적이기 때문에 변화 크기는 고려되는 시스템에 매우 유동적이기 때문에 크기를 같이 고려할 필요성이 작다.

개발된 알고리즘은 시뮬레이션의 특성상 단조 증가나 단조 감소를 가정한 상태에서 출발하였기 때문에 지역 최적해(local optima)를 전역 최적해(global optima)로 인식하는 단점을 가지고 있다. 그러나 제안 알고리즘은 단일 실행과 짧은 시뮬레이션 수행으로 전역 최적해에 가까운 값을 얻어낼 수 있는 장점을 가지며, 시뮬레이션 시작시의 초기값을 여러 개 선택하면 지역 최적해에 빠지는 문제점을 극복할 수 있고, 시간은 많이 걸리지만 전역 최적해를 보장하는 알고리즘과 혼합하여 사용하면 제안 알고리즘의 특성을 충분히 활용할 수 있을 것으로 기대된다.

추후 연구과제는 시스템의 결정변수로 정성적인 인자(시스템의 운영 룰(FIFO, LIFO 등), 부품의 작업 순서, 또는 설비 배치 문제)가 고려될 때 제안된 해 탐색 알고리즘과 안정 상태 시뮬레이션을 종료하는 알고리즘의 개선이 필요할 것으로 생각된다.

참고문헌

[1] Ahmed, M.A., T. M. Alkhamis, and M. Hasan, "Optimizing discrete stochastic systems using simulated annealing and simulation", *Computers & Industrial Engineering*, 32, 4, 823-836, 1997.

[2] Andradottir, S., "A global search method for discrete stochastic optimization", *SIAM Journal on Optimization*, 6, 2, 513-530, 1996.

[3] Andradottir, S., "A method for discrete stochastic optimization", *Management Science*, 41, 12, 1946-1961, 1995.

[4] Andradottir, S., "Discrete optimization in simulation: a method and applications", *Proceedings of the 1992 Winter Simulation Conference*, 483-486, 1992.

[5] Barry, D.A., and B. Fristedt, *Bandit problems*, Chapman and Hall, London, 1985.

[6] Bischak, D. P., W. D. Kelton, and S. M. Pollock, "Weighted batch means for confidence intervals in steady-state simulations", *Management Science*, 39, 1002-1019, 1993.

[7] Broerson, P. M., and H. E. Wensink, "On finite sample theory for autoregressive model order selection", *IEEE Transactions on Signal Processing*, 41, 194-204, 1993.

[8] Chen, H., "Stochastic root finding in system design", working paper SMS94-8, School of Industrial Engineering, Purdue University, U.S.A, 1994.

[9] Fishman, G. S., *Principles of Discrete Event Simulation*, John Wiley and Sons, NY, 1978.

[10] Fu, M. C., and K. J. Healy, "Techniques for optimization via simulation: an experimental study on (s,S) inventory system", *IIE Transactions*, 29, 3, 191-200, 1997.

[11] Fuller, W. A., *Introduction to statistical time series*, John Wiley and Sons, NY, 1996.

[12] Glynn, P. W., "Optimization of stochastic systems", *Proceedings of 1986 Winter Simulation Conference*, 52-59, 1986.

[13] Gong, W. B., Y. C. Ho, and W. Zhai, "Stochastic comparison algorithm for discrete optimization with estimation", *Proceedings of the 31st IEEE Conference on Decision and Control*, 795-800, 1992.

[14] Heidergott, B., "Sensitivity analysis of a manufacturing workstation using perturbation analysis techniques", *International Journal of Production Research*, 3, 611-622, 1995.

[15] Ho, Y. C., and X. R. Cao, *Perturbation analysis of discrete event dynamic systems*, Kluwer Academic Publishers, 1991.

[16] Jacobson, S. H., and L. W. Schruben, "Techniques for simulation response optimization", *Operations Research Letters*, 8, 1-9, 1989.

[17] Law, A. M., and W. D. Kelton, *Simulation Modeling and Analysis*, McGraw-Hill, 1995.

[18] Lee,Y. H., and K. Iwata, "Part ordering through simulation-optimization in a FMS", *International Journal of Production Research*, 29, 7, 1309-1323, 1991.

[19] Meketon, M. S., "Optimization in simulation: a survey of recent results", *Proceedings of 1987 Winter Simulation Conference*, 58-67, 1987.

[20] Pierreval, H., and L. Tautou, "Using evolutionary algorithms and simulation for the optimization of manufacturing systems", *IIE Transactions*, 29, 181-189, 1997.

[21] Rubinstein, R.Y., and A. Shapiro, *Discrete event systems: sensitivity analysis and stochastic optimization by the score function method*, John Wiley & Sons, 1992.

[22] Safizadeh, M. H., "Optimization in simulation: current issues and the future outlook", *Naval Research Logistics*, 37, 807-825, 1990.

[23] Shi, L., and O. Sigurdur, "Nested partitions method for stochastic optimization", Technical Report, Dept. of I. E., University of Wisconsin-Madison, 1997.

[24] Voss, P. A., J. Haddock, and T. R. Willemain, "Estimating steady state mean from short transient simulations", *Proceedings of the 1996 Winter Simulation Conference*, 222-229, 1996.

[25] Wild, R. H., and J. J. Pignatiello, "Finding stable system designs: a reverse simulation technique", *Communications of the ACM*, 35, 10, 87-98, 1994.

[26] Yakowitz, S., and E. Lugosi, "Random search in the presence of noise with application to machine learning", *SIAM Journal on Scientific Statistical Computing*, 11, 702-712, 1990.

[27] Yan, D., and H. Mukai, "Stochastic discrete optimization", *SIAM Journal on Control and Optimization*, 30, 594-612, 1992.

저자소개

박경종

1992 한양대학교 산업공학과 졸업

1994 한양대학교 대학원 산업공학과 졸업 (공학석사)

1998 한양대학교 대학원 산업공학과 졸업 (공학박사)

1995~1997 동일CIM부설연구소

1997~1999. 9. 대우정보시스템

1999. 10~현재 PriceWaterHouseCoopers

관심분야 : Simulation Optimization, Simulation Output Analysis, SCM(Supply Chain Management)

이영해

1977 고려대학교 산업공학, 석사

1983 Univ. of Illinois, 산업시스템공학, 석사

1986 Univ. of Illinois, 산업공학 및 경영과학, 박사

경력 한국시뮬레이션학회, 대한산업공학회,
한국경영과학회, 대한설비관리학회 이사
Purdue Univ., 오사카대학 방문교수
대우중공업(주)

1986~현재 한양대학교 산업공학과 교수

1995~현재 한국시뮬레이션학회 부회장

관심분야 : Simulation Optimization, Simulation Output Analysis, SCM (Supply Chain Management) Web-based Simulation