한국시뮬레이션학회 논문지
Vol. 17, No. 2, pp. 75-82 (2008. 6)

관측불가능한 임의순서규칙 대기행렬시스템 분석

박진수[1] · 김윤배[1†]

Analysis of Unobservable RSS Queueing Systems

Jinsoo Park · Yun Bae Kim

ABSTRACT

The times of service commencement and service completion had been used for inferring the queueing systems. However, the service commencement times are difficult to measure because of unobservable nature in queueing systems. In this paper, for inferring queueing systems, the service commencement times are replaced for arrival times which can be easily observed. Determining the service commencement time is very important in our methods. The methods for first come first served(FCFS), last come first served(LCFS) queueing discipline are already developed in our previous work. In this paper, we extend to random selection for service(RSS) queueing discipline. The performance measures we used are mean queueing time and mean service time, the variances of two. The simulation results verify our proposed methods to infer queueing systems under RSS discipline.

Key words : Queue inference, Optimization, Random selection

요 약

기존의 대기행렬시스템 추론에 관한 연구들은 서비스시작시점과 서비스종료시점을 이용하여 수행되었다. 그러나 서비스시작시점은 시스템 내부 관측치로서 실제로 관측하기 어려울 때가 많다. 본 논문에서는 서비스시작시점 대신 외부 관측치인 도착시점을 이용하여 대기행렬시스템을 추론한다. 도착 및 이탈시점에 근거한 대기행렬 모형의 분석은 서비스시작시점의 결정이 관건이다. 본 연구의 선행연구로서 선입선출 및 후입선출 시스템에 대한 서비스시작시점의 결정방법은 이미 개발되었다. 본 논문에서는 임의순서규칙 대기행렬시스템으로 추론 범위를 확장한다. 구하고자 하는 성능척도는 평균대기시간과 평균서비스시간이며 이들의 분산을 구함으로써 통계적 분석기법의 사용이 가능하도록 하였다. 마지막으로 시뮬레이션 결과로부터 얻은 데이터를 이용하여 본 연구의 추론방법을 적용함으로써 그 타당성을 검증하였다.

주요어 : 대기행렬추론, 임의순서규칙, 최적화

1. 서 론

대기행렬시스템 추론은 Larson[9,10]에 의해 서비스시작시점과 서비스종료시점을 이용하여 수행되었으며 이를 기반으로 발전해 왔다[1-3,5-7]. 물론 다른 관점에서 접근한 추론방법들도 연구되어 왔으며[8,12-14] 최근에는 베이시안 기법을 사용[4]하기도 하였다. 그러나 대부분 Larson의 방법에서 크게 벗어나지 못하거나 접근방법이 다르더라도 단수서버(single server) 시스템에 적용한 제한적인 것이었다. 이러한 추론방법들의 문제점은 도착과정을 포아송과정으로 가정한다는 것과 시스템 내부 관측치인 서비스시작시점을 사용한다는 점이다. 본 논문에서는 서비스시작시점대신 도착시점을 사용하고 복수개의 서버가 존재하는 시스템에 대해 다룬다. 본 논문은 선행된 선입선출(FCFS : first come first served)과 후입선출(LCFS : last come first served) 대기행렬시스템의 추론[9,16]방법을 임의순서규칙(RSS : random selection for service) 대기행렬시스템으로 확장한 것이다.

한 서비스가 끝나면 기다리는 고객들 중에서 아무나 골라 다음 서비스를 제공하는 시스템의 서비스 정책을 임의순서규칙이라고 한다. 이 시스템은 고객들이 도착하면 대기열의 임의의 자리에 서서 대기를 하다가 자신의 차례가 오면 서비스를 받고 시스템을 이탈하는 정책으로도 해

2008년 4월 24일 접수, 2008년 5월 28일 채택
[1] 성균관대학교 시스템경영공학과
주 저 자 : 박진수
교신저자 : 김윤배
E-mail; kimyb@skku.edu

석할 수 있다. 유휴한 서버가 존재한다면 도착하는 고객은 즉시 서비스를 제공받기 시작한다. 실생활에서는 자주 볼 수 없으나 컴퓨터시스템과 같은 속도를 요하는 경우에 종종 사용되는 정책이다. 특히 대량의 데이터를 저장하는데 쓰이는 힙 메모리(heap memory)가 대표적인 임의순서규칙 시스템이라 하겠다.

본 연구의 기본 가정은 서비스시간이 독립이라는 점이다. 이 가정을 기반으로 먼저 서버의 수를 알고 있는 경우에 대해 추론을 수행한다. 서버의 수를 알고 있다면 각 고객의 도착과 이탈시점만을 이용하여 서비스시간이나 대기시간과 같은 성능척도의 평균을 정확히 계산할 수 있다. 또한 통계적 추론을 위하여 이들의 분산을 근사적으로 구해내는 것이 가능하다. 이로부터 서버의 수를 모르는 경우에 대한 추론방법을 전개한다. 이 경우에는 서버의 수만 정확히 결정하면 모든 계산이 가능해진다. 이러한 추론방법의 타당성을 검증하기 위해 시뮬레이션으로부터 얻은 데이터로 추론을 시행하였다.

본 연구는 외부관측만이 가능한 시스템의 내부행태를 추론하는 모형을 개발한다. 따라서 서버의 성능척도나 대기시간에 의해 민감하게 대응해야 하는 시스템의 분석 및 적용이 가능하다.

본 논문은 5장으로 구성하였다. 먼저 2장에서는 서버의 수를 알고 있는 경우의 추론방법을 소개한다. 다음으로 3장에서는 서버의 수를 모르는 경우에 이를 구하는 추론 모형을 소개한다. 4장에서는 추론 모형의 타당성을 검토하기 위해 실험한 내용과 그 결과를 보여준다. 5장에서는 간단한 요약과 추후에 연구 및 보완해야 할 내용을 언급한다.

2. 서버의 수를 아는 경우

2.1 서비스시작시점

추론을 위해 먼저 다음의 외부관측치들을 정의하자.

N : 시스템을 거쳐 간 총 고객수

A_i : 고객 i의 도착시점

D_i : 고객 i의 이탈시점

c : 서버수

이들로부터 다음을 정의하고 계산할 수 있다.

$N(t)$: 시점 t에서의 고객수

$N_{\max}$: $\max\{N(t)\}$

N_i^A : i 번째 고객이 도착하면서 보는 고객수

추론의 목적인 각 고객의 대기시간과 서비스시간을 얻기 위해서는 서비스시작시점을 구해야 한다. 즉 고객 i의 대기시간과 서비스시간을 각각 Q_i, S_i라 하고 서비스시작시점을 B_i라고 하면 다음 식을 이용하여 Q_i와 S_i를 계산할 수 있다.

$$Q_i = B_i - A_i \tag{1}$$

$$S_i = D_i - B_i \tag{2}$$

고객수가 c보다 작은 경우($N_i^A < c$)는 후입선출에서와 마찬가지로 도착시점이 서비스시작시점이다. 이외 경우의 서비스시작시점을 구해내기 위해서는 혼잡기간(congestion period)을 정의해야 한다. 혼잡기간이란 모든 서버가 바쁜 기간을 말한다. 즉 고객수가 c 이상이 되는 시점부터 다시 c보다 작아지는 시점까지를 뜻한다. 혼잡기간을 정의하면 서비스시작시점들은 간단히 구해낼 수 있다. 그림 1에서 보듯이 시스템 최대고객수가 $(c+k)$인 경우 혼잡기간 내의 각 고객의 이탈시점들이 서비스시작시점이 된다.

이를 이용하면 평균서비스시간과 평균대기시간은 쉽게 구해낼 수 있다. 하지만 이들의 분산을 얻기 위해서는 각 서비스시작시점이 누구의 것인지를 알아야 한다. 즉 각 고객에 대한 서비스시작시점을 구해야 그들의 대기시간과 서비스시간을 얻을 수 있다는 것이다. 이를 구하는 것은 시스템의 특성상 불가하므로 먼저 그 근사해를 구해보고자 한다. 임의순서규칙 시스템은 먼저 도착한 고객이 먼저 서비스 받지도 않으며 나중에 도착한 고객이 먼저 서비스 받지도 않는다. 다만 추측할 수 있는 것은 먼저 서비스 받은 고객이 먼저 도착할 것이라는 점이다. 모든 고객이 그렇다는 것이 아니라 다만 근사적으로 그렇다는 것을 추측할 수 있을 뿐이다. 이를 적용하여 근사적인 서비스시작시점을 구해낼 수 있는데 그것이 바로 시뮬레이션 재구성 알고리즘이다. 이 알고리즘을 이용하면 우리가 구

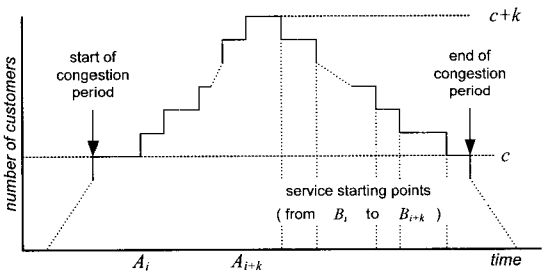


그림 1. 서비스시작시점

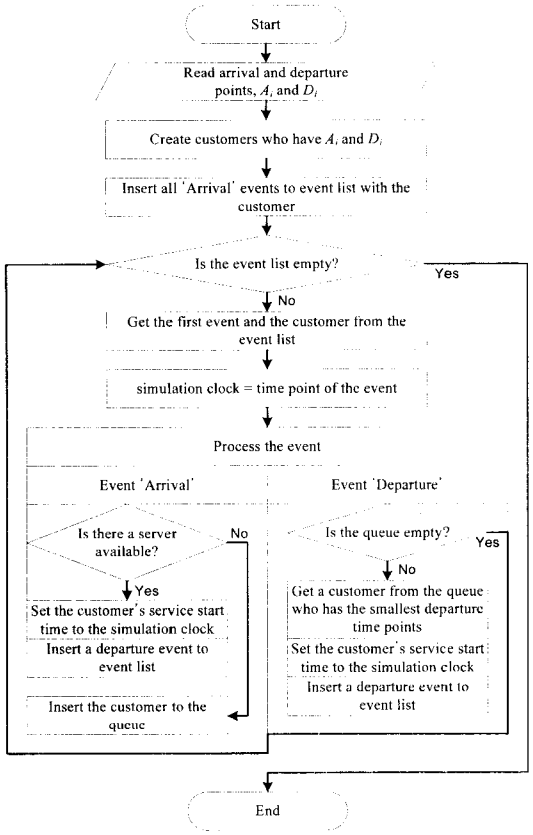


그림 2. 시뮬레이션 재구성 알고리즘

하고자하는 각 고객의 대기시간과 서비스시간을 근사적으로 얻을 수 있다.

2.2 시뮬레이션 재구성

시뮬레이션 재구성 알고리즘은 그림 2와 같다.
먼저 입출력 데이터로부터 고객데이터를 만들어내고 입력 데이터들로부터 도착이벤트를 발생시킨다. 도착이벤트의 처리과정은 유휴한 서버를 찾아낸 후 서비스를 시작하거나 유휴한 서버가 없으면 고객을 대기열에 대기시킨다. 서비스를 시작함과 동시에 이 도착에 해당하는 고객의 이탈시점으로부터 이탈이벤트, 즉 서비스종료이벤트를 발생시키게 된다. 이탈이벤트에서는 대기열을 조사하여 가장 먼저 이탈하는 고객을 우선으로 뽑아 서비스를 시작하고 그 고객의 이탈시점을 이용하여 새로운 이탈이벤트를 발생시킨다. 이러한 과정을 반복하게 되면 시뮬레이션을 재구성할 수 있다.

이러한 시뮬레이션 재구성이 원래 시스템을 그대로 반영하는 경우는 서버의 수가 1개인 경우와 서비스시간의 분산이 아주 작은 경우뿐이다. 그렇지 않은 경우는 원래 시스템에 근사한 시뮬레이션을 수행할 수 있을 뿐이다. 특히 본래 시스템의 서비스시간의 분산에 따라 그 결과의 차이가 심해진다. 서비스시간의 분산이 작으면 먼저 서비스를 시작한 고객이 먼저 나가게 될 것이다. 따라서 본 시뮬레이션 재구성 방법에 의한 결과가 실제 시스템의 값에 매우 근접하게 된다. 반면 서비스시간의 분산이 크게 되면 서비스를 시작한 시간과는 관계없이 이탈이 일어나기 때문에 서비스시간의 분산에 대한 결과가 잘 맞지 않게 된다. 다만 뒤의 결과에서 다시 언급하겠지만 대기시간의 분산에 대해서는 실제 값에 거의 유사한 값을 추론할 수 있다. 결국 시뮬레이션 재구성 방법은 서버의 수가 적거나 서비스시간의 분산이 작은 경우 매우 잘 맞고 그렇지 않은 경우 즉, 서버의 수가 많고 서비스시간의 분산이 큰 경우에 있어 서비스시간의 분산을 추정함에 있어 잘 맞지 않게 된다. 이러한 시뮬레이션 재구성 방법의 한계 때문에 이를 보완할 새로운 방법이 필요하다.

2.3 최적화 방법

시뮬레이션 재구성 방법의 한계를 극복하기 위해 일반적인 $GI/G/c/RSS$ 시스템의 대기시간과 서비스시간의 독립성을 이용하여 추론방법을 한 단계 더 발전시키는 것이 가능하다. 즉, Q를 대기시간, S를 서비스시간, T를 시스템체재시간이라 했을 때 다음이 성립한다.

$$Var(T) = Var(Q+S) = Var(Q) + Var(S) \qquad (3)$$

만일 i 번째 고객의 서비스시작시점을 잘못 추정하게 되면 대기시간과 서비스시간은 다음과 같이 추정될 것이다.

$$\hat{Q}_i = \hat{B}_i - A_i = B_i - A_i - (B_i - \hat{B}_i) = Q_i - \Delta_i \qquad (4)$$

$$\hat{S}_i = D_i - \hat{B}_i = D_i - B_i + (B_i - \hat{B}_i) = S_i + \Delta_i \qquad (5)$$

식에서 $\hat{Q}_i$, $\hat{S}_i$, $\hat{B}_i$는 각각 대기시간, 서비스시간, 서비스시작시점의 추정치이다. 여기서 주의할 점은 추정된 표본평균이 항상 같다는 것이다. 즉 $\overline{\Delta}=0$이므로 $\overline{S}=\overline{\hat{S}}$, $\overline{Q}=\overline{\hat{Q}}$ 이다. 이는 주어진 고객의 이탈시점과 도착시점으로부터 체재시간, 대기시간, 서비스시간을 추정해 내기 때문이다. $\hat{T}_i$를 고객 i의 시스템체제시간 추정치라고 하면 결국 다음이 성립한다.

$$\hat{T}_i = \hat{Q}_i + \hat{S}_i = Q_i + S_i = T_i \qquad (6)$$

이로부터 다음과 같은 관계식을 유도해낼 수 있다. 이 식에서 $\hat{Q}$, $\hat{S}$는 각각 대기시간과 서비스시간의 추정치를 뜻한다.

$$\begin{aligned} Var(T) &= Var(\hat{Q}+\hat{S}) \\ &= Var(\hat{Q}) + Var(\hat{S}) + 2Cov(\hat{Q}, \hat{S}) \end{aligned} \quad (7)$$

이제 $Cov(\hat{Q}, \hat{S})$에 식 (2)와 (3)의 관계를 대입하여 정리해 보자.

$$\begin{aligned} Cov(\hat{Q}, \hat{S}) &= Cov(Q-\Delta, S+\Delta) \\ &= E[(Q-\Delta)(S+\Delta)] - E(Q)E(S) \\ &\qquad (\because E(\Delta)=0) \end{aligned} \quad (8)$$

일반적인 $GI/G/c$ 대기행렬시스템의 확률적 특성상 정해진 T 값에 대해 $E[QS] \neq E[(Q-\Delta)(S+\Delta)]$ 일 것이므로 다음이 성립한다.

$$Cov(Q, S) \neq Cov(\hat{Q}, \hat{S}) \quad (9)$$

결국 $|Cov(\hat{Q}, \hat{S})|$ 값을 최소화함으로써 비교적 정확한 Q_i와 S_i를 구할 수 있다. 여기서 주의할 점은 (3)의 관계식을 그대로 적용하지 않았다는 점이다. 식 (3)은 이론적인 결과이지만 실제 시스템으로부터 얻은 데이터에는 미약하게나마 표본공분산 값인 $Cov(Q, S)$가 존재하기 때문이다. 이로부터 최적화식을 세우기 위해 다음의 용어를 먼저 정의한다.

$C(r)$: 혼잡기간 내의 임의의 r 번째 이탈시점

여기서 r은 임의의 정수 값으로써 $C(r)$은 혼잡기간 내에서 임의로 뽑은 이탈시점을 의미한다. 즉 $C(r)$들이 서비스시작시점인 B_i의 추정치로 쓰이는 것이다. 이로부터 다음의 최적화식을 정의하고 이를 풀면 임의순서규칙 시스템의 추론이 가능해 진다.

$$minimize \quad \left| \sum_{i=1}^{N} \left(\hat{Q}_i - \overline{\hat{Q}}\right)\left(\hat{S}_i - \overline{\hat{S}}\right) \right| \quad (10)$$

$$s.t. \quad \hat{B}_i = A_i \quad (N_i^A < c) \quad (11)$$

$$\hat{B}_i = C(r) \quad (o/w) \quad (12)$$

$$A_i \leq \hat{B}_i < D_i \quad (13)$$

$$\hat{Q}_i = \hat{B}_i - A_i \quad for\ all\ i. \quad (14)$$

$$\hat{S}_i = D_i - \hat{B}_i \quad for\ all\ i. \quad (15)$$

이 최적화 문제는 서버의 수가 많고 서비스시간의 분산이 큰 시스템의 경우에 정확한 추론결과를 얻을 수 있다. 서버의 수가 적은 경우는 시뮬레이션 재구성 방법의 결과가 보다 정확하며 서비스시간의 분산이 작은 경우에는 오히려 잘못된 추론을 가져올 수도 있다. 서비스시간의 분산이 작은 경우에는 $Cov(Q, S)$의 값이 서비스시간의 분산보다 커지는 경우가 생기기 때문이다. 따라서 이 최적화 문제를 적용하기 위한 서버의 수와 서비스시간의 분산의 상한 또는 하한을 결정하는 것이 매우 중요하다.

이 문제는 직관적인 해를 얻기가 매우 어려울 뿐더러 데이터 수인 N이 커지면 그 조합의 수가 기하급수적으로 많아지게 되어 점점 더 풀어내기 힘들어진다. 이에 본 연구에서는 이 최적화식을 풀기 위해 시뮬레이티드 어닐링(SA: simulated annealing) 방법을 이용하였다. 시뮬레이티드 어닐링은 목적함수 (10)을 최소화하는 $\hat{B}_i$, 즉 서비스시작시점을 찾도록 구성하였다.

3. 서버의 수를 모르는 경우

서버의 수를 모르는 경우는 먼저 적당한 서버의 수를 결정하고 서버의 수가 바뀜에 따라 시스템 상태가 어떻게 변하는지를 관찰하면 된다. 선입선출 시스템과 후입선출 시스템에 있어서 서버의 수가 바뀜에 따라 서비스시간의 분산이 달라지는 것을 확인한 바 있다[9,16]. 사실 임의순서규칙 시스템에서는 서버의 수를 알고 있는 경우에도 정확한 서비스시간의 분산을 추정하는 것이 어렵다. 다만 본 논문에서는 시뮬레이션 재구성에 의해서 구한 서비스시간의 분산 역시 실험적으로 달라지는 것을 확인하였다. 즉 추정된 서버의 수가 실제 시스템의 서버수와 같아질 때 서비스시간의 분산추정치가 가장 작아지는 것을 알 수 있었다. 따라서 서비스시간의 분산추정치를 최소화하는 최적화식을 풀면 실제 시스템의 서버수를 구할 수 있다. $SR_{\hat{c}}(i)$를 서버의 수를 $\hat{c}$로 가정하고 시뮬레이션 재구성에 의해 결정한 고객 i의 서비스시작시점이라 하면 다음과 같은 최적화식을 해결함으로써 실제 시스템의 서버수를 구할 수 있다.

$$minimize \quad \frac{1}{N-1} \sum_{i=1}^{N} \left(\hat{S}_i - \overline{\hat{S}}\right)^2 \quad (16)$$

$$s.t. \quad \hat{B}_i = SR_{\hat{c}}(i) \quad (i=1, \cdots, N) \quad (17)$$

$$\hat{Q}_i = \hat{B}_i - A_i \quad (18)$$

$$\hat{S}_i = D_i - \hat{B}_i \quad (19)$$

$$\hat{Q}_i \geq 0 \quad (20)$$

$$\hat{S}_i > 0 \quad (21)$$

이 식에서도 역시 $\hat{B}_i$, $\hat{Q}_i$, $\hat{S}_i$는 각각 서비스시작시점, 대기시간, 서비스시간의 추정치를 나타내며 $\hat{\bar{S}}$는 $\hat{S}_i$의 표본평균을 의미한다.

고객수과정을 살펴 시스템최대고객수를 결정하면 $\hat{c}$의 상한을 결정할 수 있다. 즉 $\hat{c}$는 시스템최대고객수 $N_{\max}$를 넘지 않는 범위 내에서 결정되어야 한다. 이는 서버의 수가 시스템최대고객수보다 커지는 경우 모든 고객이 도착과 동시에 서비스를 시작하는 것으로 간주되기 때문이다. 서버의 수를 정확히 결정하면 서비스시간의 근사해를 구하는 과정은 서버의 수를 아는 경우와 동일하다.

본 최적화식이 유일한 해를 얻지 못하는 경우는 서버이용율 ρ[1]가 매우 적을 때이다. 서버이용율이 매우 적으면 고객수가 서버의 수를 넘지 못하는 경우가 생기고 이러한 경우 모든 고객은 도착과 동시에 서비스를 받게 된다. 복수개의 해가 나오게 되는데 이때에는 가장 작은 해, 즉 $\hat{c}$을 선택하는 것을 추천한다. 하지만 일반적으로 서버이용율이 매우 적은 경우는 분석대상에서 제외되므로 본 논문에서도 서버이용율이 큰 경우에 대한 추론 결과만 고려하였다.

4. 시뮬레이션 및 추론결과

본 절에서는 3절 최적화식의 타당성을 재검토하기 위하여 시뮬레이션과 추론을 수행하였다. 시스템으로부터 각 고객의 도착시점과 이탈시점, 그리고 서비스시작시점을 각각 1000개씩 추출하여 도착시점과 이탈시점을 추론을 위한 데이터로 이용하고 서비스시작시점을 추론에 대한 참값으로 이용하였다. 또한 정확한 서비스시작시점을 얻어내는 것이 불가능하였으므로[2] 서비스시간과 대기시간의 분산 계산결과를 추가하였다.

4.1 시뮬레이션 모델

실험에 사용한 시스템은 총 81가지이다. 먼저 서버의 수를 1개, 3개, 7개일 때에 대해 각각 수행하였으며, 이 세 가지 시스템에 대해 모든 서버의 서비스율의 합 $c\mu$에 대한 입력률 λ의 비율 $\rho=\lambda/(c\mu)$을 각각 0.8, 0.9, 1.0의 3가지 형태 시스템으로 나누어 총 9가지 시스템에 대해 시뮬레이션을 수행하였다. 또한 도착과정에 도착간격이 지수분포를 따르는 포아송도착과정, 도착간격이 항상 일정한 확정적 도착과정, 도착간격이 일반분포를 따르는 일반 재생과정에 대해 실험 하였으며 서비스시간에 대해서도 마찬가지로 세 가지 분포를 이용하였다. 일반 재생과정에 쓰인 시간간격의 분포는 분산이 1인 정규분포에 시간의 역행을 막기 위해 절대값을 취하여 사용하였다. 평균도착률 λ는 1이며 표 1은 실험에 쓰인 평균서비스시간이다.

표 1. 분포에 사용한 모수

c	mean service time		
	$\rho=0.8$	$\rho=0.9$	$\rho=1.0$
1	0.8	0.9	1
3	2.4	2.7	3
7	5.6	6.3	7

표 2. 시뮬레이션 재구성에 의한 추론결과

도착	서비스	c	ρ	$Var(Q)$	$Var(\hat{Q})$	R.E.	$Var(S)$	$Var(\hat{S})$	R.E.
M	M	1	0.8	101.7	101.7	0.00	0.1	0.1	0.00
			0.9	602.1	60.2	0.00	0.8	0.8	0.00
			1.0	461.3	461.3	0.00	1.0	1.0	0.00
		3	0.8	26.0	27.4	0.05	5.4	3.8	0.29
			0.9	157.2	157.9	0.00	6.9	4.2	0.39
			1.0	326.0	323.7	0.00	9.1	4.7	0.48
		7	0.8	128.7	131.2	0.02	29.5	19.6	0.34
			0.9	101.5	119.7	0.22	37.4	20.8	0.44
			1.0	423.3	433.9	0.03	49.6	20.1	0.59
	D	1	0.8	8.1	8.1	0.00	0.0	0.0	0.00
			0.9	70.5	70.5	0.00	0.0	0.0	0.00
			1.0	248.0	248.0	0.00	0.0	0.0	0.00
		3	0.8	5.9	5.9	0.00	0.0	0.0	0.00
			0.9	72.0	72.0	0.00	0.0	0.0	0.00
			1.0	222.4	222.4	0.00	0.0	0.0	0.00
		7	0.8	9.2	9.2	0.00	0.0	0.0	0.00
			0.9	39.2	39.2	0.00	0.0	0.0	0.00
			1.0	126.1	126.1	0.00	0.0	0.0	0.00
	G	1	0.8	1224.4	1224.4	0.00	0.5	0.6	0.00
			0.9	5650.1	5650.1	0.00	0.6	0.6	0.00
			1.0	12173.5	12173.5	0.00	0.7	0.7	0.00
		3	0.8	9.6	9.5	0.01	1.0	0.8	0.17
			0.9	37.0	37.4	0.01	1.0	0.8	0.23
			1.0	524.9	523.7	0.00	1.1	0.7	0.39
		7	0.8	5.7	5.7	0.02	1.0	0.9	0.16
			0.9	24.7	24.9	0.01	1.0	0.8	0.24
			1.0	855.5	854.5	0.00	1.1	0.6	0.45

4.2 추론결과 및 분석

먼저 시뮬레이션 재구성을 통해 얻은 근사해 추론결과는 표 2에 주어진바와 같다. 모든 시스템에 대한 결과를

1) i.e. traffic intensity

2) 실제 시스템에서는 표본공분산인 $Cov(Q, S)$의 값이 존재하는 경우가 많으므로 최적해를 찾더라도 근사해가 되기 쉽다.

보일 수 없어 도착이 포아송과정인 경우에 대한 결과만을 나열하였다. 표 2에서 M은 도착과정이나 서비스시간의 분포가 마코비안 또는 지수분포라는 것을 나타낸다. 그리고 D는 확정적분포를 나타내며, GI 또는 G는 정규확률난수의 절대값을 의미한다. 표에서 쓰인 R.E.는 참값에 대한 오차의 비율인 상대오차(relative error)를 가리킨다. 각 분산의 값은 소수 셋째 자리에서 반올림 하였으며 상대오차는 소수 다섯째 자리에서 반올림 하였다.

표 2의 결과를 관찰해 보면 시뮬레이션 재구성 방법의 특징을 몇 가지 도출할 수 있다. 먼저 서비스시간이 확정적인 시스템의 경우이다. 서비스시간이 확정적인 시스템의 경우는 서비스간격이 같기 때문에 먼저 서비스를 시작한 고객이 무조건 먼저 시스템을 이탈하게 되어 있으므로 정확한 추론이 가능해 지는 것이다. 다음으로 $M/G/c$의 경우를 살펴보자. 여기에 일반분포로 쓰이는 정규분포는 그 분산을 1로 하였기 때문에 간격 자체가 매우 고른 편에 속한다. 이 경우 역시 먼저 서비스를 시작한 고객이 먼저 이탈할 확률이 크다. 따라서 실제 시스템을 잘 반영한 결과라 할 수 있다. 또한 서버의 수가 하나일 때, 즉 단일서버일 때에도 정확한 대기시간과 서비스시간을 추론할 수 있다는 점이다. 이는 서버가 하나인 대기행렬시스템의 특성을 생각해보면 당연한 결과라 할 수 있다. 즉, 서버가 하나이면 당연히 먼저 서비스 받는 고객이 먼저 나가기 때문에[3] 이탈시점의 순서를 알면 정확한 서비스시작시점을 구할 수 있다. 마지막으로 서비스시간의 분산은 서버의 수가 증가함에 따라 오차가 커지지만 대기시간의 분산은 참값에 유사한 결과를 내는 것을 관찰할 수 있다. 대기행렬시스템을 분석의 목적인 대기열 자체의 분석에 있어서는 이 결과를 근사치로 사용해도 될 것이다.

서버의 수가 많고 서비스시간에 대한 분석이 필요할 때에는 두 번째 최적화 방법을 이용해야 한다. 최적화 방법은 앞에서 나온 결과를 초기해로 설정하고 SA 알고리즘을 이용하여 최적의 근사해를 구하였다. 이는 표 3에 주어져 있다. 추론오차가 큰 경우인 서버의 수가 7개일 때와 서비스 분포가 지수분포일 때에 대해서만 그 결과를 보였다. 표에 주어진 각 항목은 앞의 결과와 동일하다. 표 3의 결과를 보면 대부분의 시스템에 있어 상대오차가 작은 것을 확인할 수 있다. 이러한 오차가 발생하는 것은 이론적인 값이 0인 공분산 항 $Cov(Q, S)$의 값이 실제 시스템의 표본에서는 0이 아닌 값으로 나오기 때문이다.

마지막으로 서버의 수를 모르는 경우에 대한 추론결과를 살펴보자. 서버의 수를 모르는 경우의 추론결과는 최적화식이 서버의 수를 올바르게 결정하는 지를 확인하려 한다. 따라서 앞서 수행한 81가지 시스템의 시뮬레이션 결과 중에서 도착시점과 이탈시점만을 이용하여 서버의 수를 추론 한 후 실제 시스템과 같은지를 살펴보았다. 추론을 수행한 결과 서버수를 정확히 찾아내어 본 결과에는 생략하였다. 다만 그림 3과 그림 4를 통해 서버의 수의 변화에 따라 식 (16)의 목적함수가 어떻게 변화하는지를 살펴보자. 그림에서 M, D, GI, G는 표 2에서 쓰인 그것들

표 3. SA 알고리즘을 이용한 최적화 결과

시스템	ρ	$Var(Q)$	$Var(\hat{Q})$	R.E.	$Var(S)$	$Var(\hat{S})$	R.E.
$M/M/7$	0.8	128.7	131.1	0.02	29.5	30.6	0.03
	0.9	101.5	115.2	0.13	37.4	39.5	0.06
	1.0	423.3	439.4	0.04	49.6	46.8	0.05
$D/M/7$	0.8	5.6	6.6	0.19	29.5	28.6	0.03
	0.9	124.6	133.0	0.07	37.4	40.1	0.07
	1.0	81.9	89.4	0.09	49.3	46.1	0.06
$GI/M/7$	0.8	0.8	0.8	0.05	29.5	29.4	0.00
	0.9	5.7	6.1	0.07	37.4	37.8	0.01
	1.0	16.2	18.0	0.11	49.3	51.0	0.03

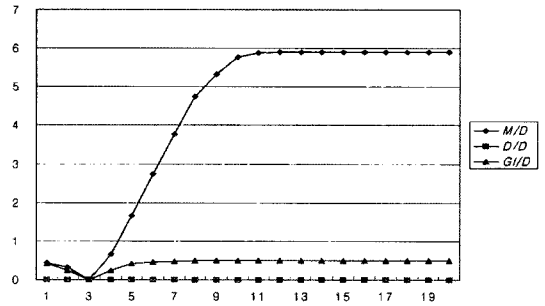


그림 3. $c=3$, $\rho=0.8$인 경우 목적함수의 변화

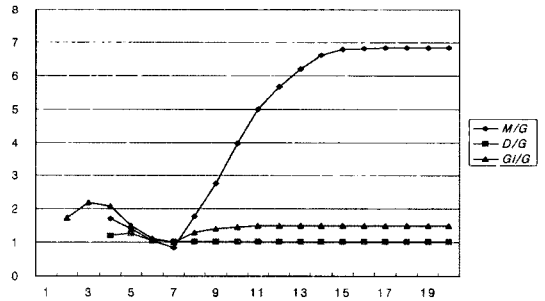


그림 4. $c=7$, $\rho=0.8$인 경우 목적함수의 변화

3) 먼저 들어온 고객이 먼저 나가는 것이 아니라는 점에 주의하자.

과 동일하다. 또한 타점이 되지 않은 부분은 제약조건을 위배하여 비가시해(非可視解; infeasible solution)가 되는 부분이다.

여기서 우리는 추정된 서버수가 실제 시스템의 서버수와 동일할 때 최소가 됨을 확인할 수 있다. 즉 아래로 볼록인 형태를 띠어 서버의 수를 찾아내는 데에 적합한 목적함수임을 알 수 있다. 물론 결과에 나오지 않은 다른 시스템의 추론 결과도 비슷한 양상을 보임을 확인하였다.

5. 결론 및 추후연구과제

본 논문에서는 임의순서규칙 대기행렬시스템의 추론을 시도 하였다. 서버의 수와 고객의 입출력 시점을 알 때 이로부터 우리는 서비스시작시점들을 추론하여 평균대기시간과 평균서비스시간을 계산하였다. 또한 각 고객의 서비스시작시점을 찾아내어 대기시간과 서비스시간의 분산에 대한 근사치를 구해낼 수 있었다. 현재까지 수행한 서비스정책 이외에도 우선순위가 존재하는 시스템이나 $N-$정책과 같은 특수한 서비스정책이 존재하는 경우에 대한 추론방법의 개발이 필요할 것이다.

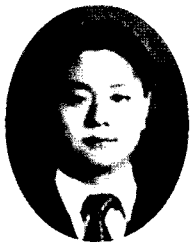
참 고 문 헌

1. Basawa, I., Bhat, U. and Lund, R., Maximum likelihood estimation for single server queues from waiting time data, *Queueing Systems* 24, 155-167, 1996.
2. Bertimas, D. and Servi, L., Deducing queueing from transactional data: The queue inference engine, revisited, Operations Research 40, S217-S228, 1992.
3. Bingham, N. and Pitts, S., Non-parametric estimation for the M/G/infinity queue, *Annals of the Institute of Statistical Mathematics* 51, 71-97, 1999.
4. Choudhury, A., Borthakur, A. C., Bayesian inference and prediction in the single server Markovian queue, Metrica, 67, 371-383, 2006.
5. Daley, D. and Servi, L., Exploiting Markov-chains to infer queue length from transactional data, *Journal of Applied Probability* 29, 713-732, 1992.
6. Dimitrijevic, D., Inferring most likely queue length from transactional data, *Operations Research Letters* 19, 191-199, 1996.
7. Jang, J., Suh, J. and Liu, C., A new procedure to estimate waiting time in GI/G/2 systems by server observation, *Computers and Operations Research* 28, 597-611, 2001.
8. Jones, L., Inferring balking behavior from transactional data, *Operations Research* 47, 778-784, 1999.
9. Kim, Y. B., Park, J., Kim, J. B., Inference for Unobservable Queues from Arrival and Departure Data, *Asian Sim Conference*, 2005.
10. Larson, R., The queue inference engine: Deducing queue statistics from transactional data, *Management Science* 36, 586-601, 1990.
11. Larson, R., The queue inference engine: Addendum, *Management Science* 37, 1062-1062, 1991.
12. Mandelbaum, A. and Zeltyn, S., Estimating characteristics of queueing networks using transactional data, *Queueing Systems* 29, 75-127, 1998.
13. Masuda, Y., Exploiting partial information in queueing -systems, *Operations Research* 43, 530-538, 1995.
14. Pickands, J. and Stine, R., Estimation for the M/G/infinite queue with incomplete information, Biometrika 84 295-308, 1997.
15. Toyoizumi, H., Sengupta's invariant relationship and its application to waiting time inference, *Journal of Applied Probability* 34, 795-799, 1997.
16. 김윤배, 박진수, 도착 및 이탈시점에 근거한 관측 불가능한 후입선출 대기행렬 모형의 분석, *한국시뮬레이션학회 논문지* 16, 75-81, 2007.

김 윤 배 (kimyb@skku.edu)

1982 성균관대학교 산업공학 학사
1986 University of Florida, Industrial and Systems Engineering 공학석사
1992 Rensselaer Polytechnic Institute Decision Science and Engineering Systems Ph. D.
1995～1998 성균관대학교 산업공학과 조교수
1998～2004 성균관대학교 산업공학과 부교수
2004～현재 성균관대학교 산업공학과 교수

관심분야 : 시뮬레이션 출력분석, 인터넷 트래픽 분석, Telecom Network Performance Analysis



박 진 수 (jsf001@skku.edu)

1998 성균관대학교 산업공학과 학사
2000 성균관대학교 산업공학과 석사
2008 성균관대학교 산업공학과 박사

관심분야 : 시뮬레이션모델링, 대기행렬이론, 인터넷 트래픽 분석