한국시뮬레이션학회 논문지
Vol. 18, No. 4, pp. 29-37 (2009. 12)

이산사건 시뮬레이션에서의 효율적인 이벤트 리스트 관리를 위한 MList의 개선 방안

김성곤[1] · 임동순[2†]

An Improved MList for Efficient Event List Management in Discrete Event Simulation

Seong-Gon Kim · Dong-Soon Yim

ABSTRACT

This paper deals with the priority queues exploited for the management of future event list in discrete event simulation. Among several implementations of priority queues, MList which consists of 3 tiers has been known to reveal the good performance. To improve the performance of MList, Dynamic-Shift MList (DSMList) is proposed in this paper. Whenever the number of events in tier 3 exceeds a critical number, DSMList creates new calendar queue in tier 2, then moves events from Tier 3 to the calendar queue. Instead of one calendar queue, therefore, a number of calendar queues are dynamically created in tier 2. Throughout experiments for the performance evaluation of DSMList, i t shows that at least 20% improvement is obtained compared with MList.

Key words : Event list management, Priority queue, Discrete event simulation

요 약

본 논문은 기존의 멀티 리스트 기반 엠리스트(MList)의 성능을 개선하기 위하여 동적 쉬프트 방식을 적용한 방법을 소개한다. 개선된 엠리스트는 일정한 수의 이벤트가 Tier 3에 쌓이게 되면 삭제 이벤트가 발생하지 않더라도 자동 쉬프트 작업을 통해 Tier 3에 저장된 이벤트들을 Tier 2의 새로운 칼렌다 큐로 이동시킨다. 즉, 기존에는 Tier 2에 하나의 칼렌다 큐가 있었으나, 개선된 방법에서는 다수의 칼렌다 큐가 동적으로 생성, 삭제된다. 이러한 동적 구조는 저장된 이벤트 수에 따라 적응력을 갖는 장점을 가져 성능측정 실험 결과 동적 쉬프트를 적용한 엠리스트는 기존의 엠리스트에 비해 20% 이상의 성능개선을 보였다.

주요어 : 사건 리스트 관리, 우선순위 큐, 이산사건 시뮬레이션

1. 서 론

이산 사건 시뮬레이션의 구성 요소는 크게 모델과 모델에서 발생하는 사건(event)들을 관리하는 엔진으로 구분할 수 있다. 시뮬레이션 엔진은 모델로부터 미래에 발생될 사건들을 입력 받아 제시간에 올바른 사건들이 발생할 수 있도록 관리한다. 이러한 사건 관리를 위해 사건들의 우선순위, 즉, 사건의 발생시각인 타임 스템프(time stamp) 순으로 하나씩 삭제되도록 하는 특징을 갖는 우선순위 큐 자료구조를 이용한다.

우선순위 큐를 구현한 방법에는 링크드 리스트를 이용한 선형 구조, 힙 트리 또는 스프레이 트리를 이용한 트리 구조, 그리고, 칼렌다 큐(Henriksen, 1977), 엠리스트(Goh 등, 2003, 2004) 등과 같은 멀티 리스트 구조 등이 있다. 시뮬레이션 엔진에 어떠한 우선순위 큐 구조를 사용하느냐에 따라 시뮬레이션 실행의 효율성이 좌우된다(임동순, 2008). 특히, 10만 건 이상의 사건을 관리하는 대형 모델에서는 우선순위 큐 구조의 성능이 매우 중요하다. 가장 바람직한 우선순위 큐 구현을 선택하기 위해서는 이들에

* 이 논문은 2009년도 한남대학교 교비학술연구조성비 지원에 의하여 연구되었음.
2009년 6월 21일 접수, 2009년 12월 7일 채택
[1] DENSO PS ELECTORNICS Corp.
[2] 한남대학교 산업경영공학과
주 저 자 : 김성곤
교신저자 : 임동순
E-mail; dsyim@hnu.kr

대한 성능 평가를 필요로 한다. 대상이 되는 시뮬레이션 모델의 사건 생성과 삭제 패턴을 묘사한 성능 실험 모델을 생성하여야 하고, 이러한 실험 모델 하에서 우선순위 큐 구현의 성능을 평가하여야 한다. 가장 일반적으로 이용되는 범용의 실험 모델로는 Hold(Jones, 1986), Up-Down(Ronngren 등, 1997), 마코브 프로세스 모델(Chung 등, 1993)이 있다. 사건의 삽입과 삭제가 안정적으로 발생하는 상황을 나타내는 Hold 모델에서 11가지의 우선순위 큐 구조의 성능을 평가한 Jones(1986)의 연구는 큐 크기가 큰 경우에는 스프레이 트리가 우수하고, 10개 이하의 큐 크기에서는 링크드 리스트가 우수함을 나타내었다. Brown(1988)은 그 자신이 제안한 칼렌다 큐를 스프레이 트리와 비교하여 칼렌다 큐가 우수하다고 보고하였다. 다단계 마코브 프로세스 실험 모델을 이용한 실험에서도 칼렌다 큐의 성능이 우수하다고 보고되었다(임동순, 2008). 엠리스트(MList)는 칼렌다 큐를 확장한 것으로 칼렌다 큐의 성능 보다 우수하다고 보고되었다(Goh 등, 2003).

본 논문에서는 우선순위 큐 구현 중 우수하다고 알려져 있는 엠리스트에 대해 성능을 개선한 결과를 논의한다. 2장에서는 칼렌다 큐와 엠리스트에 대한 자세한 설명을 하고, 3장에서는 기존의 엠리스트를 수정하여 성능을 개선한 엠리스트를 소개한다. 기존의 엠리스트는 3층 구조로 되어 있고, 2층은 하나의 칼렌다 큐로 구성되어있다. 개선된 엠리스트에서는 2층에 다수의 칼렌다 큐가 동적으로 생성 소멸되는 구조를 가져 관리하에 있는 사건의 수에 따라 적응력을 갖도록 되어 있다. 4장에서는 사건 삽입 및 삭제의 다양한 패턴을 고려한 실험을 통해 개선된 엠리스트의 성능을 비교 평가한 결과를 제시한다.

2. 칼렌다 큐와 엠리스트

2.1 칼렌다 큐

칼렌다 큐는 이산 사건 시뮬레이션에서 O(1)의 이상적인 수행속도를 구현하기 위해 제안된 알고리즘이다. 기본적으로 동일한 시간 간격을 갖는 버켓(bucket)들이 배열로 정의되어 있고, 각 버켓에는 그 시간 간격에 해당하는 노드들이 링크드 리스트 구조에 타임 스템프 순으로 정렬되어 있다. 칼렌다 큐를 운영할 때 요구되는 주요 변수는 다음과 같다.

- 총 버켓 수(nBuckets)
- 한 버켓의 기간(timeWidth)
- 현재 버켓(nowBucket)

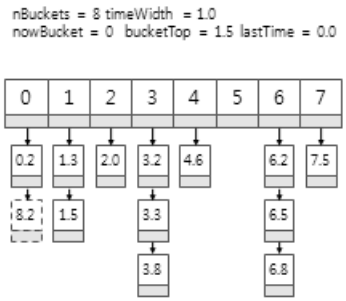

그림 1. 칼렌다 큐 구조

- 다음 버켓의 중앙 시각(bucketTop)
- 가장 최근에 삭제된 사건의 시각(lastTime)

새로운 사건을 칼렌다 큐에 넣기 위하여 사건의 타임 스템프가 속하는 버켓을 찾고, 그 버켓의 링크드 리스트에 노드가 삽입 된다. 삭제를 위해 nowBucket이 가르키는 버켓에서 가장 빠른 타임 스템프의 노드를 찾는다. 만약 버켓에 노드가 비어 있으면 다음 버켓을 조회하여 가장 빠른 타임 스템프를 갖는 노드를 삭제한다.

칼렌다 큐의 예를 나타내는 그림 1은 시간 간격이 1.0인 8개의 버켓으로 구성되어 있어 칼렌다의 1년은 8.0으로 설정되었다. 타임 스템프가 8.2인 사건을 삽입한다고 하자. 시각 8.2가 위치하여야 하는 버켓을 찾는 것은 단순하다. 아래 식에 의해 구한 인덱스 0의 버켓에 타임 스템프 8.2의 사건 노드를 삽입한다.

$$\begin{aligned} bucket &= |8.2/timeWidth| \bmod nBuckets \\ &= 8 \bmod 8 = 0 \end{aligned}$$

삭제를 위해서는 nowBucket이 가르키는 버켓에서 가장 빠른 시각인 0.2의 사건 노드를 삭제한다. 단, 이 노드의 타임 스템프가 다음 버켓의 중앙시각인 1.5보다 작을 때 가능하다. 이 노드를 삭제함으로써 변수 중 lastTime만 0.2로 바뀐다. 다시 삭제를 수행 한다고 하자. nowBucket이 가르키는 버켓에 있는 노드 중 가장 빠른 타임 스템프는 8.2이다. 그러나 다음 버켓인 인덱스 1의 중앙 시각은 1.5 이다. 따라서 8.2는 1.5 보다 크므로 8.2의 사건 노드를 삭제하지 않고, nowBucket을 1로, bucketTop을 2.5로 수정한다. 인덱스가 1인 버켓에 있는 노드 중 가장 빠른 타임 스템프는 1.3이고, 다음 버켓의 중앙 값인 2.5보다 작다. 따라서 타임 스템프 1.3의 노드를 삭제한다.

칼렌다 큐의 성능은 삽입과 삭제에 소요되는 평균 시

간으로 표시되는데 각 동작의 소요시간은 노드들이 버켓에 분산된 형태에 따라 변화한다. 어느 한 버켓에 많은 노드들이 집중적으로 저장되어 있는 경우에는 그 버켓에 노드 삽입 시 많은 노드들을 검색해야 한다. 반대로 많은 버켓들이 비어 있는 경우에는 삭제 시 많은 버켓을 검색해야 한다. 따라서 큐 동작 소요시간의 단축을 위해서는 사건 노드들의 균등한 분포가 중요하며 균등한 노드 분포는 timeWidth와 nBucket의 값에 따라 달라진다.

timeWidth가 너무 큰 경우에는 많은 노드들이 소수의 버켓들에 만 배치될 수 있고, timeWidth가 너무 작은 경우에는 삭제될 노드와 다음에 삭제될 노드 사이에 많은 버켓이 놓여져 있어 버켓 검색 시간이 길어진다. 또한, nBucket이 작으면 한 버켓에 너무 많은 노드들이 배치되고, nBucket이 너무 크면 많은 버켓들이 비어 있게 된다. 따라서 칼렌다 큐에서는 재배치(resize) 알고리즘을 사용하여 버켓의 수와 크기를 조절한다. 즉, 노드의 수가 마지막으로 재배치 작업했을 때의 버켓 수보다 2배 이상 되거나 1/2배 이하로 되는 경우에는 새로운 버켓 수를 2배 또는 1/2배 할당하고, 새로운 버켓의 간격을 계산하여 저장되어 있는 노드들을 새로 할당 된 버켓들에 재배치 한다.

2.2 엠리스트

엠리스트는 사건들을 3개의 층(tier)으로 나누어 관리함으로서 칼렌다 큐의 단점 중 하나인 재배치 작업을 하지 않는다.

가. 1 층 (T1)

T1은 링크드 리스트 구조로 되어 있다. 모든 사건 중 먼저 발생할 사건들이 타임 스템프 순으로 정렬되어 저장된다.

나. 2 층 (T2)

T2는 칼렌다 큐와 유사한 멀티 리스트 구조로 되어 있다. 그러나 각 버켓에 저장된 노드들은 칼렌다 큐와달리 정렬되어 있지 않고 단지 저장되는 순서대로 존재한다. 또한 칼렌다 큐에서는 1년 이상의 사건들도 버켓에 저장될 수 있지만 엠리스트에서는 1년 동안의 사건들 만 버켓에 저장되며 1년 이후의 사건들은 3 층에서 관리하게 된다.

다. 3 층 (T3)

T3는 T1과 동일한 링크드 리스트 구조로 되어 있다. 하지만 T1과는 달리 노드들을 정렬하지 않고 저장되는 순서대로 존재한다. 3개의 층 중 가장 늦게 발생할 사건들을 저장하고 있으며 먼 미래에 발생할 사건들이 T1, T2의 성능에 영향을 주지 않도록 버퍼 역할을 한다.

새로운 사건을 저장하기 위해서 이 사건이 T3, T2, T1

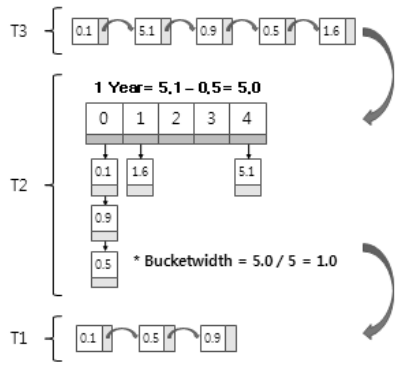

그림 2. MList 내부동작의 예

중 어느 층에 삽입되어야 하는지 조사하여 해당하는 층에 노드를 삽입한다. 삭제가 요구되면 T1의 가장 빠른 타임 스템프를 갖는 노드를 찾는다. 만약, T1에 노드가 없으면 T2의 가장 빠른 버켓에 있는 노드들이 T1으로 이동된다. 만약, T2에도 노드가 없으면 T3의 모든 노드들이 T2로 이동된 후 T2의 가장 빠른 버켓에 있는 노드들이 T1으로 이동된다. T3의 노드들이 T2로 이동될 때 T3에 있는 노드의 수를 T2의 버켓 수로 하여 버켓 간격을 결정한다.

그림 2는 엠리스트 알고리즘의 내부동작을 나타낸다. 그림에서 0.1, 5.1, 0.9, 0.5, 1.6의 타임 스템프를 갖는 5개 사건이 입력되면 T3에 차례대로 저장된다. 이 때 T1과 T2는 비어 있다고 가정하자. 삭제가 요구되면 T3에 있는 노드들이 T2로 이동되고, T2의 첫 번째 버켓에 있는 세 노드들이 정렬되어 T1으로 이동된다. 마지막으로 T1에 있는 가장 빠른 시각인 0.1의 노드가 삭제된다.

3. 동적 쉬프트 엠리스트(DSMList)

본 연구에서 제안하는 DSMList는 3개 층으로 구성되어 T1과 T3는 엠리스트의 구조와 동일하다. 그러나 그림 3에서와 같이 T2는 하나의 칼렌다가 아닌 n개의 칼렌다로 리스트로 구성 되어 있다. T3에 정해진 수의 사건이 쌓일 경우 삭제가 발생하지 않더라도 새로운 칼렌다를 생성하여 리스트에 삽입하고. 사건들을 T3에서 T2로 이동시키는 동적 쉬프트를 수행하여 다수의 칼렌다를 관리한다. DSMList를 구현하는데 사용되는 주요 변수들은 표 1과 같다.

3.1 삭제 운영

가장 빨리 발생할 사건의 삭제는 아래의 삭제 알고리

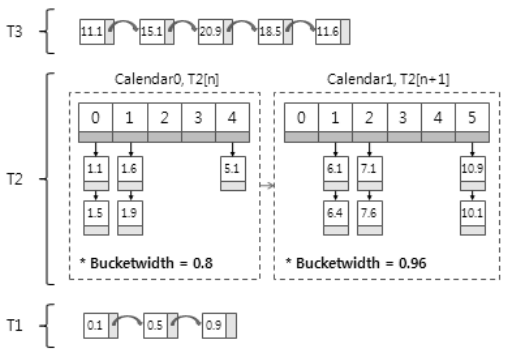

그림 3. DSMList 내부구조의 예

표 1. DSMList의 주요 변수

주요변수	내용설명
T1_nEvent	T1에 있는 사건 수
T2_DynamicShift	쉬프트 작업을 결정하는 사건 수
T2_DynamicIdx	T2에 있는 칼렌다 수
T2_DynamicCur	현재 포인터가 가르키는 Tier 2의 칼렌다
T2_nEvent	T2에 있는 총 사건 수
T2_Cur[n]	T2의 n번째 칼렌다에서 포인터가 위치한 버켓의 시작시간 값
T2_Min[n]	T2의 n번째 칼렌다에 있는 사건타임 스템프 최소값
T2_Bw[n]	T2의 n번째 칼렌다의 버켓 간격
T2_Idx[n]	T2의 n번째 칼렌다의 포인터가 위치한 버켓 (최초 0부터 시작)
T3_Min	T3의 사건 타임 스템프 최소값
T3_Max	T3의 사건 타임 스템프 최대값
T3_nEvent	T3에 있는 사건의 수

즘에 의한다. 기본적으로 엠리스트와 동일하나 T2는 1개가 아닌 n개의 칼렌다를 노드로 하는 리스트로 구성되어 있어 이를 고려한다.

삭제 알고리즘

단계 1: T1_nEvent > 0 이면 T1에서 삭제를 수행하고 종료한다.

단계 2: T2_nEvent > 0 이면 포인터 T2_DynamicCur가 위치한 T2의 칼렌다에서 포인터 T2_Idx가 위치한 버켓의 사건들을 타임 스템프 순으로 정렬하여 T1에 삽입한 후 다음 버켓으로 포인터 T2_Idx를 이동하고, 단계 1로 간다.

단계 3: T3_nEvent > 0 이면 T2에 새로운 칼렌다를 생성하여 T3의 사건들을 그 칼렌다로 이동시킨 후 단계 2로 간다.

단계 4: 삭제될 사건이 없으므로 종료한다.

3.2 삽입 운영

새로운 사건의 타임 스템프를 TS라고 할 때 아래 알고리즘은 사건을 DSMList에 삽입한다. 기존 엠리스트 알고리즘과 비교하여 T3, T1에서의 삽입과 동일하며 T2에서는 다른 운영방법을 사용한다.

삽입 알고리즘

단계 1: TS > T3_Min 이면 T3에 사건을 삽입하고 종료한다.

단계 2: T2에 있는 n개의 칼렌다에 대해 i를 n 부터 1까지 감소하여 TS > T2_Min[i] 이면 해당되는 i번째 칼렌다에 사건을 삽입하고 종료한다.

단계 3: T1에 사건을 삽입한다.

3.3 동적 쉬프트

시뮬레이션 진행 중 T1, T2에 노드의 수가 0이 되거나 동적 쉬프트 작업이 발생할 때 T3에서 T2로 사건들이 이동되는 작업이 발생하게 된다. 이 과정에서 T3에 저장 된 사건들의 타임 스템프 값이 고르게 분포되어 있다면 T2로 이동시 각 버켓에 1개 사건들이 삽입되어 가장 이상적인 성능을 보여주게 된다. 하지만 반대로 사건들의 대부분이 한 방향으로 모여 있다면 일부의 버켓에는 수 많은 사건들이 삽입되고 나머지 버켓에는 빈 버켓이 존재하여 시뮬레이션 성능에 나쁜 영향을 주게 된다.

T3에서 T2로 사건들을 이동시 그림 3과 같은 불규칙적인 사건 타임 스템프 분포에 대응하기 위하여 다음과 같은 방법을 사용하였다.

첫째, T3에 저장된 사건들의 타임 스템프 값 평균, 중간값을 구하고 평균 / 중간 값을 하여 1보다 작으면 사건들이 중간보다 앞쪽으로 치우쳐 있다고 할 수 있는데 이중에서 1/4 이상 사건들이 치우쳐진 경우를 대상으로 하기 위하여 0.5보다 작은 경우에만 사건 타임 스템프 분포에 대응하게 되며 0.5보다 큰 경우에는 정상적으로 진행된다.

둘째, 가장 최근에 T3에 입력 된 사건 샘플 25개를 뽑아 표준편차를 구하고 표준편차 값의 왜곡을 피하기 위하여 평균+표준편차 보다 큰 샘플은 제외한 후 한번 더 조정된 표준편차를 구한다. 만약 T3에 사건들의 수가 25개보다 작아 샘플 수 25개를 채울 수 없으면 T3 사건 수를

샘플 수로 한다.

셋째, 평균+조정된 표준편차 값으로 T3_Max를 변경한 후 T3에서 T2로 사건 이동작업을 진행하였으며 평균+조정 표준편차 값보다 큰 T3의 사건들은 T3에 그대로 남겨둔다.

4. 성능 실험 및 분석

4.1 실험 모델

3장에서 소개한 DSMList의 성능을 평가하기 위하여 Hold 모델과 Up-Down 모델의 두가지 성능 실험 모델 하에서 칼렌다 큐, MList와 비교하였다. Hold 모델(Chung 등, 1993; Goh 등, 2003)은 n개의 사건을 큐에 넣은 후 삭제와 삽입을 반복하여 큐의 크기가 항상 n이 되도록 유지한다. 반면, Up-Down 모델(Henriksen, 1977)은 n개의 사건을 연속하여 삽입 한 후 큐에 있는 n개의 사건을 연속하여 삭제하는 사이클을 반복한다. 따라서 Hold 모델은 이산 사건 시뮬레이션의 안정 상태가 장기간 유지되는 경우에 적합하고, Up-Down 모델은 불안정 상태가 오랫동안 유지되는 경우에 적합하다.

표 2. 랜덤시각 분포 함수

분포 함수	표현식
Uni(0, α)	α * x
Tri(0, α)	α * sqrt(x)
NegTri(0, α)	α * (1-sqrt(1-x))
Biomodal	9.95238*x + if x<0.1 then 9.5238 else 0
Exp(1)	if x=0 then infinity else -ln(x)
Exp Mix	if(x<0.9) then Exp(1) else Exp(100)
Pareto(α)	pow(1/(1-x),1/α)

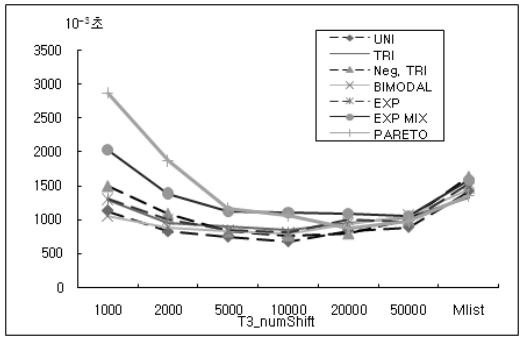

그림 4. Hold 모델에서의 T2_DynamicShift 변화에 따른 성능

성능 실험 모델에서는 삽입 패턴에 따라 가상적인 사건들이 생성되어야 하고, 사건들은 발생 시각인 타임 스템프를 가지고 있어야 한다. 이러한 타임 스템프는 시뮬레이션 현재 시각(가장 최근에 삭제된 사건의 타임 스템프)에 랜덤의 시각을 더한 값으로 한다. 본 연구의 실험에서는 표 2와 같은 7가지 분포함수를 이용하여 랜덤의 시각을 생성하였다. 표 2의 표현식은 랜덤의 시각을 구하는 식을 나타내며 x는 0부터 1 사이의 난수를 의미한다.

4.2 동적 쉬프트 파라미터

동적 쉬프트 MList에서는 T3에 일정한 수(T2_DynamicShift)의 사건이 쌓이게 되면 자동적으로 T2에 새로운 칼렌다를 생성하여 이동된다. 때문에 적정한 값의 T2_DynamicShift를 결정하여야 하는 부가적인 문제를 포함하여 이를 위한 실험을 수행하였다.

그림 4와 그림 5는 각각 Hold 모델과 Up-Down 모델에서 큐 크기를 100,000으로 하였을 때 7가지 랜덤 시각 분포 하에서 T2_DynamicShift 수의 변화에 따른 평균 실행시간을 나타낸다. 두 모델 모두 T2_DynamicShift값이 1,000에서 10,000으로 증가할수록 더 좋은 성능을 보인다. 10,000 이상에서는 T2_DynamicShift의 값이 증가하여 T2_DynamicShift 값이 무한대인 즉, 기존의 MList에 가까움에 따라 성능은 나빠진다. Pareto 랜덤시각 분포 함수를 제외하면 대체로 10,000의 T2_DynamicShift에서 좋은 성능을 보인다.

100,000외의 다른 9가지 큐 크기에 대한 실험 결과에서 위의 실험 결과와 유사하여 T2_DynamicShift 값이 큐 크기의 0.1배 일때 대체로 좋은 성능을 보였다. 따라서 이후의 실험에서는 T2_DynamicShift의 값을 큐 크기의 0.1배로 고정하였다.

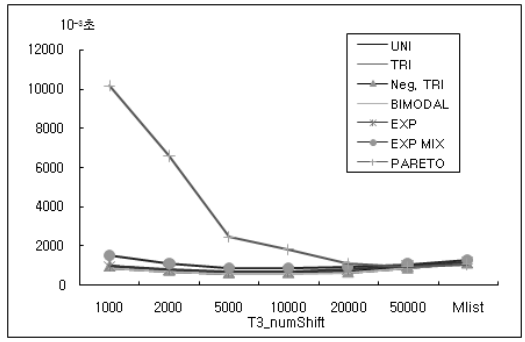

그림 5. Up-Down 모델에서의 T2_DynamicShift 변화에 따른 성능

표 3. Hold Model 하에서의 실험 결과

분포	큐 구조	큐 크기										평균
		1000	2000	3000	4000	5000	6000	7000	8000	9000	10000	
Uni (0,100)	DSMList	47 (-52%)	109 (42%)	155 (29%)	280 (36%)	298 (39%)	423 (37%)	452 (47%)	564 (44%)	682 (40%)	733 (46%)	(31%)
	MList	47 (-52%)	172 (9%)	313 (-44%)	452 (-4%)	595 (-23%)	765 (-15%)	983 (-14%)	1016 (0%)	1188 (-4%)	1388 (-2%)	(-15%)
	CQ	31	188	218	435	485	668	859	1014	1139	1360	
Tri (0,1.5)	DSMList	62 (22%)	125 (34%)	203 (40%)	281 (33%)	378 (47%)	562 (52%)	626 (65%)	638 (75%)	767 (79%)	844 (82%)	(53%)
	MList	63 (20%)	124 (34%)	265 (22%)	341 (19%)	594 (17%)	652 (45%)	666 (63%)	968 (63%)	1171 (68%)	1313 (72%)	(42%)
	CQ	79	188	338	420	719	1175	1779	2592	3607	4736	
Neg Tri (0,1000)	DSMList	63 (-34%)	141 (-13%)	219 (12%)	250 (40%)	315 (42%)	453 (37%)	500 (35%)	609 (39%)	642 (49%)	765 (47%)	(25%)
	MList	61 (-30%)	204 (-63%)	328 (-32%)	484 (-16%)	625 (-14%)	891 (-24%)	1015 (-33%)	1188 (-19%)	1407 (-11%)	1558 (-8%)	(-25%)
	CQ	47	125	249	419	546	721	765	999	1264	1443	
Bimo dal	DSMList	46 (27%)	94 (33%)	172 (32%)	282 (25%)	373 (28%)	453 (35%)	548 (31%)	670 (33%)	704 (41%)	795 (45%)	(33%)
	MList	79 (-25%)	171 (-21%)	328 (-29%)	438 (-16%)	595 (-15%)	732 (-6%)	921 (-16%)	1076 (-8%)	1265 (-7%)	1406 (2%)	(-14%)
	CQ	63	141	254	376	516	692	795	997	1186	1441	
Exp (1.0)	DSMList	47 (0%)	112 (28%)	156 (45%)	267 (31%)	376 (35%)	453 (42%)	593 (38%)	673 (38%)	672 (47%)	817 (42%)	(35%)
	MList	62 (-32%)	188 (-21%)	282 (0%)	392 (-1%)	546 (6%)	654 (16%)	858 (10%)	954 (13%)	1204 (5%)	1372 (3%)	(0%)
	CQ	47	155	282	387	580	781	954	1093	1266	1420	
Exp Mix	DSMList	79 (-23%)	172 (25%)	233 (40%)	343 (38%)	469 (33%)	568 (38%)	657 (39%)	785 (40%)	843 (43%)	1110 (32%)	(31%)
	MList	94 (-47%)	188 (18%)	314 (20%)	468 (15%)	612 (13%)	780 (15%)	939 (13%)	1142 (13%)	1263 (15%)	1469 (10%)	(9%)
	CQ	64	230	391	549	703	922	1078	1311	1483	1641	
Pareto (1.0)	DSMList	110 (-41%)	142 (17%)	265 (1%)	315 (33%)	420 (29%)	594 (19%)	768 (21%)	717 (36%)	874 (34%)	1063 (31%)	(18%)
	MList	2188 (-2705%)	21482 (-12390%)	70687 (-26375%)	137016 (-29114%)	194267 (-32883%)	282201 (-38452%)	330657 (-34024%)	387063 (-34398%)	507048 (-38139%)	570985 (-36833%)	(-28531%)
	CQ	78	172	267	469	589	732	969	1122	1326	1546	

4.3 Hold 모델에서의 실험결과

표 3은 Hold 모델을 이용한 실험 결과를 나타내며 괄호 안의 백분율은 칼렌다 큐(CQ) 성능에 비해 실행시간 면에서 향상된 정도를 나타낸다. 표의 결과에 따르면 랜덤 시각의 분포와 큐 크기에 따라 우선순위 큐 구조들의 성능이 약간씩 상이하다. 특히, MList는 랜덤 시각 분포에 민감하여 TRI, EXP MIX 분포의 랜덤시각에서는 칼렌다 큐 보다 좋은 성능을 보이나 UNI, NegTRI, BIMODAL, Pareto 분포에서는 칼렌다 큐 보다 더 나쁜 결과를 나타낸다. 특히, Pareto 분포 하에서는 칼렌다 큐에 비해 평균 28,531%의 성능 저하를 가져온다. 반면 DSMList는 큐의 크기가 작은 10,000에서 UNI, NegTRI, EXP MIX, Pareto 분포일 경우 칼렌다 큐 보다 나쁜 결과를 가져오지만 이를 제외한 모든 큐 크기와 분포에서 칼렌다 큐에 비해 향상된 결과를 가져온다. 기존의 MList가 매우 나쁜 결과를 낳는 Pareto 분포에서도 DSMList는 평균 18%된 향상된 결과를 나타낸다.

그림 6은 DSMList를 기존의 MList와 비교하여 실행시간 면에서 향상된 백분율을 나타낸다. 모든 분포와 큐 크기에서 MList에 비해 매우 향상된 결과를 가져온다. 가

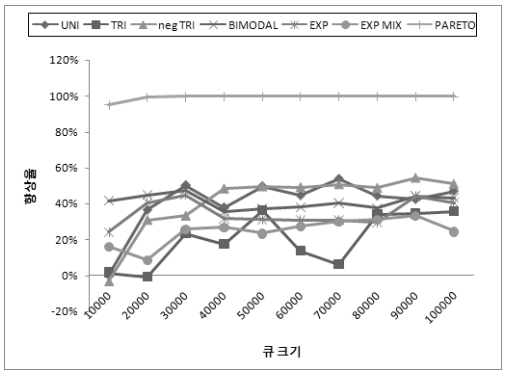

그림 6. Hold Model 하에서 MList에 대한 DSMList의 향상율

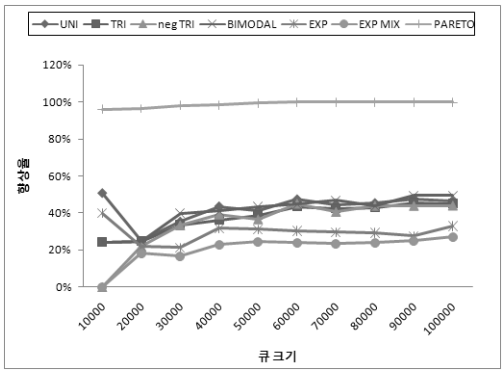

그림 7. Up-Down Model 하에서 MList에 대한 DSMList의 향상율

장 적은 향상율을 보이는 TRI 분포에서 조차 평균 20%의 향상율을 보이고, Pareto 분포에서는 100%의 향상율을 나타낸다.

4.4 Up-Down 모델에서의 실험결과

표 4는 Up-Down Model을 이용한 실험 결과를 나타낸다. MList는 Pareto 분포를 제외한 모든 분포에서 칼렌다 큐 보다 더 좋은 성능을 나타낸다. 특히, Tri 분포에서는 평균 80%의 향상율을 나타낸다. 그러나 Hold 모델에서와 마찬가지로 Pareto 분포에서는 28,810%의 성능 저하율을 나타낸다.

DSMList는 Pareto 분포에서의 큐 크기 40,000과 100,000을 제외하면 모든 분포와 큐 크기에서 칼렌다 큐 보다 향상된 결과를 가져온다. Tri 분포에서는 평균 80%의 향상을 보이고, Pareto 분포에서도 평균 18%의 향상율을 나타낸다. 이 같은 결과는 Hold 모델 보다 Up-Down 모델에서 DSMList의 성능이 칼렌다 큐에 비해 더욱 성능이 우수하다는 것을 나타낸다. 이는 칼렌다 큐가 Up-Down 모델에서는 빈번한 재배치가 발생하여 성능이 좋지 않다는 사실에 기인한다.

그림 7은 DSMList를 기존의 MList와 비교하여 실행시간 면에서 향상된 백분율을 나타낸다. 모든 분포와 큐 크기에서 MList에 비해 매우 향상된 결과를 가져온다. 가장 적은 향상율을 보이는 Tri 분포에서 조차 평균 38%의 향상율을 보이고 Pareto 분포에서는 100%의 향상율을 나타낸다.

5. 결 론

시뮬레이션 환경이 크고 복잡해질수록 더 많은 이벤트들이 발생되고 처리되는 과정에서 시뮬레이션 엔진에 많은 부하를 주게 된다. 시뮬레이션 프로젝트에서 모델의 실행시간이 중요한 요소라면 엔진의 미래사건 리스트들을 관리하는 우선순위 큐의 자료구조에 대한 신중한 선택이 요구된다. 본 논문에서는 이산사건 시뮬레이션의 사건 리스트 관리에 널리 사용되는 칼렌다 큐와 MList의 구조 및 단점에 대해 알아보고 MList의 단점을 개선하기 위해 DSMList를 소개하였다. 기존의 MList는 칼렌다 큐의 단점을 보완하기 위해 내부구조를 3개의 층으로 나누었으나 T2에 사건의 수가 0이 될 때 까지는 T3에서 T2로 이동작업이 발생하지 않는다. 그러므로 계속해서 삽입 만 발생하여 T3에 많은 사건들이 쌓이게 된다면 후에 삭제가 발생하여 T3에서 T2로 사건 이동 작업 시 처리에 부하를 주게 된다. 제안된 DSMList에서는 T3에 정해진 수(T2_DynamicShift)의 사건이 쌓이게 되면 자동적으로 새로운 칼렌다를 생성하여 T2로 이동시킨다. 성능 분석을 위한 실험 결과는 사건의 타임 스템프를 생성하는 랜덤 분포함수에 따라 차이는 있지만 기존 MList와 비교하여 실행시간 면에서 최소 20% 이상의 향상을 보여주는 매우 안정적인 결과를 가져왔다.

DSMList를 효과적으로 이용하기 위해서는 자동 쉬프트를 수행하기 위한 조건인 T2_DynamicShift의 적절한 값을 결정하여야 한다. 본 연구의 실험 결과로서는 이 값이 최대 큐 크기의 0.1배 정도일 때 바람직한 성능을 가져오는 것으로 파악된다. 실제 시뮬레이션 환경에서 이 같은 결과를 이용하기 위해서는 시뮬레이션 대상이 되는 모델에서의 최대 큐 크기를 사전에 알 수 있다는 가정이 수반되어야 한다. 이러한 가정을 극복하기 위하여 T2_DynamicShift의 값을 자동적으로 할당하고, 시뮬레이션 실행

표 4. Up-Down Model 하에서의 실험 결과

분포	큐 구조	큐 크기										평균
		1000	2000	3000	4000	5000	6000	7000	8000	9000	10000	
Uni (0,100)	DSMList	31 (67%)	94 (57%)	141 (53%)	203 (64%)	266 (60%)	312 (60%)	407 (56%)	469 (55%)	515 (57%)	610 (55%)	(58%)
	MList	63 (33%)	125 (43%)	218 (27%)	360 (36%)	453 (33%)	594 (24%)	734 (20%)	859 (17%)	985 (17%)	1140 (16%)	(27%)
	CQ	94	219	297	562	672	781	922	1031	1187	1360	
Tri (0,1.5)	DSMList	47 (70%)	94 (78%)	156 (71%)	219 (80%)	297 (78%)	344 (81%)	422 (83%)	500 (83%)	578 (86%)	641 (88%)	(80%)
	MList	62 (60%)	125 (70%)	235 (56%)	343 (69%)	485 (64%)	609 (65%)	735 (70%)	875 (71%)	1062 (74%)	1172 (78%)	(68%)
	CQ	156	422	531	1094	1344	1765	2438	3000	4032	5281	
Neg Tri (0,1000)	DSMList	47 (57%)	110 (53%)	156 (50%)	219 (64%)	297 (59%)	344 (58%)	437 (56%)	500 (55%)	562 (55%)	657 (53%)	(56%)
	MList	47 (57%)	141 (40%)	234 (25%)	360 (41%)	468 (35%)	625 (25%)	735 (27%)	890 (20%)	1000 (20%)	1172 (17%)	(31%)
	CQ	110	234	312	609	719	828	1000	1110	1250	1406	
Bimodal	DSMList	47 (57%)	94 (60%)	141 (53%)	203 (65%)	266 (60%)	328 (58%)	391 (58%)	484 (54%)	532 (55%)	594 (57%)	(58%)
	MList	62 (43%)	125 (47%)	234 (21%)	344 (40%)	469 (30%)	594 (24%)	734 (22%)	859 (18%)	1047 (12%)	1172 (16%)	(27%)
	CQ	109	234	297	578	672	781	938	1047	1187	1391	
Exp (1.0)	DSMList	47 (50%)	110 (61%)	172 (50%)	235 (65%)	312 (60%)	391 (59%)	484 (54%)	563 (53%)	656 (52%)	703 (54%)	(56%)
	MList	78 (17%)	141 (50%)	219 (36%)	344 (49%)	453 (42%)	562 (41%)	688 (34%)	797 (34%)	906 (33%)	1047 (31%)	(37%)
	CQ	94	281	344	672	781	953	1047	1203	1360	1515	
Exp Mix	DSMList	78 (45%)	141 (59%)	234 (48%)	313 (67%)	390 (64%)	500 (60%)	610 (56%)	703 (57%)	797 (55%)	891 (55%)	(57%)
	MList	78 (45%)	172 (50%)	281 (38%)	406 (57%)	516 (52%)	656 (48%)	797 (43%)	922 (43%)	1062 (40%)	1219 (38%)	(45%)
	CQ	141	344	453	954	1078	1250	1391	1625	1781	1969	
Pareto (1.0)	DSMList	78 (28%)	297 (0%)	375 (0%)	782 (-4%)	609 (32%)	766 (29%)	875 (24%)	797 (40%)	891 (41%)	1843 (-10%)	(18%)
	MList	1969 (-1706%)	8109 (-2630%)	20891 (-5471%)	47250 (-6200%)	115141 (-12837%)	251625 (-23242%)	410734 (-35400%)	767469 (-57601%)	925187 (-61579%)	1361766 (-81345%)	(-28810%)
	CQ	109	297	375	750	890	1078	1157	1328	1500	1672	

중 이 값을 상황에 따라 조절하는 방안에 대한 연구가 향후 수행되어야 할 것이다.

참 고 문 헌

1. 임동순 (2008), "이산사건 시뮬레이션에서의 우선순위 큐 성능 분석을 위한 다단계 프로세스 모델: 창조 모델에 대한 사례연구", *한국시뮬레이션학회지*, 제17권, 제4호, pp. 61-69.
2. Brown, R. (1988), "calendar queues: A fast O(1) priority queue implementation for the simulation event set problem", *Comm. of the ACM*, Vol. 24, No. 12, pp. 825-829.
3. Chung, K., Sang, J. and Rego, V. (1993), "A performance comparison of event calendar algorithms: an empirical approach", *Software-Practice and Experience*, Vol. 23, No. 10, pp. 1107-1138.
4. Goh, R.S.M. and Thng, I.L. (2003), "MLIST: An Efficient Pending Event Set Structure for Discrete Event Simulation", *International J. of Simulation*, Vol. 4, No. 5/6, pp. 66-77.
5. Goh, R.S.M. and Thng, I.L. (2004), "An improved dynamic MLIST for the pending event set problem", *International J. of Simulation*, Vol.5, No.3/4, pp.26-36.

6. Henriksen, J. O. (1977), “An improved event list algorithm”, *Proceedings of the 1977 Winter Simulation Conference*, pp. 547-557.
7. Jones, D. W. (1986), “An empirical comparison of priority-queue and event-set implementations”, *Comm. of the ACM*, Vol. 29, No. 4, pp. 300-311.
8. McCormack, W. M. and Sargent, R. G. (1981), “Analysis of future event-set algorithms for discrete event simulation, *Comm. of the ACM*, Vol. 24, No. 12, pp. 801-812.
9. Ronngren, R. and Ayani, R. E. (1997), “Parallel and sequential priority queue algorithms”, *ACM Trans. On Modeling and Computer Simulation*, Vol. 2, pp. 157-209.

김 성 곤 (movien@hanmail.net)
2006 대구가톨릭대학교 정보통신공학과 공학사
2009 한남대학교 산업공학과 공학석사
2006 OPTRONTEC Corp.
2009～현재 DENSO PS ELECTRONICS Corp.

관심분야 : 모델링&시뮬레이션, MES, APS, POP

임 동 순 (dsyim@hnu.kr)
1983 한양대학교 산업공학과 공학사
1986 한국과학기술원 산업공학과 공학석사
1991 아이오와주립대학교 산업공학과 공학박사
1992～현재 한남대학교 산업경영공학과 교수

관심분야 : 모델링&시뮬레이션, 네트워크 알고리즘