확률적 배낭 문제들에 대한 적응적 정책: 병렬 롤아웃 접근*

최상희[O1] 장형수[1]
[1]서강대학교 컴퓨터공학과
cpof36[O]@sogang.ac.kr, hschang@sogang.ac.kr

An Adaptive Policy for Stochastic Knapsack Problems: Parallel Rollout Approach

Sang Hee Choi[O1] Hyeong Soo Chang[1]
[1]Department of Computer Science and Engineering, Sogang University

요 약

본 논문에서는 "확률적 배낭(stochastic knapsack)" 문제에 대하여 잘 알려진 Complete Sharing(CS), Complete Partitioning(CP)이 트래픽 특성에 따라 성능의 차이가 나타난다는 약점과 각 정책(policy)들의 평균 성능에 대한 최적 파라미터들을 트래픽 특성에 따라 결정하여야 한다는 약점을 보완하는 모델에 근거한 "Parallel Rollout(PR)"에 기초한 적응적인 정책(adaptive policy)을 제안한다. 주어진 트래픽 모델을 병합한 확률적 배낭 문제를 마코프 결정 과정(Markov Decision Process, MDP)으로 모델링하고, 마코프 결정 과정상에서 기존의 주어진 정책들을 PR 기법을 적용, 하나의 정책으로 융합하고 그 정책이 기존의 주어진 어떤 정책보다도 성능이 같거나 더 뛰어나다는 이론적 근거를 실험을 통하여 확인한다.

1. 서 론

본 논문에서는 "확률적 배낭(stochastic knapsack)" 문제[1]에 대해 잘 알려진 구조적 정책(structured policy)들, Complete Sharing (CS)[2,3] 및 Complete Partitioning (CP)[2,3]이 트래픽의 특성에 따라 성능의 차이가 난다는 약점과 각 정책(policy)들의 최적 파라미터들이 트래픽의 특성에 따라 결정되어야 한다는 약점을 보완하기 위한 모델에 근거한 "Parallel Rollout(PR)"에 기초한 적응적인 정책을 제안한다. 이 정책은 현재 상태(state)에서 취할 수 있는 각 행동(action)의 유틸리티(utility)를 주어진 트래픽 모델을 통해 기존의 주어진 구조적 정책들을 시뮬레이션(simulation)하여 측정하고, 가장 좋은 행동 유틸리티(action utility)를 갖는 행동을 선택함으로서 적응성을 달성한다.

확률적 배낭 문제를 간단히 정의하면 다음과 같다. 크기가 C인 배낭(knapsack)은 K개의 클래스(class)로 나뉘는 객체(object)들을 수용한다. b_k 크기를 갖는 클래스-k 객체들은 λ_k rate를 갖고, 포아송 과정에 따라 확률적으로 배낭에 도착한다. 본 논문에서 객체들은 콜(call)단위가 아닌 패킷(packet)단위로 고려되고, 따라서 배낭에는 다수의 객체들이 도착하게 된다. 객체들의 서비스 시간은 exponentially distributed되어 있고, 평균 서비스 시간은 $1/\mu_k$이다. 수용(admit)된 클래스-k 객체들은 배낭에서 서비스를 받고 보상(revenue) r_k를 얻지만, 거절(배낭에 들어가지 못한, reject)된 클래스-k 객체들은 비용(cost) c_k를 발생한다. 이 문제의 목적은 infinite horizon(또는 시간)에 대하여 total expected discounted 보상을 최대화하거나 total expected discounted 비용을 최소화하기 위해 배낭에 도착하는 객체들을 수용/거절하는 정책을 찾는 것이다. 본 논문에서는 최소화 문제를 고려한다.

* 이 논문은 2007년도 정부(과학기술부)의 재원으로 한국 과학재단의 지원을 받아 수행된 연구임(No. R01-2007-000-10511-0).

확률적 배낭 문제를 풀기위한 대표적인 구조적 정책에는 CS와 CP가 있다. 이 정책들은 두 가지 약점이 있다. 첫 번째, 트래픽 특성에 따라 각 정책의 성능이 달라진다. CS는 대략 트래픽 로드(load)가 적은 경우에 잘 동작하지만, CP는 트래픽 로드가 큰 경우에 CS에 비해 상대적으로 "적절한" 파티션(partition)에 의해 나은 성능을 보인다[2,3]. 두 번째, 이런 정책들은 높은 성능을 보장하기 위해서 트래픽 특성에 따른 최적 파라미터들을 결정해야 한다[2,3]. 특히 CP는 트래픽 특성에 따라 최적 파티션의 크기가 결정되어야 하고 트래픽 모델이 복잡하면 파라미터를 결정하기가 쉽지 않다.

본 논문에서는 CS, CP 각각에 유리한 트래픽 특성이 주기적으로 번갈아가며 나타나는 "진동하는(oscillating)" 트래픽을 생성하는, 각 클래스마다 2개의 MMBP를 융합한 트래픽 모델을 통하여 이러한 정책들의 약점을 실험적으로 보이고, 이런 트래픽을 생성하는 트래픽 모델을 병합한 확률적 배낭 문제를 MDP로 모델링하여 MDP상에서 시뮬레이션을 사용하는 PR[4]을 적용한다. "PR"은 단일(single) 정책에 대해 향상된 정책을 찾는 롤아웃(rollout) 알고리즘[5,6]을 확장한 알고리즘으로 다수의 정책들에 대해 향상된 융합 정책을 찾는다. PR을 적용하여 찾은 정책은 다수의 정책들의 강점을 융합한 효과를 얻을 수 있기 때문에 본 논문에서 제안하고자 하는 "PR"에 기초한 적응적인 정책은 트래픽 특성에 따라 각 정책들의 성능이 달라지는 약점을 보완할 수 있다. 이 정책을 찾는 대략적 과정은 다음과 같다. 주어진 상태에서 취할 수 있는 각 행동을 취하고, 도달한 다음 상태에서 트래픽 모델에 의해 생성된 샘플(sampled) 트래픽에 대해 다수의 정책에 대한 유틸리티를 측정한다. 측정된 결과 중 가장 작은 유틸리티 값과 각 행동의 직접적인(immediate) 비용과 병합한 값을 행동 유틸리티 값으로 기억하고, 행동 유틸리티 값 중 가장 작은 값을 갖는 행동을 현재 상태에서 취함으로서 시뮬레이션에 근거한 적응적인 정책(adaptive policy)을 갖게 된다. 이 정책의 성능은 이론적으로 증명되었고[4], 이를 7장의 실험을 통하여 확인할 수 있다.

본 논문의 구성은 다음과 같다. 2장에서 확률적 배낭 문제에

대한 MDP formulation을 하고, 3장에서는 대표적인 구조적 정책인 CS, CP에 대한 간단한 소개와 MDP formulation을 한다. 4장에서는 시뮬레이션 트래픽 모델을 정의하고, 5장에서는 각 구조적 정책이 트래픽 특성에 따라 다른 성능을 보임을 시뮬레이션 트래픽 모델에 의해 생성된 트래픽을 사용하여 실험적으로 보인다. 6장에서는 PR에 대해 소개를 하고, 7장에서는 실험결과를 통해 "PR"에 기초한 적응적인 정책의 성능을 보인다.

2. 문제 정의

확률적 배낭 문제의 정의는 앞에서 간단히 살펴보았다. 우리는 확률적 배낭 문제에 대한 두 가지 목적 중에서 total expected discounted 비용을 최소화하는 정책을 찾기 위한 확률적 배낭 문제를 MDP로 모델링한다.

2.1 MDP formulation

확률적 배낭 문제는 discrete time $t(t\in\{0,1,2,\cdots\})$에 대해서 정의된다. MDP로의 모델링은 총 4개의 요소들이 정의되어야 한다. 각각은 상태들의 유한 집합 X, 행동들의 유한 집합 A, 상태 전이(transition) 함수 P, 비용 함수 R이고, 지금부터 각 요소들의 모델링 결과를 설명한다.

배낭의 상태로서 가능한 모든 상태들의 유한 집합을 X로 나타낸다. 시간 t에서 배낭의 상태는 4개의 요소들로 구성되고, $x_t=(n_t,y_t,h_t,s_t)$, $x_t\in X$와 같이 표현한다. 상태 구성요소 중 n_t는 시간 t에 배낭 안에 있는 객체들을 나타낸다.

$n_t=(n_{t1},n_{t2},\cdots,n_{tK})$, $n_{tk}\in\{0,1,2,\cdots,\lfloor C/b_k\rfloor\}$, $k=1,2,\cdots,K$.

n_{tk}를 배낭에 있는 클래스-k 객체들의 개수라고 하면, 각 클래스에 대해 객체들의 개수와 크기를 곱한 값의 총합은 배낭의 크기 C를 넘지 않아야 한다:

$$\sum_{k=1}^{K} n_{tk}b_k \le C.$$

배낭에는 시간 간격(time interval)마다 트래픽 모델에 의해 생성된 객체들이 도착(incoming)하는데, 이렇게 트래픽 모델에 의해 시간 간격 $[t-1,t)$동안 생성되는 객체들을 y_t로 나타낸다.

$y_t=(y_{t1},y_{t2},\cdots,y_{tK})$, $y_{tk}\in\{0,1,2,\cdots,N_{\max}\}$, $k=1,2,\cdots,K$.

이 때, 시간 간격 $[t-1,t)$ 동안 트래픽 모델에 의해서 생성된 클래스-k 객체들의 개수를 y_{tk}로 나타내고, 각 시간 간격 동안 생성 가능한 객체들의 최대 개수는 $N_{\max}$로 나타낸다.

시간 t에 도착하는 객체의 수 y_t는 시간 간격 $[t-1,\ t)$동안 트래픽 모델의 상태에 따라 결정되는데, 이 때, 클래스-k에 대한 트래픽 모델의 상태를 s_{tk}라고 하면, 트래픽 모델의 상태 s_t는 다음과 같다.

$$s_t=(s_{t1},s_{t2},\cdots,s_{tK}),\ k=1,2,\cdots,K.$$

트래픽 모델에 의해 생성된 객체들은 배낭에 도착하면 수용/거절되는데 수용된 각 클래스 객체들은 배낭에서 얼마나 오랫동안 서비스를 받을지 결정해야 한다. 서비스 시간 h_t는 배낭에 수용되어 서비스를 받거나 받기 시작한 객체들이 서비스가 종료될 때까지 얼마나 많은 시간이 남았는지 나타낸다. 이는 exponentially distributed 되어 있고, 클래스에 관계없이 평균 서비스 시간 $1/\mu_k$을 갖는다. 또한 각 객체들의 서비스 시간은 시간 간격마다 1씩 감소한다. 클래스-k의 각 객체들의 남은 서비스 시간을 h_{tk}로 나타내면, h_t는 다음과 같다.

$h_t=(h_{t1},h_{t2},\cdots,h_{tK})$, $h_{tk}=(h_{tk}^1,h_{tk}^2,\cdots,h_{tk}^{B_k})$, $k=1,2,\cdots,K$.,

이 때, h_{tk}^{γ}는 음이 아닌 정수로 클래스-k의 γ번째 객체의 남은 서비스 시간을 나타내고, 클래스 k에 대해 배낭에서 최대로 받아들일 수 있는 객체의 개수를 $B_k=C/b_k$로 정의한다.

행동은 배낭에 도착한 객체들을 얼마나 수용/거절할 것인지를 나타내고, n_t, y_t, 그리고 서비스가 종료된 객체들을 고려하여 결정된다. 여기서 서비스가 종료된 객체들은 서비스 시간이 0이 된 객체들을 말하며, 서비스 시간 h_t에 의해 결정된다. 이 때, 서비스가 종료된 클래스-k 객체들의 개수를 z_{tk}라 하면, 서비스가 종료된 객체들은 z_t로 나타내고, 다음과 같다.

$$z_t=(z_{t1},z_{t2},\cdots,z_{tK}),\ k=1,2,\cdots,K.$$

도착한 객체들에 대해 선택 가능한 모든 행동들의 집합을 A라 정의하고, 주어진 상태 x_t, $x_t\in X$에서 선택할 수 있는 행동들의 집합을 $A(x_t)\subseteq A$라고 정의하면, 행동 $a_t\in A(x_t)$는 다음과 같이 정의된다.

$$a_t=(a_{t1},a_{t2},\cdots,a_{tK}),\ 0\le a_{tk}\le y_{tk},\ k=1,2,\cdots,K.$$

하지만 행동 a_t를 취했을 때 배낭 안의 객체들의 수는 배낭의 크기 C를 초과해서는 안 되므로 행동 a_t는 다음의 조건을 만족해야한다:

$$\sum_{k=1}^{K}(n_{tk}-z_{tk}+a_{tk})b_k \le C.$$

주어진 상태에서 어떤 행동을 취할지 결정되면, 수용되지 못한 클래스-k 객체들은 거절되고, 거절된 객체들은 그에 대해 비용 c_k를 발생한다. 형식적으로, 상태 x_t에서 배낭에 도착한 클래스-k 객체 수가 y_{tk}이고, a_{tk}개의 클래스-k 객체들을 수용하면, $y_{tk}-a_{tk}$개의 클래스-k 객체들은 거절하게 되고 발생한 비용은 $(y_{tk}-a_{tk})c_k$이다. 하지만 비용은 모든 클래스에 대해 고려되어야 하므로 상태 x_t에서 행동 a_t을 취해 상태 x_{t+1}로 전이 될 때, 발생한 총 비용은 다음과 같다:

$$c(x_t,a_t,x_{t+1})=\sum_{k=1}^{K}(y_{tk}-a_{tk})c_k.$$

주어진 상태에서 행동을 취하면 확률에 의해 상태 전이가 발생한다. 이 확률은 시간 간격동안 트래픽 모델에 의해서 얼마나 많은 객체들이 생성 되었는지, 트래픽 모델의 상태가 어떻게 바뀌는지에 따라 정의된다. 확률을 정의하기 전에 전이 후의 다음 상태에 대해서 먼저 정의한다. 상태 x_t에서 행동 a_t를 취하여 x_{t+1}로 전이했을 때, 다음 상태 x_{t+1}은 다음과 같다.

$$x_{t+1}=(n_{t+1},\ y_{t+1},\ h_{t+1},\ s_{t+1}),$$
$$n_{t+1\,k}=\max(n_{tk}-z_{tk}+a_{tk},\ 0),\ k=1,2,\cdots,K.$$

만약 상태 x_t에서 트래픽 모델의 상태가 s_{tk}이고, 다음 상태 x_{t+1}에서 트래픽 모델의 상태가 $s_{t+1\,k}$에 있으며, 트래픽 모델에 의해 $y_{t+1\,k}$개의 객체가 생성되어 도착하면, 상태 전이 확률은 다음과 같이 정의된다:

$$P(x_{t+1}|x_t,a_t)=\prod_{k=1}^{K}\left(p_{s_{tk}s_{t+1\,k}}P_{s_{tk}}(y_{t+1\,k})\right),\ a_t\in A(x_t),\ x_t,x_{t+1}\in X.$$

이 때, $p_{s_{tk}s_{t+1\,k}}$는 트래픽 모델의 상태 s_{tk}에서 $s_{t+1\,k}$로 전이할 확률을 의미하고, $P_{s_{tk}}(y_{t+1\,k})$는 트래픽 모델의 상태 s_{tk}에서 $y_{t+1\,k}$개의 객체를 생성할 확률을 의미한다.

배낭에 도착하는 객체들을 얼마나 받아들일지 결정하는 것은 이 문제에서 가장 중요하다. 이는 정책에 의해 결정되는데, 정책은 상태 집합 X에서 행동 집합 A로의 함수의 시퀀스(sequence)이고, π로 나타낸다. Horizon(또는 시간) H에 대한 정책 π는 다음과 같이 정의된다:

$\pi=\{\pi_0,\pi_1,\pi_2,\cdots,\pi_H\}$, $\pi_t: X\to A$ with $\pi_t(x)\in A(x)$, $\forall x\in X$.

그리고 $H=\infty$(infinite horizon)일 때, 정책 π는 다음과 같이 정의된다:

$\pi=\{\pi_0,\pi_1,\pi_2,\cdots\}$, $\pi_t: X\to A$ with $\pi_t(x)\in A(x)$, $\forall x\in X$.

모든 정책들의 집합을 Π라고 정의하고, infinite horizon에 대

해 정책 π, $\pi \in \Pi$가 주어져 있다고 하자. X_t를 시간 t에서 상태를 나타내는 확률변수라고 하면, 주어진 정책 π에 따라 매 결정시간에서 배낭에 도착한 클래스 객체들을 거절함으로서 발생하는 총 비용의 기대 값은 다음과 같다.

$$V_\infty^\pi(X_0) = E\left[\sum_{t=0}^{\infty}\gamma^t c(X_t,\pi(X_t),X_{t+1})\right].$$

그리고 V_∞^*는 다음과 같이 정의된다.

$$V_\infty^*(X_0) = \min_{\pi\in\Pi} V_\infty^\pi(X_0).$$

이 확률적 배낭 문제의 목적은 infinite horizon에 대하여 매 결정시간에 발생하는 비용의 총합을 최소화 하는 최적의 (optimal) 정책을 찾는 것이고, 이러한 최적의 정책은 다음과 같이 결정된다[7]:

$$\pi^*(X_0) \in \arg\min_{a\in A(X_0)}\left\{\bar{c}(X_0,a) + \gamma\sum_{X_1} P(X_1|X_0,a)\, V_\infty^*(X_1)\right\}$$

$$, \ \bar{c}(X_0,a) = E_{X_1}[c(X_0,a,X_1)]. \quad (1)$$

최적의 정책은 value iteration과 policy iteration을 이용하여 찾을 수 있다[7]. 하지만, 상태 공간의 크기 또는 행동 집합의 크기가 큰 경우에는 엄청난 시간과 계산의 양을 요구하기 때문에 디멘젼의 저주가 발생한다. 이런 경우에는 빠른 처리와 함께 높은 성능을 요구하는 네트워크 문제에는 적용하기 쉽지 않다.

3. 다양한 구조적 정책들

확률적 배낭 문제에 대해 잘 알려진 많은 구조적 정책들이 있다. 특히 구현과 계산이 간단하고, 뛰어난 성능을 보이는 대표적인 정책은 CS이고, 이와 더불어 CP도 잘 알려져 있다. 3장은 CS, CP의 간단한 소개와 그에 대한 MDP formulation을 한다. 단, 각 정책의 formulation에서 각 클래스 객체들은 오름차순으로 고려한다(오름차순으로 고려하는 이유는 비용의 값이 내림차순으로 정렬되어 있으므로, 클래스가 낮을수록 우선순위가 높은 클래스기 때문이다).

3.1 Complete Sharing

CS는 클래스 객체들이 배낭에 도착했을 때 그것들을 수용할 수 있는 충분한 공간이 있다면 가능한 많은 객체들을 수용하는 정책이다. CS는 트래픽의 로드가 적을 때 유리한데, 그 이유는 배낭에 매 시간 여유 공간이 확보되므로 가능한 많은 객체들을 수용할 수 있기 때문이다. 클래스 k에 대해 $E_t^{CS}(k,\alpha)$를 $C-\left(\sum_{i=1}^{K}(n_{ti}-z_{ti})b_i + \sum_{i=1}^{k-1}a_{ti}b_i + \alpha b_k\right)$라 하자. 이 식에서 괄호 안의 첫 번째 term은 서비스가 종료된 객체들을 제외하고 배낭에 남아있는 객체들이 차지하는 크기를 의미하고, 두 번째 term은 클래스-k 객체들 보다 먼저 CS에 의해 행동으로 결정된 (오름차순으로 고려하기 때문에 k보다 작은 클래스들을 고려한다) 클래스 객체들이 배낭에서 차지하는 크기를 의미하며, 마지막 term은 도착한 클래스-k 객체들을 배낭에 얼마나 수용할지 결정한다. 이 때, 집합 $N_{k,t}^{CS}$는 다음과 같다:

$$N_{k,t}^{CS} = \{\alpha | E_t^{CS}(k,\alpha) \geq 0\}.$$

집합 $N_{k,t}^{CS}$의 원소 α에 대해 클래스-k 객체들에 대한 행동 a_{tk}는 다음과 같이 결정되고, 행동 $a_t = (a_{t1}, a_{t2}, \cdots, a_{tK})$를 구성한다:

$$a_{tk} = \arg\min_{\alpha\in N_{k,t}^{CS}} E_t^{CS}(k,\alpha).$$

3.2 Complete partitioning

CP에서 배낭은 각 클래스의 파티션으로 나뉜다. CP는 도착한 클래스 객체들에 대해 해당 클래스 파티션에 충분한 공간이 있는 경우, 가능한 많은 객체들을 수용하는 정책이다. 트래픽의 로드가 큰 경우에 CS보다 상대적으로 유리한데, 그 이유는 각 클래스 객체에 대해 해당 클래스의 파티션 크기만큼의 서비스를 보장하기 때문이다. 각 클래스 파티션들의 총합은 배낭의 크기 C와 같고, 클래스-k 파티션 크기를 P_k로 표현하면 다음을 만족한다:

$$\sum_{k=1}^{K} P_k = C.$$

클래스 k에 대해, $E_t^{CP}(k,\alpha)$를 $P_k - \{(n_{tk}-z_{tk}+\alpha)b_k\}$라고 정의하자. 이 식에서 중괄호 안의 식은 서비스가 종료된 객체들을 제외하고 배낭에 남아있는 객체들에 대해 α개의 객체들을 수용한 후 남은 파티션의 크기를 나타낸다. 이 때, 각 클래스-k에 대한 집합 $N_{k,t}^{CP}$는 다음과 같다:

$$N_{k,t}^{CP} = \{\alpha | E_t^{CP}(k,\alpha) \geq 0\}.$$

집합 $N_{k,t}^{CP}$의 원소 α에 대해 클래스-k 객체들에 대한 행동 a_{tk}는 다음과 같이 결정되고, 행동 $a_t = (a_{t1}, a_{t2}, \cdots, a_{tK})$를 구성한다:

$$a_{tk} = \arg\min_{\alpha\in N_{k,t}^{CP}} E_t^{CP}(k,\alpha).$$

4. 시뮬레이션 트래픽 모델

구조적 정책들은 트래픽 특성에 따라 상이한 성능을 보인다. 만약 각 정책이 뛰어난 성능을 보이는 트래픽이 번갈아 생성된다면, 그 트래픽은 어떤 정책에 대해서도 다른 정책들과 비교해 우수한 성능을 보인다고 할 수 없다. 이 장에서는 이와 같은 트래픽을 생성하는 트래픽 모델에 대해 설명한다.

우리는 각 클래스마다 Markov Modulated Bernoulli Process(MMBP) 2개를 결합하여 트래픽 모델을 구성한다. 클래스-k MMBP를 M_m^k ($m \in \{1,2\}$)라고 하면, m값이 같은 각 클래스 MMBP는 하나의 그룹을 이룬다. 결과적으로 트래픽 모델은 두 개의 그룹으로 구성 되는데, 각 그룹은 CS, CP에 유리한 트래픽을 생성하게 된다. 또한 각 클래스의 MMBP간의 전이를 정의하여 두 가지 정책에 유리한 트래픽을 번갈아서 생성한다.

각 클래스 MMBP는 3개의 상태를 갖는다. 3가지 상태는 객체들을 많이 생성하는 상태(상태 1), 객체들을 적게 생성하는 상태(상태 2), 그리고 객체들을 상태 1과 상태 2의 중간 정도 양의 객체들을 생성하는 상태(상태 3)로 구성되고, 클래스-k MMBP M_m^k의 3가지 상태는 $S_1^{k,m}$, $S_2^{k,m}$, $S_3^{k,m}$으로 나타낸다. 이 때, 클래스-k MMBP 상태 $S_i^{k,m}$에서 상태 $S_j^{k,m}$로의 전이 확률을 $p_{S_i^{k,m}S_j^{k,m}}$, $i,j \in \{1,2,3\}$, 로 나타내고. 같은 상태로 전이할 확률(self transition probability)은 $p_{S_i^{k,m}S_i^{k,m}}$로 나타낸다.

각 클래스-k MMBP의 각 상태에서는 최대 $N_{\max}$개의 객체들을 생성할 수 있기 때문에 각 상태에서 생성된 객체의 수를 $\alpha \in \{0,1,\cdots,N_{\max}\}$로 나타낸다. 이 때, 클래스-$k$ MMBP M_m^k의 상태 $S_i^{k,m}$에서 하나의 객체를 생성할 확률을 $q_{S_i^{k,m}}$라 하면, α개의 객체를 생성할 확률은 다음과 같이 정의된다:

$$P_{S_i^{k,m}}(\alpha) = \binom{N_{\max}}{\alpha}(q_{S_i^{k,m}})^\alpha (1-q_{S_i^{k,m}})^{N_{\max}-\alpha}, \ k = 1,2,\cdots,K.$$

그리고 각 클래스-k MMBP 상태 $S_i^{k,m}$에 대해 다음을 만족한다:

$$\sum_{\alpha=0}^{N_{\max}} P_{S_i^{k,m}}(\alpha) = 1.$$

이 트래픽 모델에서 가장 중요한 것은 객체들을 생성할 확률보다 각 클래스 MMBP간의 전이이다. 그 이유는 2개의

MMBP를 이동하면서 CS, CP에 유리한 트래픽을 번갈아 생성해야 하기 때문이다. 이 전이는 앞서 정의한 그룹 단위로 모든 클래스에서 확률 p_{trans}에 의해 동시에 발생한다. 즉, MMBP간의 전이가 발생하면 모든 클래스에서 동시에 해당 클래스의 다른 MMBP로 전이한다. 전이 후 각 class의 MMBP 상태는 균등한 확률(uniform probability) p_{trans}/[각 클래스 MMBP의 상태의 총 개수]에 의해 결정된다. 이 때, MMBP M_m^k의 상태 $S_i^{k,m}$에서의 전이 확률의 총 합은 1을 만족한다:

$$\sum_{x\in\{S_i^{k,m},S_i^{k,m},S_i^{k,m}\}} p_{S_i^{k,m}x} + p_{trans} = 1.$$

그림 1은 이러한 트래픽 모델을 나타낸다.

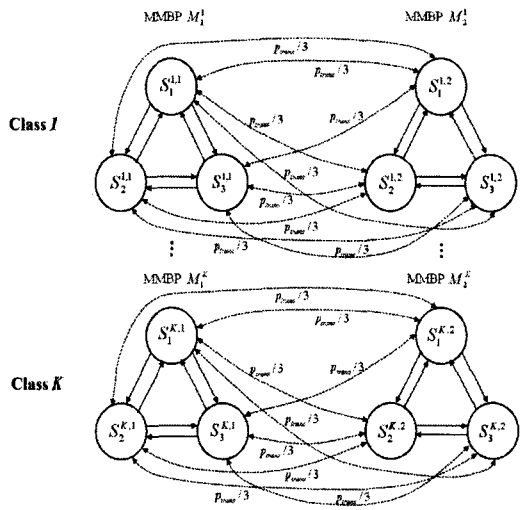

그림 1. Simulation 트래픽 모델

5. "진동하는" 트래픽

실제 네트워크 트래픽은 상황에 따라 급격하게 변하고 다양한 특징을 갖는다. 그리고 이에 따라 구조적 정책들의 성능은 달라진다. 만약 CS, CP에 유리한 트래픽이 번갈아 생성된다면 두 정책 중 어떤 것도 더 나은 성능을 보인다고 할 수 없다. 또한 CP는 파티션 크기를 어떤 비율로 나누느냐에 따라서도 성능이 달라진다. 이렇게 어느 한 정책의 성능이 다른 정책의 성능보다 더 뛰어나지 못하고, 각 정책의 성능이 서로 엇갈리는 상황을 발생시키는 트래픽을 "진동하는" 트래픽이라고 정의한다. 그리고 이 장에서는 앞서 설계한 트래픽 모델을 이용하여 "진동하는" 트래픽이 존재함을 실험적으로 보인다.

트래픽 모델의 첫 번째 그룹은 CS, 두 번째 그룹은 CP가 뛰어난 성능을 보이는 트래픽을 생성한다. 각 클래스-k MMBP 상태의 같은 상태로 전이할 확률 ρ는 $[0.9, 1.0-p_{trans}]$의 범위에서 균등하게 선택(uniform selection)하여 임의로 생성하고, 나머지 2개 상태로의 전이 확률 중 하나는 $[0,\ 1-p_{trans}-\rho]$ 범위에서 균등하게 선택하여 임의로 생성하며 나머지 하나는 앞서 결정된 두 확률을 $1-p_{trans}$에서 빼서 결정한다. 그리고 각 클래스의 MMBP 상태의 $q_{S_i^{k,m}}$은 각 3가지 상태의 특성에 따라 $[0.7, 1.0]$, $[0.0, 0.3]$, $[0.3, 0.7]$의 범위에서 균등하게 선택하여 임의로 생성하고 MMBP간의 전이 확률 p_{trans}는 2×10^{-4}으로 설정한다. 트래픽 로드는 각 클래스 MMBP에 대해 3개의 상태에서 생성하는 d 객체 개수의 기대 값을 계산하고, 안정 상태 확률(steady state probability)을 구하여 기대 값을 계산하면 쉽게 얻을 수 있다.

실험은 두 가지 경우로 나누어 이루어진다. 첫 번째 실험은 클래스는 2개, 배낭의 크기는 6인 경우고 두 번째 실험은 클래스가 4개, 배낭의 크기가 20인 경우이다. 두 번째 실험은 [8]에서 사용된 조건을 그대로 사용하였다. 지금부터 모든 실험에서 $\gamma=1$, $b_k=1$로 설정한다.

그림2, 3은 각각 첫 번째, 두 번째 실험결과이다. 그림 2는 2개의 클래스에 대한 비용이 각 객체마다 10^{-1}, 10^{-3}이고 CP의 파티션의 비율이 5:1인 상황에 대한 실험이고, 그림 3은 4개의 클래스에 대한 비용이 각 객체마다 10^{-1}, 10^{-3}, 10^{-5}, 10^{-7}이고, CP의 파티션의 비율이 14:3:2:1인 실험이다. 그리고 그림 4는 2개의 클래스에 대한 비용이 각 객체마다 10^{-1}, 10^{-3}이고, 2개의 CP에 대한 파티션의 비율이 각각 5:1, 4:2인 실험이며, 그림 5는 4개의 클래스에 대한 비용이 각 객체마다 10^{-1}, 10^{-3}, 10^{-5}, 10^{-7}이고, 3개의 CP에 대한 파티션의 비율이 각각 14:3:2:1, 10:6:3:1, 6:6:6:2인 실험이다.

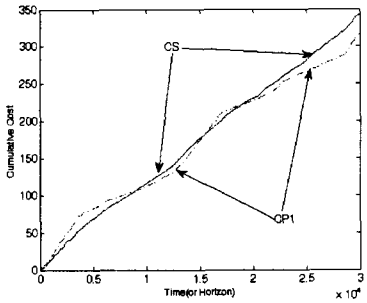

그림 2. "진동하는" 트래픽에 대한 CS, CP 성능(2 클래스)

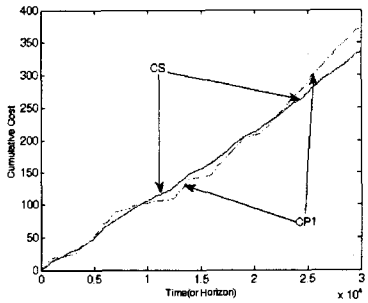

그림 3. "진동하는" 트래픽에 대한 CS, CP 성능(4 클래스)

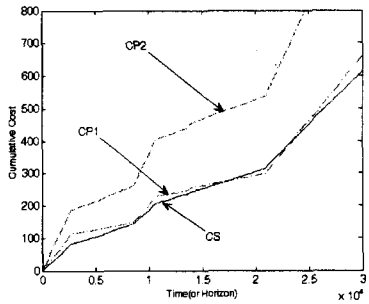

그림 4. "진동하는" 트래픽에 대한 CP 파티션에 따른 성능(2 클래스)

그림 2,3을 보면 알 수 있듯이 트래픽 모델에 의해 생성된 트래픽에 대해 CS, CP의 누적 비용이 교차하여 나타남으로서 CS, CP 중 어떤 정책도 생성되는 트래픽에 대해 유리하지 않음을 보여준다. 이 뿐만 아니라 그림 4,5을 보면, CP의 성능은 클래스 파티션의 비율에 따라 달라짐을 알 수 있다.

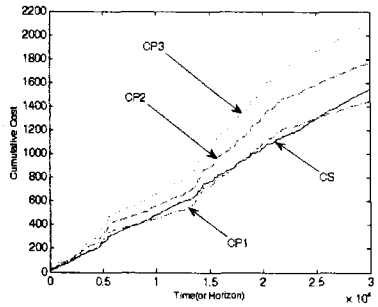

그림 5. "진동하는" 트래픽에 대한 CP 파티션에 따른 성능(4 클래스)

6. Parallel Rollout(PR)

존재하는 모든 트래픽을 고려하여 최적의 정책은 찾는 것은 계산이 매우 복잡하고 엄청난 시간을 소모하는 일이다. 특히 MDP를 이용해서 최적의 정책을 찾는 과정은 디멘젼의 저주로 인해 불가능하다. 따라서 우리는 시뮬레이션에 기초한 접근법인 PR을 사용한다.

PR은 Bertsekas와 Castanon에 의해서 제안된 롤아웃을 기본으로 한다. 이것은 주어진 base heuristic 정책보다 향상된 정책 π를 생성하기 위하여 시뮬레이션을 사용한다. PR은 롤아웃 알고리즘을 확장하여 다수의 base heuristic 정책들에 대해 향상된 정책 π를 찾는 알고리즘으로 이는 주어진 상태에서 취할 수 있는 각 행동에 대한 유틸리티를 측정하는 것에 의해 달성된다. 최적의 정책은 식 (1)을 이용하여 계산할 수 있지만 이 식에서 V_∞^*를 구하는 것은 디멘젼의 저주 때문에 불가능하기 때문에 V_∞^*의 근사 값을 측정해야 한다. 근사 값의 측정은 주어진 다양한 트래픽을 시뮬레이션 하여 계산할 수 있다. 이 과정은 다음과 같다. 먼저 다수의 샘플 트래픽을 생성하고, 주어진 상태에서 해당 상태에서 취할 수 있는 행동을 취한 후 도달한 다음 상태에서 각 트래픽에 대해 다수의 base heuristic 정책들의 유틸리티(total expected cost)를 측정한다. 각 행동에 대해 행동을 취하여 발생한 직접적인 비용과 다수의 정책에 대한 유틸리티의 최소값을 합한 행동 유틸리티 중 가장 작은 값을 기억하고, 각 행동에 대한 행동 유틸리티 값 중 가장 작은 값을 갖는 행동을 취한다. 주어진 상태 x_t에서 행동 $a_t \in A(x_t)$를 취하여 다음 상태 확률 변수 X_{t+1}로 전이했을 때 발생한 직접적인(immediate) 비용을 $c(x_t,a_t,X_{t+1})$, 샘플 트래픽에 대해 sampling horizon H_s동안 측정된 각 정책 π의 성능을 $V_{H_s}^{\pi}$라 하면, 상태 x_t에서 행동 a_t를 취했을 때의 행동 유틸리티 $\tilde{Q}(x_t,a_t)$는 다음과 같이 정의된다:

$$\tilde{Q}(x_t,a_t) = E\{c(x_t,a_t,X_{t+1}) + \min_{\pi\in\Pi} V_{H_s}^{\pi}(X_{t+1})\}.$$

이 때 "PR"에 기초한 적응적인 정책 $\pi_{pr}(x_t)$는 다음과 같이 정의된다:

$$\pi_{pr}(x_t) \in \arg\min_{a_t\in A(x_t)}(\tilde{Q}(x_t,a_t)).$$

이 과정을 자세히 살펴보면, 주어진 상태에서 선택된 행동들은 주어진 샘플 트래픽에 대한 정책의 성능을 비교하여 결정되므로, 트래픽에 적응하고 성능 또한 더 나음을 보장한다. 따라서 이 정책은 여러 정책들이 여러 가지 트래픽에 대해서 나타나는 강점들만을 융합한 정책라고 할 수 있다. PR의 이론적 결과와 구현에 대한 pseudocode는 [4]에 자세히 나와 있다.

7. 실험

실험은 5장에서 했던 트래픽 모델의 설정을 그대로 사용한다. 각 실험은 클래스의 수가 2, 배낭의 크기가 6인 경우, 클래스의 수가 4, 배낭의 크기가 20인 경우에 대해서 "PR" 기초한 적응적인 정책의 성능을 측정한다.

그림 6은 2개의 클래스에 대해 객체 하나의 비용이 10^{-1}, 10^{-2}이고, CP 파티션의 비율이 각각 5:1, 4:2인 실험이다. 그림 7은 그림 4의 실험 기본설정에 "PR" 기초한 적응적인 정책의 성능을 추가한 실험이고, 그림 8,9은 그림 5의 실험 기본설정에 "PR"에 기초한 적응적인 정책의 성능을 추가한 실험이다.

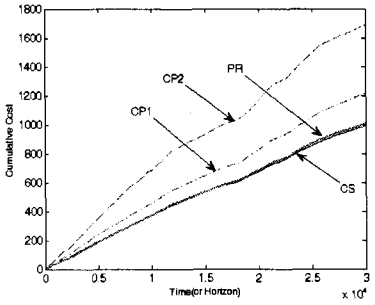

그림 6. "PR"에 기초한 적응적인 정책의 성능(2 클래스)

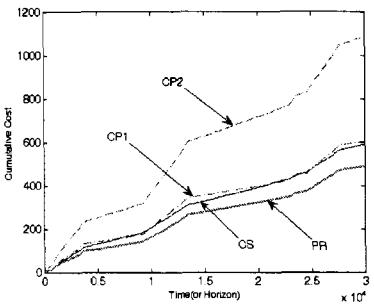

그림 7. "PR"에 기초한 적응적인 정책의 성능(2 클래스)

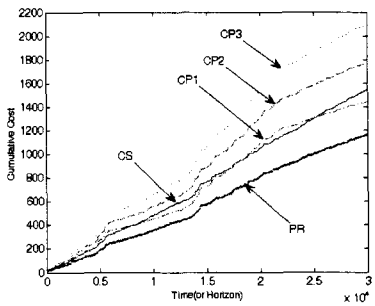

그림 8. "PR"에 기초한 적응적인 정책의 성능(4 클래스)

그림 6을 보면 CS가 CP와 비교해 압도적인 성능을 보이고, "PR"에 기초한 적응적인 정책의 성능이 CS와 거의 비슷한 성

능을 보인다. 따라서 어떤 정책이 트래픽에 대해 지배적인 성능을 보인다면 "PR"에 기초한 적응적인 정책은 그 정책과 비슷한 성능을 보이는 정책임을 알 수 있다.

그림 7에서는 CS, CP 중 어느 하나의 정책이 유리하다고 말할 수 있는 상황이 아니다. 이때는 "PR"에 기초한 적응적인 정책이 두 정책들보다 더 나은 성능을 보인다. 그리고 그림 8,9을 보면 그림 7과 같은 결과를 클래스의 수가 4인 경우에서도 관찰할 수 있다. 결과적으로 생성된 "PR"에 기초한 적응적인 정책은 기존의 구조적 정책인 CS, CP보다 같거나 더 나은 성능을 보임을 알 수 있다.

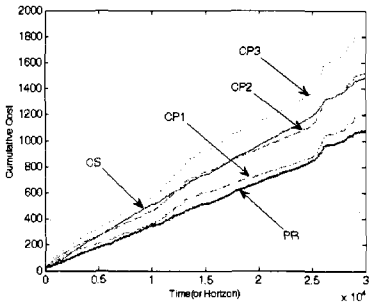

그림 9. "PR"에 기초한 적응적인 정책의 성능(4 클래스)

8. 결 론

많은 구조적 정책들은 트래픽의 특성을 고려하지 않기 때문에 이 정책들은 트래픽의 특성에 따라 다른 성능을 보인다. 또한 최적의 정책을 찾는 것은 상태 공간 또는 행동 집합의 크기가 큰 경우에 계산, 시간적인 측면을 고려해 볼 때 불가능하다. 이를 해결하기 위해 우리는 먼저 두 정책이 높은 성능을 보이는 트래픽을 번갈아 생성하는 트래픽 모델을 통해 "진동하는" 트래픽이 존재함을 보이고, 이 트래픽 모델을 융합한 확률적 배낭 문제를 MDP을 사용해 모델링하였다. 그리고 트래픽 모델에 의해 생성된 샘플 트래픽에 대한 행동 유틸리티를 측정하여 주어진 정책들의 강점만을 융합한 행동을 취하는 "PR"에 기초한 적응적인 정책을 제안하였다. 이 정책의 성능 측정은 두 단계로 이루어졌다. 먼저 각 CS, CP의 성능을 보장할 수 없는 트래픽이 존재함을 보이고, 이 트래픽에 대해 제안된 적응적인 정책의 성능을 측정하였으며, 그 성능은 CS, CP와 같거나 그보다 뛰어난 성능을 보였다.

본 논문에서는 네트워크상의 하나의 큐의 형태가 아닌 다수의 패킷의 형태로 분석과 실험을 하였다. 따라서 서버, 라우터, 자원 할당(resource allocation) 등 더욱 넓고 다양한 분야에 적용이 가능하다. 그리고 QoS를 지원하기 위한 네트워크 프레임워크 중 IntServ의 확장성 문제를 해결하기 위해 각 트래픽을 특성에 따라 다수의 클래스로 그룹화 한 DiffServ[9]에도 적용할 수 있다. DiffServ는 클래스 기반의 프레임워크이므로 이를 구현하는 세부 기술에 본 논문에서 제안한 적응적인 정책을 충분히 적용할 수 있다. 제안된 적응적인 정책을 적용하기 위한 DiffServ의 대표적인 세부 기술로는 Call Admission Control과 Weighted Random Early Detection(WRED)[10], 그리고 Weighted Fair Queuing (WFQ)[11]을 예로 들 수 있다.

일반적으로 서로 다른 트래픽에 대해 다른 성능을 갖는 여러 가지 정책들이 존재하는 다른 문제에 대해서도 그 문제를 MDP로 모델링한다면 더 나은 성능을 보장하는 "PR" 기초한 적응적인 정책을 찾을 수 있을 것이다. 특히, 이 연구의 향후 과제로 확률적 배낭 문제에 대해 최근에 많은 연구가 이루지는 threshold 정책[8]와 reservation 정책[8]에 대해서 각 정책이 높은 성능을 보이는 트래픽을 찾고, 이러한 트래픽을 본 논문에서 제안된 트래픽 모델에 적용한다면, 두 정책이 융합된 "PR"에 기초한 적응적인 정책을 찾을 수 있을 것이다.

참고문헌

[1] K. W. Ross, Danny H. K. Tsang, "The Stochastic Knapsack Problem", *IEEE Trans. Commun.*, Vol. 37, No. 7, pp. 740-747, July, 1989.

[2] J. W. Causey, H. S, Kim, "Comparison of buffer allocation schemes in ATM switches", *IEEE Internation Conference on SUPERCOMM/ICC*, Vol. 2, pp. 1164-1168, May 1994. [2]

[3] B. Epstein, M. Schwartz, "Reservation strategies for multi-media traffic in a wireless environment", *IEEE 45th Vehicular Technology Conference*, Vol. 1, pp. 165-169, July 1995.

[4] H. S. Chang, R. Givan, Edwin K. P. Chong, "Parallel Rollout for Online Solution of Partially Observable Markov Decision Processes", *Discrete Event Dynamic Systems: Theory and Applications*, 14, pp. 309-341, 2004.

[5] D. P. Bertsekas, "Differential Training of Rollout Policies", *Proc. of the 35th Allerton Conference on Communication, Control, and Computing*, Allerton Park IL., 1997.

[6] D. P. Bertsekas, D. A. Castanon, "Rollout algorithms stochastic scheduling problems", *J. of Heuristics*, Vol 5, pp. 89-108, 1990.

[7] M. L. Putterman, *"Markov Decision Process: Discrete Stochastic Dynamic Programming"*, New York, Wiley, 1994.

[8] D. H. K. Tsang, Sekhar Tatikonda, Brahim Bensaou, "optimal and structured Call Admission Control Policies for Resource-Sharing Systems", *IEEE Trans. Commun.*, Vol. 55, No. 1, pp. 158-170, January 2007.

[9] M. Carlson, et al., "An Architecture for Differentiated services", in *Network Working 그룹 - RFC2475*, http://www.ietf.org/rfc/rfc2475.txt, 1998

[10] M. Wurtzler, "Analysis and Simulation of Weighted Random Early Detection(WRED) Queues", *Infomation and Telecommunication Technology Center(ITTC)*, December, 2003.

[11] A. Demers, S. Keshav, S. Shenker, "Analysis and simulation of a Fair Queuing algorithm", *Sysmpsium proceedings on Commun. architechtures & Protocols*, pp. 1-12, 1989.

[12] V. G. Kulkarni, T. E. Tedijanto, "Optimal Admission Control of Markov-Modulated Batch Arrivals to A Finite-Capacity Buffer", *Commun. Statist. - Stochastic Models*, Vol 14, pp. 95-122, 1998.

[13] E. Altman, T. Jimenez, G. Koole, "On Optimal Call Admission Control in a Resource-Sharing System", *IEEE Trans. Commun.*, Vol. 49, No. 9, pp. 1659-1668, September 2001.

[14] K. Mosharaf, J. Talim, I. Lambadaris, "Call admission and fairness control in WDM networks with grooming capabilities", *IEEE Conference on Decision and Control*, Vol 4, pp. 3738-3743, December 2004. 2006.

[15] M. Ghaderi, R. Boutaba, "Call admission control in mobile cellular Networks: a comprehensive survey", *Wireless Commun. & Mobile Computing*, Vol. 6, No. 1, pp. 69-93, February.