사례 기반 결정 이론을 융합한 포텐셜 기반 강화 학습

(Potential-based Reinforcement Learning Combined with Case-based Decision Theory)

김 은 선 † 장 형 수 ††
(Eun Sun Kim) (Hyeong Soo Chang)

요 약 본 논문에서는 다수의 강화 학습 에이전트들의 학습 결과 및 Expert의 지식을 하나의 학습 알고리즘으로 융합하는 강화학습인 "potential-based" reinforcement learning(RL)기법에 불확실한 환경에서의 의사결정 알고리즘인 Case-based Decision Theory(CBDT)를 적용한 "RLs-CBDT"를 제안한다. 그리고 테트리스 실험을 통하여 기존의 RL 알고리즘에 비해 RLs-CBDT가 최적의 정책에 더 빠르게 수렴하는 것을 보인다.

키워드 : Potential-based 강화학습, Sarsa(0), 사례 기반 결정 이론, 테트리스 문제

Abstract This paper proposes a potential-based reinforcement learning, called "RLs-CBDT", which combines multiple RL agents and case-base decision theory designed for decision making in uncertain environment as an expert knowledge in RL. We empirically show that RLs-CBDT converges to an optimal policy faster than pre-existing RL algorithms through a Tetris experiment.

Key words : Potential-based RL, Sarsa(0), Case-based Decision Theory, Tetris problem

· 본 연구는 지식경제부 및 정보통신산업진흥원의 대학 IT연구센터 지원 사업의 연구결과로 수행되었음(NIPA-2009-(C1090-0903-0004))
· 이 논문은 2009 한국컴퓨터종합학술대회에서 'Case-based 결정 이론을 융합한 Potential-based 강화 학습'의 제목으로 발표된 논문을 확장한 것임

† 학생회원 : 서강대학교 컴퓨터공학과
wowesun@sogang.ac.kr
†† 정 회 원 : 서강대학교 컴퓨터공학과 교수
hschang@sogang.ac.kr

논문접수 : 2009년 8월 14일
심사완료 : 2009년 10월 8일

정보과학회논문지: 컴퓨팅의 실제 및 레터 제15권 제12호(2009.12)

1. 서 론

로봇과 같이 지능을 가진 에이전트(agent)는 Markov Decision Process(MDP)[1]로 기술된 환경에서 순차적으로 의사를 결정해야하는 문제를 자주 접한다[2]. 에이전트가 MDP의 일부 파라미터들을 알 수 없을 때, 매 학습 시간 스텝(time-step)마다 직접 행동을 선택함으로써 얻어지는 상태 정보와 강화 신호(reinforcement signal, reward)들을 관찰함으로써 최적의 정책을 학습해나가야 한다.

최적의 정책을 학습하는데 널리 사용되고 있는 학습 알고리즘으로 강화 학습(reinforcement learning, 이하 RL) 알고리즘[2,3]이 있다. RL 알고리즘은 어떤 일반적인 조건들을 만족할 경우 최적 정책에 수렴하지만, 그 수렴 속도가 느리기 때문에 실제 문제에 RL 알고리즘을 적용하는 데는 어려움이 있다. 이를 극복하기 위하여 최근에 "감독된 지식(supervised knowledge)"을 RL의 과정에 융합하여 수렴 속도를 향상시키려는 연구가 진행되었다.

특히 Chang[4]은 Ng et al.[5]의 "potential-based reinforcement function"을 이용하여 Sarsa(0) 학습 알고리즘[2]에 다수 학습과 Expert의 조언을 조합하는 새로운 학습 프레임워크를 제시하였다. Chang[4]은 이 학습 프레임워크를 통하여 학습의 수렴 속도를 향상시키고자 하였고, 이론적인 최적 정책으로의 수렴성을 확립하였다. 그러나 Chang[4]의 논문에서는 각 상태에서 Expert들의 조언을 행동들의 공간(action space)에 대한 확률 분포 형태로 기본 에이전트의 학습에 반영하는 이론적인 아이디어만 제시하고 있을 뿐, Expert의 조언을 구체적으로 적용하는 방법에 대해서는 제시하지 않고 있다. 따라서 본 논문에서는 "potential-based" RL 기법에 Case-based Decision Theory(CBDT)[6,7]를 적용한 "RLs-CBDT"를 제안한다.

CBDT는 불확실한 환경 하에서의 의사결정을 내리는 방법 이론으로 거의 모든 문제에 적용이 가능하다. "비슷한 문제는 비슷한 해결책을 갖는다."라는 가정으로부터, 에이전트는 현재 상황과 비슷한 과거의 경험에서 취했었던 행동의 성능에 기반을 두어 현재 상황에서 취할 행동을 선택한다. 이러한 특징을 갖기 때문에, CBDT는 Expert로써 기본 에이전트에 행동 공간에 대한 확률 분포의 형태로 조언을 주기에 가장 적합한 알고리즘이라 할 수 있다.

본 논문의 구성은 다음과 같다. 2절에서는 강화 학습과 CBDT의 기본적인 내용을, 3절에서는 다수의 강화 학습을 조합하는 기법에 CBDT를 융합하는 "RLs-CBDT" 기법을 설명한다. 4절에서는 RLs-CBDT를 테트리스 실험에 적용하기 위한 구체적인 모델을 정의하고, 5절에서는 RLs-CBDT의 성능을 확인한다. 마지막으로 6절에서는 본 논문의 결론을 내린다.

2. Related Works

2.1 MDP 모델과 Sarsa(0)

학습 에이전트는 MDP 모델로 기술된 어떤 환경과 상호작용을 한다. MDP M은 4개의 튜플-(X,A,P,R)로 구성된다. 여기서 X는 시스템 안에 존재하는 상태(state) x들의 유한 집합이고, A는 행동(action) a들의 유한집합이다(간결성을 위하여 모든 행동은 모든 상태에서 실행가능하다고 가정한다.). P는 $\{(x,a)|x\in X, a\in A\}$를 X에서 가능한 모든 확률 분포들의 집합으로의 매핑하는 상태 전이 함수이다. 상태 x에서 행동 a를 취하여 다른 상태 y로 전이할 수 있는 확률을 $P(y|x,a)$라 하자. R은 $X\times A\times X$를 실수집합 $\Re$로 매핑하는 보상함수라 하고, 상태 x에서 행동 a를 취하여 다른 상태 y로 전이하였을 때 얻는 보상을 $R(x,a,y)$라 하자.

$X\times A$에서의 Q^*-함수를 다음과 같이 정의한다:

$$Q^*(x,a)=\left\{\sum_{y\in X}P(y|x,a)(R(x,a,y)+\gamma\max_{a'\in A}Q^*(y,a')\right\}. \quad (1)$$

여기서 $\gamma\in(0,1)$는 discount factor이다. 에이전트가 학습한다는 것은 에이전트가 M으로 기술된 환경과 상호작용하는 동안 상태 전이 함수 P와 보상 함수 R을 알 수 없을 때 Q^*-함수를 학습하는 것이다.

Sarsa(0)는 오직 "다음"시간 단위에서 관찰되는 값만을 기반으로 하여 Q-값을 갱신한다. 비동기(asynchronous) Sarsa(0)에서는 각각 이산적인 시간스텝 $t\geq 0$에서 학습 에이전트가 상태 x_t를 관찰하고 학습 전략 ϕ_t에 따른 x_t에서의 행동 a_t를 취한다. 그리고 확률적으로 결정되는 다음 상태 x_{t+1}를 관찰하고 ϕ_t에 따라 x_{t+1}에서의 행동 a_{t+1}을 선택한다. 이러한 과정을 통해 얻어지는 값들을 이용하여 $X\times A$집합으로 정의된 $Q_t(x_t,a_t)$-부분을 다음의 식에 의해 갱신한다.

$$Q_{t+1}(x_t,a_t)\leftarrow Q_t(x_t,a_t)+\alpha_t(x_t,a_t)[R(x_t,a_t,x_{t+1}) +\gamma Q_t(x_{t+1},a_{t+1})-Q_t(x_t,a_t)] \quad (2)$$

$\alpha_t(x_t,a_t)$는 음의 값을 갖지 않는 스텝크기(stepsize)를 나타내는 계수이며, ϕ_t로 $\varepsilon_t-greedy$ 전략[8]을 사용할 것이다. 다시 말해 시간 t에서 $1-c/n(x_t)$($c\in(0,1)$)의 확률로 greedy한 행동 $a_t\in\arg max_{a\in A}Q_t(x_t,a)$가 선택되며, $c/(|A|n(x_t))$의 확률로 $a_t=a, a\in A$가 된다. $n_t(x_t)$는 시간 스텝 t까지의 상태 x_t로의 방문 회수를 의미한다.

2.2 Case-based Decision Theory (CBDT)

CBDT는 경험을 사례화하여 저장한다. 이러한 사례들의 집합은 $C=P_c\times A_c\times R_c$로 주어진다. P_c는 문제(problem)의 유한집합, A_c는 행동의 유한집합, R_c는 결과의 집합을 의미한다. 특히 $r\in R_c$은 $p\in P_c$에 $a\in A_c$를 적용하였을 때의 결과를 의미하고 $r=r(p,a)$로 쓴다. 효용 함수 $u: R_c\rightarrow\Re$($u: P_c\times A_c\rightarrow\Re$)는 결과에 대한 효용 값을 의미한다. 유사도함수 $\sigma: P_c\times P_c\rightarrow[0,1]$는 두 문제에 대한 유사도를 측정하는데 사용된다. 의사 결정자는 아래처럼 표기되는 사례를 저장하기 위한 제한된 크기의 메모리 M_c을 가지고 있다:

$$M_c=\{(p_1,a_1,r_1),(p_2,a_2,r_2),...,(p_n,a_n,r_n)\},$$
$$p_i\in P_c,\ a_i\in A_c,\ r_i=r(p_i,a_i)\in R_c,\ n=|P_c|,\ 1\leq i\leq n.$$

과거에 경험하였던 비슷한 문제가 메모리 M_c에 존재한다면 어떤 행동의 성능을 평가할 수 있다. 주어진 새로운 문제 $p_0\in P$에 대하여, CBDT는 아래의 식 (3)을 통하여 M_c에 저장된, p_0와 유사한, 과거 사례들에 대한 가장 좋은 결과를 이끌어주었던 행동 a를 선택한다.

$$V_{p_0}(a_0)=\sum_{(p',a_0,r')\in M_c}\left[\frac{\sigma(p_0,p')\cdot u(r')}{\sum_{(p'',a_0,r'')\in M_c}\sigma(p_0,p'')}\right],\ a_0\in A_c \quad (3)$$

3. 다수의 강화학습들과 CBDT의 융합 기법

본 절에서는 다수의 강화 학습들을 조합하는 기법[4]에 CBDT를 융합하는 방법에 대하여 자세히 설명한다.

MDP $M=(X,A,P,R)$이 주어져 있다고 할 때 학습 에이전트의 목표는 Q_M^*-함수를 학습하는 것이다. 다음으로 potential 함수 $\Phi: X\rightarrow\Re$에 의해서 R이

$$R'(x,a,y)=R(x,a,y)+\gamma\Phi(y)-\Phi(x) \quad (4)$$

와 같이 변환된 $M'=(X,A,P,R')$를 고려해 보자. Ng et al.[5]은 모든 $x\in X$, $a\in A$에 대해 어떤 potential 함수 $\Phi: X\rightarrow R$에 대해서, $Q_{M'}^*$-함수를 학습하는 것은 Q_M^*-함수를 학습하는 것과 같다(equivalent)는 것을 보였다. 기본 에이전트와 서브에이전트 $i=1,2,...,m$가 있다고 가정하자. 다수 학습은 M'에서 Sarsa(0)를 사용하는 기본 에이전트와, Q_M^*를 학습하는 M에서 각각 다른 convergent 강화 학습 방법을 사용하는 여러 개의 서브에이전트들이 동시에 학습함으로써 이루어진다. 기본 에이전트는 위의 potential 함수를 사용하여 서브에이전트들의 학습 결과가 자신의 학습에 반영될 수 있도록 기본 에이전트의 강화 신호를 조성(shaping)한다. 모든 서브에이전트들과 기본 에이전트는 비동기적인 자신들만의 학습 갱신 시간스텝(learning update time-step)을 갖는다. 기본에이전트의 현재 갱신 시간이 t일 때 서브에이전트 i의 갱신 시간을 t_i라고 하자. 그리고 $Q_{t_i}^i$를 서브에이전트 i의 t_i에서의 Q_M^*-함수에 대한 추정 값이라고 하자. 그리고 CBDT 알고리즘이 로컬 시간 k에서 제공해 주는, 상태 $x\in X$에서 행동 $a\in A$가 optimal한 행동일 확률을 $\rho_k(x,a)$라 하자. 기본에이전트는 Sarsa(0)의

갱신 룰 (2)을 따르되 시간 t에서의 ϕ_t에 대해 $\varepsilon_t-greedy$ 전략을 사용하며, MDP M'에 대한 학습을 하게 된다. 그 갱신 룰은 다음과 같다:

$$Q_{t+1}(x_t,a_t)\leftarrow Q_t(x_t,a_t)+\alpha_t(x_t,a_t)[R(x_t,a_t,x_{t+1}) +\gamma\Phi_t(x_{t+1};t_1,\ldots,t_m,k)-\Phi_t(x_t;t_1,\ldots,t_m,k) +\gamma Q_t(x_{t+1},a_{t+1})-Q_t(x_t,a_t)]. \quad (5)$$

함수 $\Phi(x_t)(=\Phi(x_t;t_1,..,t_m,k))$는 다음과 같이 정의된다.

$$\Phi(x_t)=\sum_{a\in A}\left[\frac{1}{m}\sum_{i=1}^{m}Q_{t_i}^i(x_t,a)\cdot\theta(x_t,a;t_1,..,t_m,k)\right] \quad (6)$$

$$\theta(x_t,a;t_1,..,t_m,k)\leftarrow \frac{\sum_{i=1}^{m}I(a\in \arg max_{b\in A}Q_{t_i}^i(x_t,b))}{\sum_{a'\in A}\sum_{i=1}^{m}I(a'\in \arg max_{b\in A}Q_{t_i}^i(x_t,b))}\times\frac{\rho_k(x_t,a)}{\sum_{a'\in A}\rho_k(x_t,a')} \quad (7)$$

$$\rho_k(x,a)=\frac{V_x(a)}{\sum_{a'\in A}V_x(a')} \quad (8)$$

여기서 $V_x(a)$는 2.2에서 정의한 CBDT 의사결정 식 (3)과 동일하다. 만일 에이전트가 상태 x에 대해 어떠한 조언도 가지고 있지 않다면, 행동 집합 A에 대해 동일한 확률 분포를 취할 것이다. 또한 시간 t에서 $x_t\in X$, $a_t\in A$에 대하여 $\alpha_t(x,a)$는 $(x,a)=(x_t,a_t)$인 경우에 $1/n_t(x,a)$ 값을 갖고, 나머지 $(x,a)\neq(x_t,a_t)$인 경우에는 0의 값을 갖는다. 함수 $I(a\in \arg max_{b\in A}Q_{t_i}^i(x_t,b))$ 는 $a\in A$일 때 $a\in \arg\max_{b\in A}Q_{t_i}^i(x_t,b)$이 참일 경우 1, 그 이외에는 0을 갖는 것으로 주어져 있다. $n_t(x,a)$는 시간 스텝 t까지의 상태 x에서 행동 a를 취한 횟수를 의미한다. Potential 함수 Φ를 위와 같이 정한 이론적 근거는 논문[4]에 자세히 언급되어 있다.

ρ_k가 $k\rightarrow\infty$일 때, 시간에 대해 stationary한 확률 분포로 수렴하면 Φ 역시 stationary한 함수로 수렴하게 되며, 이는 $Q_{M'}^*$-함수를 학습하는 Sarsa(0)의 수렴을 보장한다. 아래의 그림 1은 기본에이전트가 수행하는 RLs-CBDT의 알고리즘이다.

1. Initialize $Q_0(x,a)$ arbitrarily for all $x\in X, a\in A$
2. Initialize γ, $t\leftarrow 0$
3. Calculate the a probability distribution over A $\rho_k(x,a)$ of (8) for all $x\in X$, $a\in A$ by using CBDT's memory M_t and decision making equation (3)
4. REPEAT (for each episode):
   Initialize x_t
   Select a_t from x_t by using $\varepsilon_t-greedy$ strategy
   REPEAT (for each step of episode):
      Execute action a_t
      Observe $R(x_t,a_t,x_{t+1})$, x_{t+1}
      Select a_{t+1} from x_{t+1} by using $\varepsilon_t-greedy$ strategy
      Calculate potential function (6)
      Update $Q_{t+1}(x_t,a_t)$ by Sarsa(0) update rule (5)
      $x_t\leftarrow x_{t+1}$; $a_t\leftarrow a_{t+1}$; $t\leftarrow t+1$;
   UNTIL x_t is terminal

그림 1 RLs-CBDT 알고리즘

4. 융합 알고리즘의 실험환경

4.1 Tetris problem

본 연구에서는 강화학습 알고리즘 Sarsa(0), CBDT 알고리즘과 RLs-CBDT의 성능을 비교하기 위하여 기존의 테트리스를 간단하게 재구성한 Melax가 제안한 테트리스[9]를 이용하였다. Melax가 제안한 테트리스는 그림 2와 같이 블록의 종류가 기존의 태트리스의 블록보다 작고 모양이 단순하다. 각각의 블록은 균일한 (uniform) 분포로 선택되어 필드에 떨어진다. 블록의 집합 $B=\{b^1,b^2,b^3,b^4,b^5\}$는 그림 2와 같다.

그림 2 블록의 집합

실험을 위하여 너비(width)는 6, 높이(height)는 제한되지 않은 필드(field)를 사용하였다. 그리고 1 game은 정해진 K개의 조각이 모두 필드에 떨어지면 종료된다. game의 결과는 한 game을 진행한 후 "*삭제된 총 줄 수*"를 측정한다. 1 round에서는 총 G번의 game이 진행된다. 이와 같이 설정한 이유는 1 game안에 많은 상태를 방문할 수 있게 하기 위해서이다. 학습 알고리즘은 여러 번의 game을 진행하면서 최적의 정책을 학습해 나간다.

4.2 테트리스에 적용할 MDP 모델과 CBDT 모델

기본 에이전트로 Sarsa(0) 알고리즘을 사용하는 "RLs-CBDT" 융합 알고리즘은 테트리스 실험을 위하여 서브 에이전트로 Q-Learning[2]을 사용한다. 기본 에이전트와 서브에이전트는 MDP $M_{tetris}=(X,A,P,R)$ 도델로 기술된 실험 환경과 상호작용을 한다.

상태 $x\in X$는 필드에 블록이 쌓여 있는 형태(configuration)와 필드에 떨어뜨릴 임의로 선택된 블록으로 표현된다. 필드에 블록이 쌓여있는 형태는 Top To Level Representation(TTL)[9]을 적용하여 구한다. TTL은 아래의 그림 3과 같이 블록이 가장 높이 쌓여있는 그 행에서 아래의 행의 필드의 형태를 이진수(0과 1)로 표현한 것이다. 블록이 가장 높이 쌓여있는 column의 높이를 m이라 하자. 그러면 이 TTL 표현은 m과 $m-1$의 행의 정보를 나타낸다. TTL 표현이 가지는 정보를 나타내는 기호, $f_{m,i},f_{m-1,i}\in\{0,1\}$ $(1\le i\le 6)$를 아래의 그림 4와 같이 정의한다.

그림 3 TTL

$f_{m,1}$	$f_{m,2}$	$f_{m,3}$	$f_{m,4}$	$f_{m,5}$	$f_{m,6}$
$f_{m-1,1}$	$f_{m-1,2}$	$f_{m-1,3}$	$f_{m-1,4}$	$f_{m-1,5}$	$f_{m-1,6}$

그림 4 TTL의 수식 표현

그러면 필드의 형태 F는 이진수로 표현될 수 있다. 또한, 임의로 선택된 블록도 상태에 포함되어야 하기 때문에 위에서 언급한 F와 현재의 블록 $b \in B$ 또는 $b=0$으로 상태 x는 $x=\langle F,b\rangle$, $b \in B \cup \{0\}$와 같이 표현된다. 그러면 상태 공간의 크기는 $|X|=2^{12}\times 6$이 된다. $b=0$인 상태 $x=\langle F,0\rangle$는 떨어뜨릴 블록이 없을 때를 의미한다. $b=0$인 4096개의 상태 $x=\langle F,0\rangle$들을 final state x_q라고 하고, **final state들의 집합**을 X_q라고 하자. 그리고 $x_e \in \{x | x=\langle 000000000000_{(2)}, b\rangle, b \in B\}$와 같이 정의된 상태를 5개의 empty 필드 state x_e라고 하고 **empty 필드 state들의 집합**을 X_e라고 하자. x_e는 아무런 블록이 쌓이지 않은 필드의 상태를 갖는다.

테트리스의 행동 $a \in A$는 주어진 블록을 필드에 쌓는 방법으로 표현된다. 이때 블록을 필드의 어느 위치에서 얼마만큼 회전시켜서 수직으로 떨어뜨리는 지만 결정하면 된다. 그러므로 하나의 행동 a는 $a=<$*블록의 column의 위치, 회전 횟수*$>$와 같이 블록의 column위치와 회전 횟수를 속성으로 가지는 벡터로 표현된다. 이때 블록이 떨어지게 될 column의 위치 값은, [1,6]사이의 값을 가지고, 블록의 회전 횟수는 [0,3]사이의 값을 가진다. 그리고 1개의 special 행동으로 $P(x_e|x_q,a_q)=1(x_q \in X_q, x_e \in X_e)$를 만족하는 **final 행동** a_q를 정의한다.

상태 전이 함수 P는 다음과 같다. $y \in X_e$, $x \in X_q$일 때 $P(y|x,a_q)=1$이고, 다른 경우에는 $P(y|x,a_q)=0$이다. 그리고 모든 $x,y \in X-(X_q \cup X_e)$, $a \in A-\{a_q\}$에 대해서 $P(y|x,a)$는 알 수 없다.

보상 함수 R은 떨어뜨릴 블록을 행동 a_t에 따라 필드에 쌓은 후 삭제되는 줄 수를 이용하여 다음과 같이 정의된다.

$$R(x_t,a_t,x_{t+1})=(\text{삭제된 줄 수})^2 \tag{9}$$

CBDT의 사례 $C_{tetris}=P_c\times A_c\times R_c$에서 문제들의 집합 P_c와 행동들의 집합 A_c는 MDP M_{tetris}에서 각각 상태들의 집합 X와 A와 같다. 유사도 함수 σ는 두 상태 $x=<F,b>$, $x'=<F',b'>$, $x,x' \in X$가 있다고 하면, $b=b'$인 경우에 대하여 위에서 언급한 TTL로 표현된 두 F,F'에서 각 자리의 값(0 또는 1)을 비교한다. 상태를 이진수로 표현하였기 때문에 XOR의 연산을 이용하면 두 상태의 유사도를 간단히 계산할 수 있다.

$$\sigma(F,F')=\frac{W^*2-ND(F,F')}{W^*2} \tag{10}$$

여기서 $ND(F,F')=\sum_{i=1}^{W^*2}((F)XOR(F'))_i$이고, 는 F와 F'를 XOR하여 나온 이진수의 값에서 i번째 위치에 있는 숫자(0 또는 1)의 값을 $((F)XOR(F'))_i$로 나타낸다. $b \neq b'$인 경우에는 $\sigma(F,F')=0$의 값을 갖는다. 테트리스에서는 효용값을 수행 즉시 알아낼 수 있으므로 사례화할 때 결과 대신 수행 후 알 수 있는 효용값을 직접 저장하여 사례화하였다.

$$u(r(x,a))=(\text{삭제된 줄 수})^2*100 \tag{11}$$

위와 같이 정의한 모델을 통하여 테트리스 실험 상에서 기존의 Sarsa(0) 알고리즘, CBDT, 휴리스틱 알고리즘과 RLs-CBDT 융합 알고리즘과의 학습 수렴 속도와 성능 차이를 분석하였고, 5절에서 설명된다.

5. RLs-CBDT의 실험 결과 및 분석

RLs-CBDT 알고리즘과 경쟁자 알고리즘들 간의 성능 차를 분석하기에 앞서 최고의 성능 간의 비교를 위하여, 주어진 $\gamma=0.9$에 대해서, $\varepsilon_t-greedy$ 전략에서 사용하는 파라미터 상수 $c \in (0,1)$의 최적 값을 찾는 실험을 하였다.

테트리스 실험을 위하여 4.1에서 정의한 파라미터 $K=500$, $G=100$으로 하고, 총 2000번의 round를 진행한다. 그림 5, 6, 7, 8의 그래프에서 x축은 round를 나타내고, y축은 average deleted lines를 나타낸다. 이는 1 round동안 "삭제된 줄 수"의 평균값을 의미한다.

그림 5, 6은 c로 주어질 수 있는 여러 값들을 적용하여 실험한 결과의 그래프이다. 이 그래프를 통하여 기존 Sarsa(0)와 RLs-CBDT 모두 $c=0.9$일 때 가장 좋은 성능을 보여줌을 알 수 있다. c가 1에 가까울수록 $\varepsilon_t-greedy$ 전략에서 방문한 상태의 수가 적은 경우에는 exploration이 빈번히 일어나고, 같은 상태에 여러 번 방문한 경우에는 exploitation이 빈번히 일어나기 때문에 가장 좋은 성능을 낸다.

그림 7, 8은 기존의 Sarsa(0), CBDT, 휴리스틱 알고리즘과 RLs-CBDT과의 성능을 비교한 그래프이다. 실험에 사용한 휴리스틱 알고리즘은 현재 주어진 상태 x_t에서 가상으로 모든 블록과 모든 행동에 대해 2 시간스텝을 진행해 본 후 최대의 결과를 주는 행동을 시간 t에서 a_t로 정한다. RLs-CBDT와 Sarsa(0) 알고리즘은 가장 좋은 성능을 내는 $c=0.9$로 설정하고 실험을 하였다. 그림 7을 통하여, 연속적인 의사결정을 해야 하는 테트리스 문제에서, RLs-CBDT가 CBDT보다 약 3배, 휴리스틱보다 약 2.6배 좋은 성능을 보이고 있음을 알 수 있다. 그림 8은 그림 7의 부분 그래프로 1 round부터 100 round 사이의 그래프를 확대한 것이다. 휴리스틱과 CBDT가 초반에는 앞서다가 약 10 round 이후에서부터 RLs-CBDT의 성능이 증가하고 있고, 초반에는 Sarsa(0)와 RLs-CBDT의 성능이 비슷하다가 약 30 round에서부터 RLs-CBDT의 성능이 증가되고 있음을 볼 수 있다. 또한 Sarsa(0)는 약 800 round에서부터, RLs-CBDT는 약 400 round

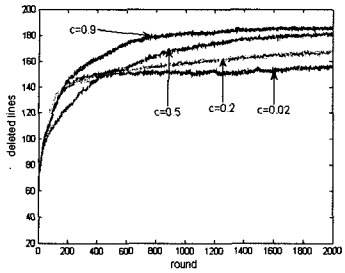

그림 5 Sarsa(0)의 c값에 따른 성능 비교

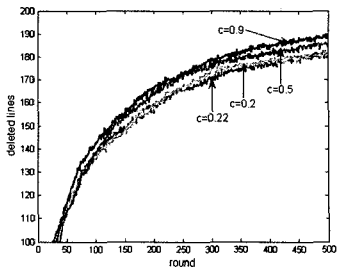

그림 6 RLs-CBDT의 c값에 따른 성능 비교

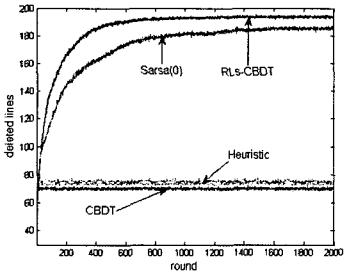

그림 7 RLs-CBDT와 경쟁자 알고리즘들간의 성능 비교

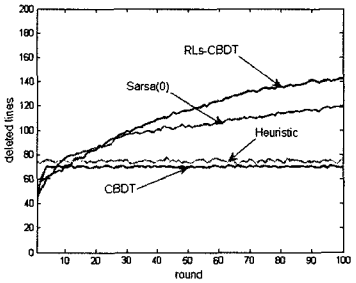

그림 8 그림 7의 부분 그래프

에서부터 최적 정책에 수렴해 가는 것을 확인할 수 있고, 이를 통해 RLs-CBDT의 학습 수렴 속도가 Sarsa(0)보다 약 2배정도 빠름을 알 수 있다.

이 실험을 통하여, 순차적으로 의사를 결정해야하는 문제에서 CBDT의 지식과 서브에이전트들의 학습결과가 융합된 RLs-CBDT의 학습 수렴 속도가 기존 Sarsa(0)의 학습 수렴 속도보다 향상되었음을 확인할 수 있다.

6. 결 론

지금까지 다수의 강화학습을 조합한 기법에 CBDT를 융합한 RLs-CBDT에 대해서 알아보았다. 이는 여러 RL을 수행하는 서브에이전트들의 독립적인 학습과 CBDT의 지식이 감독(supervision)된 형태로 강화 학습과 조합된 융합 알고리즘이다. 이 융합 알고리즘은 서브에이전트들이 어떤 종류의 학습을 하더라도, CBDT가 어떤 시스템을 기반으로 하더라도 별다른 제한 없이 조합이 가능하다.

RLs-CBDT 알고리즘을 Melax 테트리스에 적용하는 실험을 통하여 기존의 RL 학습 알고리즘에 비하여 하나 이상의 강화학습과 CBDT의 지식이 융합된 RLs-CBDT 알고리즘이 최적의 정책에 더 빠르게 수렴하는 것을 확인할 수 있었다.

참 고 문 헌

[1] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, wiley, New York, 1994.

[2] R. Sutton and A. Barrto, *Reinforcement Learning*, MIT Press, 2000.

[3] L. P. Kaelbling, Michael L. Littman, Andrew W. Moore, "Reinforcement learning: A survey," *Journal of Articial Intelligence Research*, vol.4, pp.237-285, 1996.

[4] H. S. Chang, "Reinforcement Learning with Supervision by Combining Multiple Learnings and Expert Advices," in *Proc. of the 2006 American Control Conference*, pp.4159-4164, June, 2006.

[5] A. Y. Ng, D. Harada, S. Russel, "Policy invariance under reward transformations:theory and application to reward shaping," in *Proc. of the 16th Int. Conf. on Machine Learning*, pp.278-287, 1999.

[6] I. Gilboa and D. Schmeidler, "Case-based decision theory," *Quart. J. Economics*, vol.110, no.4, pp.605-639, 1995.

[7] E. Hllermeier "Experience-based decision making: a satisficing decision tree approach," *IEEE Transactions on Systems, Man, and Cybernetics*, vol.35, no.5, pp.641-653, 2005.

[8] S. Singh, T. jaakkola, M. Littman, and C. Szepesvari, "Convergence results for single-step on-policy reinforcement learning algorithms," *Machine Learning*, vol.38, pp.287-308, 2000.

[9] S. Melax "Reinforcement learning tetris example," 1998. URL http://www.melax.com/tetris/.