Consiglio Nazionale delle Ricerche
Istituto di Calcolo e Reti ad Alte Prestazioni

Cellular Transfer Learning for lighting control system based on Edge Infrastructure

Emilio Greco

RT- ICAR-CS-20-05 **Settembre 2020**

Consiglio Nazionale delle Ricerche, Istituto di Calcolo e Reti ad Alte Prestazioni (ICAR)
– Sede di Cosenza, Via P. Bucci 8-9C, 87036 Rende, Italy, URL: *www.icar.cnr.it*
– Sezione di Napoli, Via P. Castellino 111, 80131 Napoli, URL: *www.icar.cnr.it*
– Sezione di Palermo, Via Ugo La Malfa, 153, 90146 Palermo, URL: *www.icar.cnr.it*

Indice generale

Premessa

La conoscenza collettiva umana ha chiaramente beneficiato del fatto che le innovazioni sviluppate da alcuni individui sono poi state trasferite ad altri attraverso la comunicazione. Allo stesso modo dei gruppi sociali umani, gli agenti in un sistema di apprendimento distribuito trarrebbero probabilmente vantaggio dalla comunicazione per condividere conoscenze ed apprendere abilità. Il problema di trasferimento della conoscenza per migliorare l'apprendimento tra agenti è un argomento molto dibattuto, in questo articolo affronteremo la problematica nel contesto dell'Internet of Things (IoT) platform for mission critical applications, ovvero applicazioni con rigidi requisiti in termini di tempo di risposta e mobilità. Il problema del trasferimento della conoscenza ha delle complessità intrinseche relative alla misurazione sugli impatti a lungo termine che possono deviare il processo di apprendimento. In questo lavoro presentiamo un algoritmo scalabile e decentralizzato per gli agenti intelligenti che imparano in ambienti indipendenti e dai consigli provenienti dal vicinato. Il vicinato, inteso come prossimità di ambienti fisici in scenari applicativi reali. In tal senso i vicini vengono ritenuti più esperti del dominio applicativo rispetto ad altri perché capaci di cogliere informazioni locali sull'ambiente altamente dinamico. Secondo il nostro approccio, gli agenti allo stesso tempo assolvono alla funzione di studenti ed insegnanti in un ambiente multiagente. Ogni agente impara da un gruppo di conoscenti o vicini valutando i consigli utili al proprio obiettivo riducendo così al minimo la comunicazione. I consigli ricevuti vengono utilizzati sia per migliorare l'apprendimento locale che per migliorare le proprie performance da insegnante. La metodologia proposta si basa sul paradigma computazionale degli Automi Cellulari applicato a sistemi eterogenei usato sia come approccio al calcolo parallelo e sia come tecnica per condividere e diffondere la conoscenza. Conoscenza che può realizzarsi indipendentemente e più velocemente in alcune celle, per poi propagarsi all'interno delle altre celle che compongono l'intero sistema.

Introduzione

Il bagaglio culturale di una società si basa sul trasferimento delle conquiste e delle conoscenze compiute da alcuni individui verso gli altri attraverso canali di comunicazione [1], questo non solo serve a migliore le prestazioni finali dell'oggetto dell'innovazione, ma anche a far progredire la conoscenza del gruppo sociale. Esistono diversi lavori in cui gli agenti interagiscono, apprendono ed adattano i loro comportamenti durante l'interazione con un ambiente che può essere indipendente o condiviso (ad esempio, auto autonome o assistite, robot, semafori intelligenti). Sebbene un determinato agente potrebbe non essere un esperto, durante l'apprendimento, può acquisire conoscenze locali che potrebbero essere successivamente utili ai compagni di squadra ancora inconsapevoli. Simile ai gruppi sociali umani, gli agenti in un sistema di apprendimento distribuito

trarrebbero probabilmente vantaggio dalla comunicazione per condividere conoscenze ed apprendere abilità, migliorando così l'efficacia dell'apprendimento a livello di sistema.

Il vantaggio dell'insegnamento peer-to-peer appreso è che può accelerare l'apprendimento anche senza fare affidamento sull'esistenza di Insegnanti “onniscienti”. Lavori recenti sull’apprendimento cooperativo multiagente per rinforzo si sono mostrati di grande aiuto per capire come le prestazioni delle attività possono essere migliorate introducendo meccanismi di comunicazione inter-agent [2]. In questo elaborato facciamo riferimento ad un nuovo approccio in cui gli agenti imparano ed insegnano ciò che è stato acquisito comunicando consigli sull'azione nell’intorno del proprio vicinato, sfruttando a pieno le conoscenze locali, migliorando così le prestazioni finali, accelerando il processo di apprendimento e minimizzando le comunicazioni di rete.

In [3], vengono descritte le complessità intrinseche che si riscontrano in sistemi di apprendimento multiagent, lasciando aperte diverse questioni chiave che devono essere affrontate. In primo luogo, gli agenti devono imparare quando insegnare, cosa insegnare e come imparare da ciò che viene loro trasmesso. Secondo, nonostante ci sia l’esigenza di coordinarsi in un ambiente condiviso, gli agenti possono essere discenti, avere un apprendimento indipendente o avere vincoli di privacy (come ad es. robot di società distinte che non possono condividere le politiche complete) e quindi devono apprendere ed allo stesso tempo insegnare sotto questi vincoli.

Un terzo problema è che gli agenti devono stimare l'impatto di ciascun consiglio sui progressi di apprendimento del loro compagno di squadra [4]. Una delle principali ragioni per la mancanza di progressi nell'affrontare le sfide inerenti questi aspetti è il significativo aumento della complessità di calcolo nonché problemi di congestione di rete con l’aumento della comunicazione. Il nostro algoritmo supporta l'insegnamento tra compagni di squadra eterogenei e si applica a contesti dell’Internet of Things (IoT) platform for mission critical applications, ovvero applicazioni con rigidi requisiti in termini di tempo di risposta e mobilità. In tali contesti è necessario quindi adottare un modello HPC (High Performance Computing) scalabile che supporti l'implementazione di applicazioni sui dispositivi decentralizzati ma con risorse limitate dell'IoT. Recentemente, all'interno delle applicazioni sono stati introdotti algoritmi decisionali che utilizzano algoritmi di Reinforcement Learning (RL) che interagiscono direttamente con l’ambiente. Tuttavia, la maggior parte degli algoritmi RL sono progettati per ambienti centralizzati e consumano tempo e risorse. Esempi di tali applicazioni sono sistemi di controllo intelligenti per veicoli autonomi, robot per la pianificazione di percorsi, gestione della rete e semafori [5,6]. In questo ultimo periodo, stiamo assistendo all'aumento della tendenza verso un'infrastruttura altamente decentralizzata in cui i nodi di elaborazione vengono distribuiti all'estremità della rete [7,8]. In generale, i sistemi di controllo intelligenti funzionano

risolvendo il problema di ottimizzazione correlato che è alla base della maggior parte degli approcci di Machine Learning (ML). Uno di questi approcci è l'apprendimento per rinforzo (RL) [9]. Q-Learning (QL) è uno degli algoritmi di RL più efficienti e più semplici che non ne richiedono conoscenza precedente del problema o dell'ambiente dinamico. Un aspetto importante di QL è che converge ad una politica ottimale se viene utilizzata per l'apprendimento una rappresentazione tabellare della funzione di utilità [10]. Tuttavia QL comporta due principali problemi che ne limitano l'applicabilità in quegli ambienti. Innanzitutto, richiede un lungo tempo di addestramento per apprendere la politica ottimale e in secondo luogo, poiché la convergenza è garantita solo se viene utilizzata una rappresentazione tabellare, la dimensione della tabella può crescere rapidamente con conseguenti problemi di scalabilità.

I limiti dell'applicazione di QL su problemi su larga scala sono stati affrontati principalmente utilizzando approssimatori di funzioni [13], come le reti neurali (NN) [14]. La metodologia consente di risolvere problemi con ampi spazi degli stati e l'individuazione di politiche di selezione delle azioni più rapidamente rispetto agli approcci basati su tabelle. Tuttavia, la funzione di approssimazione non ha garanzie di convergenza teorica [15] e risulta più lenta nel trovare politiche accurate (o ottimali) rispetto agli approcci basati su tabelle [16]. In questo articolo proponiamo un approccio di apprendimento multi-agente che supporta problemi su larga scala senza perdere la garanzia di convergenza. I contributi del lavoro sono duplici. Innanzitutto, forniamo una metodologia di apprendimento multi-agente che può essere utilizzata sia per sistemi di apprendimento Collaborative Multi Agent Q-Learning [11] che per sistemi Parallel Q-Learning (PQL). In secondo luogo, presentiamo un algoritmo di QL basato su tabelle per ambienti decentralizzati che riduce al minimo la comunicazione Inter-Agent, la fase si addestramento e mantiene nel contempo garanzia di convergenza e scalabilità in termini di memoria ed elaborazione. L'efficacia dell'approccio è stata dimostrata applicando l’algoritmo alla risoluzione del problema del labirinto. Si è realizzata una struttura computazionale ad CA con 100 celle costituite da altrettanti labirinti con differenti configurazioni.

Il documento è strutturato come segue: Dalla sezione II alla sezione VII vengono introdotti alcuni concetti di base utili per comprendere il resto del documento e alcuni lavori correlati; Sezione VIII vengono descritti i parametri e gli obiettivi considerati del problema che si intende modellare e risolvere. La sezione IX mostra la metodologia applicata al problema del labirinto e fornisce alcuni risultati sperimentali. Infine, vengono tratte alcune conclusioni.

QL problem

Nel classico QL sequenziale, un agente acquisisce lo stato corrente s ∈ S rilevandolo dall'ambiente, quindi seleziona un'azione a ∈ A. Come risultato dell'azione selezionata, l'agente si sposta in un nuovo stato s 0 ∈ S e riceve una ricompensa r ∈ R. L'obiettivo dell'agente è di massimizzare la ricompensa totale cumulata nel tempo. Le ricompense ottenute vengono utilizzate per stimare la funzione valore-azione che rappresenta l'utilità attesa nell'intraprendere una determinata azione in un determinato stato e seguendo una certa politica ottimale. La politica ottimale è costruita selezionando l'azione con il valore atteso più alto per ogni stato. L'esecuzione dell'algoritmo si basa su episodi che iniziano in uno stato specificato e termina quando l'agente raggiunge uno stato obiettivo o una determinata condizione viene soddisfatta. QL memorizza la stima prevista delle ricompense future, che probabilmente l'agente otterrà partendo dallo stato s e seleziona l'azione a, in una tabella di valori Q (s, a). L'algoritmo 1 mostra lo pseudo-codice dell'algoritmo QL.

Algorithm 1 Q-Learning

Require: Initialize Q-Table with $|S||A|$ values
repeat(for each episode)
  Initialize s
  repeat(for each iteration)
    Use behavior policy to choose a
    Take action a, observe r, s'
    $TDerror = \max_{a'} Q(s', a') - Q(s, a)$
    $Q(s, a) \leftarrow Q(s, a) + \alpha \cdot (r + \gamma \cdot TDerror)$
    $s \leftarrow s'$
  until s is goal state
until max episodes or convergence is True

Dove $0 \leq \gamma \leq 1$ è il fattore di sconto, un parametro che serve a bilanciare l'importanza delle ricompense attuali rispetto a quelle future, $0 < \alpha \leq 1$ è il tasso di apprendimento, un parametro che determina come le nuove informazioni sovrascriveranno le vecchie informazioni e TDerror è l'errore di differenza temporale tra il valore Q effettivo e la stima del valore ottimale futuro.

MULTI-AGENT model for QL

QL è uno degli algoritmi di RL più efficienti e più semplici che non necessita di alcuna conoscenza del problema o sulla dinamica dell'ambiente. Tuttavia, richiede molto tempo di addestramento per apprendere la politica ottimale e la sua convergenza è garantita solo se viene usata una rappresentazione tabellare, con conseguenti possibili problemi di scalabilità della memoria.

Collaborative Multi-Agent Q-Learning (C-MAQL) e PQL sono due framework che mirano a ridurre l'esperienza necessaria di ogni agente QL, aumentando il numero di agenti che risolvono il problema e sviluppando strategie per condividere e diffondere la conoscenza tra di loro. In C-MAQL, più istanze di QL (cioè agenti intelligenti) risolvono parti diverse del problema o lavorano in un ambiente che è alterato dalle azioni di altri agenti. In PQL, gli agenti imparano lo stesso compito in isolamento e in modo indipendente senza interagire tra di loro attraverso l'ambiente. Relativamente a PQL, è stato già dimostrato che il numero di episodi di apprendimento di una politica viene ridotto proporzionalmente di 1 / n [17], dove n è il numero di agenti che aggiornando la tabella Q condivisa.

Mentre la performance, in termini di numero di episodi, dipende dalle strategie e dalla frequenza di condivisione dell'esperienza tra gli agenti. Le implementazioni centralizzate sono solo parzialmente scalabili in quanto la tabella Q condivisa diventa un collo di bottiglia allorquando al crescere del numero di agenti aumenta anche la concorrenza e di conseguenza aumentano le risorse di elaborazione utilizzate per eseguire sia l'algoritmo che la gestione della concorrenza.

Il problema della scalabilità della memoria e dell'elaborazione di PQL in esecuzione su ambienti centralizzati potrebbe essere risolto distribuendo sia la tabella Q che gli agenti tra più nodi di elaborazione. Tuttavia, la distribuzione di agenti e tabella comporta diverse sfide in termini di sincronizzazione tra agenti, partizionamento delle tabelle, controllo della concorrenza tra tabelle e garanzia di convergenza basata su una tabella distribuita. Inoltre, distribuendo il calcolo su diverse macchine collegate tramite una rete tradizionale si aggiunge una latenza sulle operazioni della tabella di almeno un centinaio di microsecondi, che vengono tradotti in iterazioni in 2-3 ordini di grandezza, quindi la versione distribuita risulterebbe più lenta della versione centralizzata che esegue l'archiviazione in memoria.

Communicating to Speed Learning

Alcune ricerche, in particolare sull'apprendimento per rinforzo, fanno riferimento ad agenti che hanno accesso a una tabella di utilità comune o a una tabella di politiche congiunte a cui ciascuno può contribuire, anche se gli agenti sono studenti che lavorano su ambienti separati. Questo tipo di comunicazione è chiamata comunicazione hard-coded implicita: gli studenti si istruiscono a vicenda sulla base delle informazioni apprese. Ad esempio, in [20] i ricercatori sostengono che più studenti che condividono una tabella Q congiunta possono apprendere molto più velocemente degli studenti indipendenti. L'insegnamento quindi può anche essere fatto implicitamente o può essere combinato con la segnalazione esplicita.

In [21] si analizza il problema di inseguimento predatore-preda e mostra come la collaborazione tra gli agenti di apprendimento, che condividono la conoscenza del compito, superano in modo significativo gli studenti indipendenti. La condivisione della conoscenza avviene in tre modi: condivisione di informazioni immediate dei sensori (stato), condivisione di episodi (sequenze di “state, action, reward” sperimentata da un agente) o condivisione di informazioni sulla politica (sotto forma di stato, azione, utilità). La maggior parte dei benefici dell'apprendimento condiviso si ottengono in questi ultimi due approcci a spese però di un maggiore utilizzo delle comunicazioni, problema che in qualche modo viene risolto usando una comunicazione implicita come nel primo metodo.

Collaborative Multi-Agent Q-Learning

Le tecniche di apprendimento cooperativo [25] sono una classe di tecniche progettate per affrontare la cooperazione tra agenti in ambienti multi-agente con lo scopo di migliorare la qualità, l'accuratezza e la velocità del processo di apprendimento. La condivisione della conoscenza [26] è una delle tecniche di apprendimento cooperativo in cui gli agenti cercano di imparare gli uni dagli altri e di condividere ciò che apprendono. Usando questa tecnica si consente agli agenti di utilizzare la conoscenza dei vicini, raccolta attraverso i loro processi di apprendimento individuali.

L'apprendimento (C-MARL) è alla base di un'ampia gamma di applicazioni come il controllo dei semafori [27], guida autonoma [28] e controllo di una stazione radio base cellulare [28]. L'obiettivo del Reinforcement Learning (RL) è ottimizzare le azioni intraprese dagli agenti in ambienti complessi, da azioni di apprendimento che massimizzano efficacemente una ricompensa a lungo termine. Tecniche come C-MARL sono state introdotte per adattare gli algoritmi RL agli ambienti in cui un team di più agenti di RL interagiscono per perseguire un obbiettivo di squadra. Gli agenti che formano un team collaborano a un obiettivo comune. C-MARL presenta una serie di sfide da affrontare come l'ambiente non stazionario, la larghezza di banda limitata per la condivisione delle informazioni interne a un team [30] e la parziale osservabilità (un determinato agente potrebbe non avere sempre accesso a tutte le osservazioni fatte da altri agenti del team).

Gli attuali metodi MARL all'avanguardia applicano un programma di formazione centralizzato, dove gli agenti possono scambiare le loro informazioni locali durante la fase di formazione e agire in modo indipendente durante l'esecuzione. Un approccio per la formazione centralizzata e decentralizzata viene descritta in [31].

Cellular Automaton

Un automa cellulare descrive uno stato continuo rappresentato da un insieme di celle (elementi di una griglia) che viene generato a partire da un seme che può essere una singola cella o più celle. Un insieme di celle all'istante "t + 1" è un aggiornamento dell'insieme a "t" secondo una "regola" di aggiornamento. La regola di aggiornamento prevede calcoli che dipendono dalle celle adiacenti. Esistono tre definizioni standard di una cella adiacente (Fig.1).

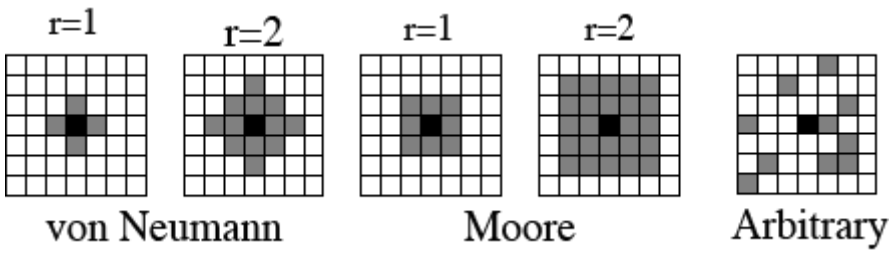

Fig. 1- types of neighborhoods

A seconda del diverso tipo di quartiere, una variegata selezione di celle (di colore grigio) contribuisce all'aggiornamento di una specifica cella (di colore nero). Pertanto, lo stato dei valori contenuti in una cella influiscono direttamente sulle altre celle. Quindi, un automa cellulare si evolve come un'entità continua. L'influenza dei vicini su una cella varia a seconda del tipo di griglia ed una regola determina come una generazione, passa alla generazione successiva rispetto ai suoi vicini.

Il campo degli CA si è evoluto rapidamente da semplici modelli a modelli molto complessi. È stato fatto molto lavoro sulla creazione di griglie bidimensionali e di dimensioni superiori. L'interazione tra le celle e le regole di aggiornamento hanno assunto la forma di espressioni matematiche altamente complicate. Una buona rassegna di lavori esistenti nell'apprendimento delle regole degli automi cellulari è presentato in [19].

Cellular Learning Automata

Un automa di apprendimento (LA) è un'entità semplice che opera in un ambiente casuale e sconosciuto. Nella sua formulazione più semplice, l'automa ha un insieme finito di azioni tra cui scegliere e in ogni fase la sua scelta dipende dal suo vettore di probabilità. Per ogni azione scelta dall'automa, l'ambiente fornisce un segnale di rinforzo con una certa distribuzione di probabilità che è sconosciuta. L'automa quindi aggiorna il suo vettore di probabilità in base al segnale di rinforzo ricevuto in quella fase, e si evolve seguendo il comportamento finale desiderato [22].

Un LA è definito dalla quadrupla {α, β, p, T} in cui α = {α1,α2,…, αn} rappresenta l'insieme di azioni degli automi, β = {β1,β2,…,βr} rappresenta l'insieme di input, p = {p1,p2,…,pr} rappresenta l'insieme di probabilità di azione, e infine p (n + 1) = T [α (n), β (n) , p (n)] rappresenta la funzione di aggiornamento/transizione dell'algoritmo di apprendimento. Questo automa funziona come segue: in base alla probabilità di azione impostata p, l'automa seleziona in modo casuale un'azione αi e la esegue sull'ambiente (Fig.2). Dopo aver ricevuto il segnale di rinforzo dell'ambiente, l'automa aggiorna la sua probabilità di azione impostata sulla base dell'equazione (1) per le risposte favorevoli e sull'equazione (2) per quelle sfavorevoli.

$$pi(n+1)=pi(n)+a.(1-pi(n)) \quad (1)$$

$$pj(n+1)=pj(n)-a.pj(n) \quad \forall j \; j \neq i$$

$$pi(n+1) = (1-b).pj(n) \quad (2)$$

$$pj(n+1)=b/(r-1)+(1-b)pj(n) \quad \forall j \; j \neq i$$

In queste due equazioni, a e b sono parametri di ricompensa e penalità, rispettivamente. Il pieno potenziale di un LA si realizza quando più automi interagiscono tra loro. L'interazione può assumere forme diverse come un albero, un mesh, array e così via. A seconda del problema da risolvere, può essere scelta una di queste strutture per l'interazione. Nella maggior parte delle applicazioni, la piena interazione tra tutti gli automi di apprendimento non è necessaria e non è naturale.

D'altra parte, gli automi cellulari sono modelli matematici per sistemi costituiti da un gran numero di semplici componenti identici con interazioni locali. CA e LA vengono combinati per ottenere un nuovo modello chiamato Cellular Learning Automata (CLA). Questo modello ha prestazioni maggiori rispetto agli automi cellulari per la sua capacità di apprendere ed allo stesso tempo ha prestazioni superiori ai singoli automi di apprendimento perché rappresentano una raccolta di automi che possono interagire tra loro. Pertanto possiamo definire un CLA come un modello matematico per sistemi dinamici complessi composto da un insieme di componenti semplici. I componenti semplici hanno capacità di apprendimento e agiscono insieme per produrre modelli comportamentali complicati.

Un CLA è un automa cellulare in cui un automa di apprendimento sarà assegnato a ogni sua cella [23]. L'automa che apprende e che risiede in ogni cella determina lo stato della stessa sulla base del suo vettore di probabilità. Come gli CA, esiste una regola secondo cui gli automi di apprendimento cellulare operano in base ad essa. La regola degli CLA e le azioni selezionate dai vicini determinano il segnale di rinforzo per l'automa che risiede in una cella (Fig.3). In CLA, i vicini della cella costituiscono il suo ambiente locale. Questo ambiente non è stazionario a causa del fatto che cambia

al variare dei vettori di probabilità di azione degli Automi di Apprendimento vicini [24]. I CLA possono essere classificati in due tipi: *sincrono* e *asincrono*. Negli automi sincroni di apprendimento cellulare, tutte le celle sono sincronizzate con un orologio globale ed eseguite allo stesso tempo [18]. È dimostrato che gli automi sincroni di apprendimento cellulare convergono a uno stato globalmente stabile per una classe di regole chiamate regole commutative.

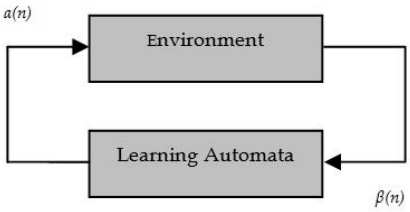

Fig. 2- Learning Automaton model

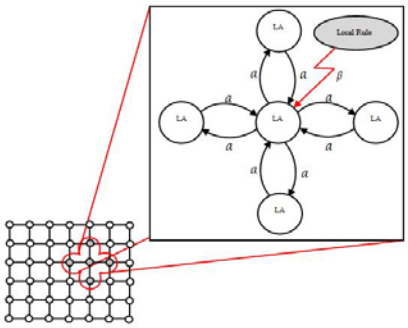

Fig. 3 – Cellular Learning Automaton schema

Cellular Transfer Learning for lighting control system

In letteratura sono diversi i lavori presentati per il controllo dell'illuminazione e degli oscuranti per migliorare il comfort dell'utente e ridurre allo stesso tempo l'energia consumata. Recentemente sono state introdotte tecnologie di apprendimento automatico che consentono di migliorare le prestazioni di controllo inserendo informazioni in tempo reale garantendo adattabilità e flessibilità al sistema considerando il giusto equilibrio tra confort e risparmio, coinvolgendo in tale sistema il feedback sulle percezioni degli utenti [32]. Il sistema proposto è stato realizzato in pratica in un edificio ad alta efficienza energetica. Rispetto al controllo tradizionale, questo sistema ha dimostrato di garantire condizioni di confort luminosi accettabili ed allo stesso tempo ha prodotto un efficientamendo dal punto di vista energetico. L'impianto realizzato, basato sulla soddisfazione degli utenti è costituito da un sistema di controllo a ciclo chiuso che utilizza un controller con apprendimento per rinforzo migliorato per ottenere una strategia di controllo ottimale delle luci e degli oscuranti. Sistema realizzato all'interno del progetto COGITO [35] incentrato sull'integrazione di Internet of Things

(IoT) con sistemi dinamici e cognitivi con l'obiettivo di migliorare la gestione degli edifici pubblici e residenziali con funzionalità cognitive e di self-developed. Si tratta cioè di rendere autonomo, adattivo ed attivo l'ambiente di vita dell'edificio. Una caratteristica chiave che accomuna i sistemi cognitivi, è la raccolta e l'analisi dei dati ambientali e delle informazioni sulle abitudini dei consumatori. Tali informazioni ci consentono di operare in modo proattivo al fine di gestire in modo efficiente le risorse con l'obiettivo di migliorare le prestazioni energetiche e aumentare il livello di benessere e sicurezza per gli stessi abitanti che occupano l'edificio. Per mantenere un buon livello di adattabilità, reattività e proattività del sistema è stato studiato un ciclo di addestramento guidato dal numero di reclami ricevuti in un fissato arco temporale (Fig.4). I reclami possono provenire da due fonti: i sensori presenti nella stanza e dagli utenti che la occupano. Un aumento del numero di reclami provenienti dai sensori è un'indicazione che il sistema sta perdendo la sua proattività, ovvero non riesce più ad anticipare le variazioni delle condizioni climatiche al fine di evitare situazioni di disconfort. Un aumento del numero dei reclami provenienti dagli occupanti indica che sono state segnalate situazioni di abbagliamento o di poca luce in alcune ore della giornata per cui è necessario che il sistema si riadatti ai nuovi requisiti richiesti dagli utenti. Lo schema seguente mostra i meccanismi di innesco tra periodo di addestramento e periodo di produzione con le relative informazioni di input per ogni fase.

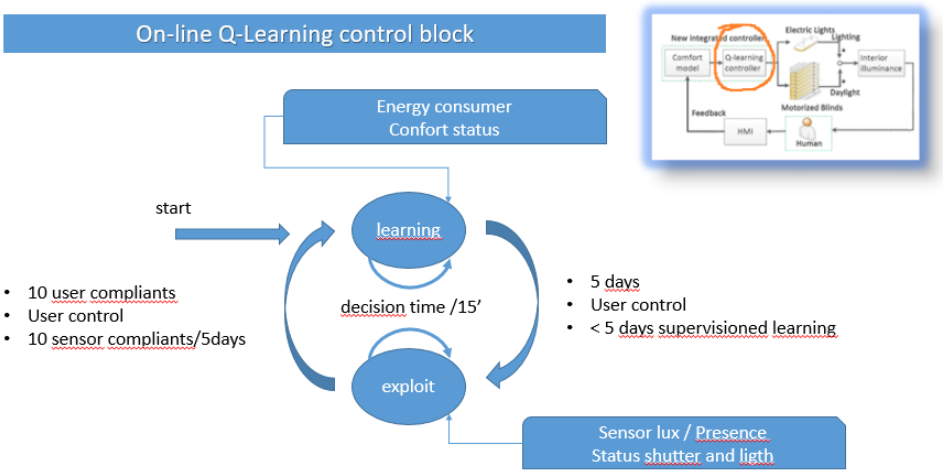

Fig. 4 – Q-Learning on-line training cycle

Considerando di lavorare con uno step decisionale di 15 minuti, si ottiene che in media il tempo di addestramento con interazione diretta con l'ambiente avviene in 5/7 giorni in cui si combinano apprendimento supervisionato e non supervisionato, come mostrano i dati sperimentali in (Fig.5)

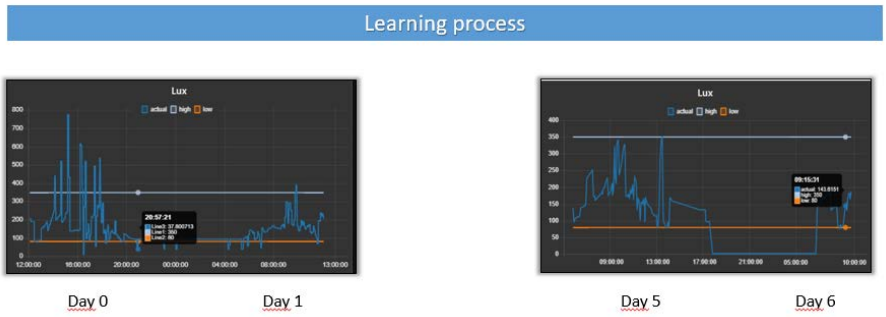

Fig. 5 - Daily training rate

Con una accurata progettazione al fine di ridurre il numero di stati ed azioni, si riesce ad ottenere una convergenza dell'algoritmo di QL in circa 200 iterazioni, come mostrato in (Fig.6).

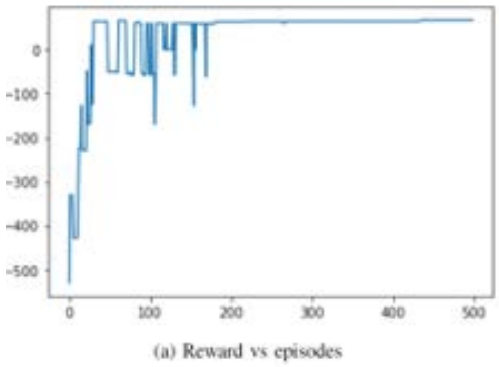

Fig. 6 – Q-Learning on-line convergence test

Considerando un numero medio di episodi per giorno pari a 36 (uno ogni 15 minuti tra le 8 e le 17), in 5 giorni avremo 180 episodi, sufficienti a ridefinire una nuova politica. Tuttavia, i dati raccolti si riferiscono ad una condizione di riaddestramento, ovvero il processo di addestramento considera una matrice Q che ha valori iniziali diversi da zero, adottando una politica Ɛ-greedy con Ɛ=0.5. Nel caso in cui l'algoritmo inizia la fase di addestramento con valori tutti nulli della matrice Q, ovvero non contiene informazioni parziali sull'ambiente, i tempi potrebbero raddoppiarsi.

In un contesto di sistemi multiagente la fase di addestramento dell'algoritmo potrebbe dimezzarsi sfruttando la conoscenza proveniente dai sistemi di controllo limitrofi strutturalmente somiglianti dal punto di vista delle dinamiche ambientali. Parliamo ad esempio di uffici disposti sulla stessa facciata, allo stesso piano di un edificio, che riportano quindi lo stesso orientamento, inoltre se gli ambienti sono adibiti per le medesime funzioni, tipicamente riportano anche una conformità strutturale sugli arredi e le superfici vetrate. Sfruttando queste somiglianze strutturali è possibile sfruttare meccanismi di Transfer Learning per ridurre i tempi necessari per l'addestramento dei vari sistemi di controllo cognitivi.

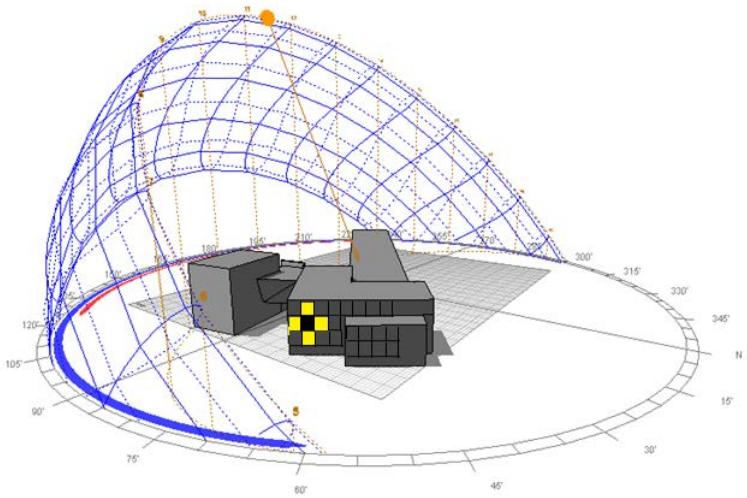

Fig. 7 - Trajectory of the sun's rays

In Fig.7, ogni cella descrive un ufficio che dispone di un nodo di edge computing [33,34] adibito al controllo cognitivo per la gestione del confort visivo. La cella rappresentata in nero rappresenta un agente in fase di addestramento, mentre le celle in giallo gli agenti del vicinato che potrebbero essere utili nel processo di condivisione della conoscenza.

Si vuole realizzare un sistema di trasferimento della conoscenza che coinvolga più agenti nello stesso dominio mantenendo la conformità strutturale degli algoritmi di QL on-line già operanti nel sistema di controllo all'interno dell'edificio. In questo scenario multi-task, assumiamo che più attività svolte nello stesso dominio siano svolte o apprese da uno o più agenti strutturalmente simili (stesso algoritmo di QL, stesso insieme degli stati e delle azioni), e che il processo TL utilizzi le conoscenze raccolte dagli agenti del vicinato che dispongono di un sufficiente livello di rating (agenti virtuosi sia in termini di soddisfazione degli utenti che in termini di risparmio energetico) al fine di migliorare l'apprendimento dell'attività di destinazione. Si assume che gli agenti coinvolti di volta in volta nel processo di TL usano lo stesso spazio stato-azione. Sotto questa ipotesi è possibile un trasferimento peer-to-peer poiché esiste una compatibilità strutturale tra le attività di origine e destinazione. Occorre tuttavia prestare attenzione perché sebbene ci sia compatibilità strutturale, ciò non significa che il trasferimento sia sempre rilevante. I due sistemi di controllo possono avere acquisito modelli differenti o probabilità di transizione significativamente diverse proprio perché operano in ambienti differenti, questo può rendere il trasferimento inefficace o addirittura dannoso in termini di prestazioni di apprendimento. È necessario quindi realizzare metodi mirati che tendano ad utilizzare alcune conoscenze sulle attività di origine per adattare l'attività di destinazione consentendo un processo di trasferimento selettivo.

La configurazione del sistema che si vuole realizzare ha molti elementi in comune con gli automi cellulari, infatti come questi:

- **Parallelismo**. Si rappresenta un sistema dinamico discreto, dove tempo e stati del sistema sono discreti. È possibile un trasferimento della conoscenza simultaneo tra gli agenti coinvolti ed un processo di apprendimento indipendente;
- **Località**. Gli stati delle celle variano secondo una regola locale, cioè lo stato di una cella ad un dato istante di tempo dipende dallo stato della cella stessa e dalle informazioni delle celle vicine. Lo stato di una cella è influenzata soltanto dalla conoscenza dei suoi vicini più prossimi i quali riescono a caratterizzare meglio di altri il modello dell'ambiente che lo circonda;
- **Omogeneità**. Il passaggio da uno stato a quello successivo è dovuto alla composizione di due informazioni, il vicinato, che specifica quali celle influiscono sulla data cella, e la funzione di transizione. È possibile definire un unico algoritmo di controllo in grado di operare in ogni ambiente dell'edificio, invece di realizzare sistemi ad hoc.

Problem modeling and valuation results

Gli CA ci forniscono una metodologia valida per definire e realizzare un algoritmo di distribuzione dei consigli che integri meccanismi di parallelismo, di località ed omogeneità. Similmente agli CLA vogliamo abbinare gli CA con gli algoritmi di RL, nel nostro caso QL, per combinare insieme le potenzialità di un CA con un algoritmo di QL. La soluzione proposta è stata validata applicandola alla soluzione del problema del labirinto (Fig.8). In questo caso, abbiamo utilizzato un labirinto considerando quattro direzioni ammissibili: nord, sud, est, ovest. Ogni transizione produce una ricompensa pari a 0, tranne quando l'agente raggiunge uno stato terminale dove riceve una ricompensa pari a 100.

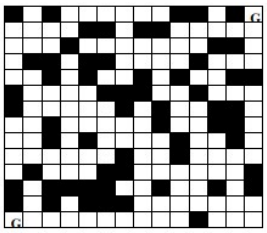

Fig. 8 – Reference maze

Per riprodurre diversi ambienti mantenendo inalterati il numero di stati e di azioni, vengono generati per ogni cella del CA altrettanti labirinti partendo dal labirinto di riferimento (Fig.9). I labirinti

differiscono uno dall'altro, eseguendo in modo random uno scambio tra celle occluse e celle libere. Il numero di celle che si è deciso di scambiare è pari al 10% del totale.

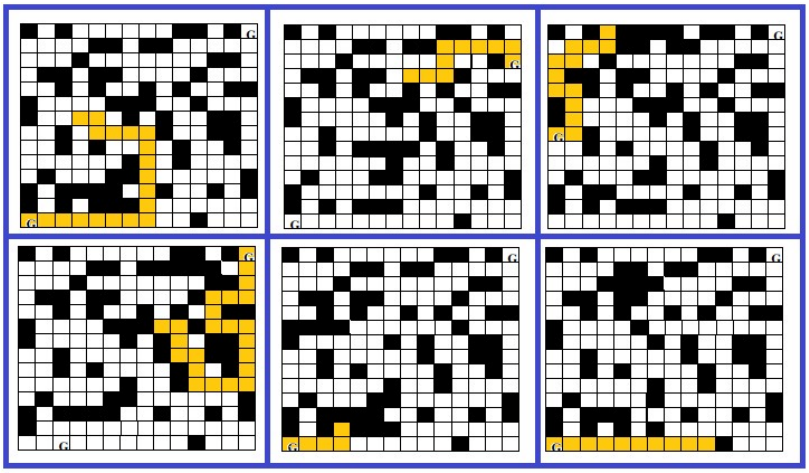

Fig. 9 – Cellular Trasnsfer Learning schema

Affinché una tale struttura di elaborazione possa essere considerata un Automa Cellulare deve soddisfare le tre seguenti caratteristiche: **Parallelismo**. Ogni generazione dell'automa produce un'azione su tutti i labirinti presenti nelle celle; **Località**. Un agente in un labirinto prima di muovere un'azione chiede consiglio ai suoi vicini; **Omogeneità**. La regola di transizione dell'automa e l'algoritmo di RL implementato è identico per ogni cella. Consideriamo lo stato di ogni cella pari a 0 se la cella è in fase di learning e 1 se ha già acquisito una politica ottimale. Ogni cella passa dallo stato 0 allo stato 1 se in trenta cicli di apprendimento consecutivi non rileva una variazione dello score. Lo score è calcolato come sommatoria di tutti i valori contenuti in Q, normalizzati con il suo valore massimo. L'obiettivo è raggiungere il valore 1 per tutte le celle utilizzando in minor numero di generazioni, sfruttando le informazioni provenienti dal vicinato. In questo caso utilizzeremo il vicinato di Von Neumann. L'algoritmo proposto consiste principalmente di tre classi: la classe che realizza l'automa cellulare composta da due matrici rappresentanti rispettivamente lo stato attuale e lo stato successivo dell'automa, la funzione di transizione e la funzione che consente di definire i semi nella configurazione iniziale (celle addestrate con stato 1); La classe che realizza il rendering a video della dinamica dell'automa, basata sulla libreria "allegro". Infine abbiamo la classe che realizza la singola cella dell'automa. Quest'ultima è composta di vari elementi che incapsulano l'algoritmo di QL e funzioni che consentono l'uso selettivo dei consigli sulla base al rating e dell'utilità rispetto al proprio obbiettivo.

Algoritm 2 Transition rule CA

```
transitionFunction (int x, int y)
 1:    if( cells(x,y).getLearning ==1 )
 2:        int state = cells(x,y).getState();
 3:        int north = cells(x-1,y).inference_best_action (state);
 4:        int south = cells(x+1,y).inference_best_action (state);
 5:        int east = cells(x,y+1).inference_best_action (state);
 6:        int west = cells(x,y-1).inference_best_action (state);
 7:        cells(x,y).episode_update(north, south, east, west);
 8:        cells(x,y).learning_update();
 9:        if( cells(x,y).getLearning ==0 )
10:              writeMatrix[x][y].setLearning(0);
```

La funzione di transizione, descritta nell'algoritmo 2, riceve lo stato della cella (learning or not learning), così come il suo stato interno (posizione dell'agente nel labirinto) nonché le informazioni del vicinato. Quest'ultime corrispondono alle best action per lo stato corrente. Al punto 7 viene eseguito uno step di apprendimento della cella considerata, mentre al punto 8, qualora l'avanzamento al punto precedente si fosse concluso con un nodo terminale, si verifica la convergenza della cella andando ad analizzare lo score negli ultimi trenta episodi. Se lo score rimane invariato per trenta episodi successivi si esegue una transizione di stato della cella, punto 10.

Al punto 7 viene seguito uno step di apprendimento dall'algoritmo QL contenuto nella cella, sfruttando le conoscenze provenienti dal vicinato. Queste informazioni costituiscono l'insieme dei consigli che incidono sulla policy della singola cella. La policy che rispetta i requisiti che ci siamo posti viene descritta di seguito nell' algoritmo 3.

Algoritm 3 behavior policy to choose action

```
double Ɛ1=0.8   -- percentage explore vs exploit episodes
double Ɛ2=0.5   -- percentage random vs transfer in explore episodes
Choose_action (north, south, east, west)
 1:  double numb1, numb2 = choose_random_number (0,1);
 2:  if (numb1< Ɛ1)      -- explore
 3:       if (numb2< Ɛ2)    -- random
 4:            action = random (possible_action);
 5:       else    -- transfer learning
 6:            action = best_action_recommended north, south, east, west);
 7:            if ( !find ( action, possible_action ))      -- not admissible action
```

```
 8:                  action = random (possible_action);
 9:                  rating [id_action] - - 1;
10:            else      -- admissible action
11:                  if ( α · (r + γ · TDerror) > Q(s,a) )    -- value function increase
12:                        rating [id_action] ++1;
13:                  else
14:                        rating [id_action] - - 1;
15: else    -- exploit
16:       action = getMaxAction ( state );
```

Rispetto ad una politica Ɛ-greedy classica (algoritmo 1), la fase di esplorazione nel nostro caso si sdoppia secondo il parametro Ɛ2. Una data percentuale di azioni verranno scelte in modo random, utili a garantire l'acquisizione di conoscenze locali sull'ambiente attraverso l'esplorazione, mentre la restante parte verrà scelta selezionando dai consigli provenienti dal vicinato. Quest'ultime rappresentano il bagaglio di conoscenza acquisito dal gruppo e consentono di velocizzare il processo di apprendimento. Tra i quattro consigli provenienti dal vicinato, verrà scelta l'azione proveniente dal vicino con rating maggiore. A parità di rating verrà scelto il consiglio del vicino che produce il maggiore incremento della funzione utilità, linea 6. Se operiamo con ambienti disomogenei potrebbe accadere che il consiglio elargito dal vicino che possiede il più alto rating non sia ammissibile per la cella locale, pertanto si provvederà a scartare il consiglio scegliendone uno random ed infine a ridurre il rating relativo, linea 9. Se il consiglio risulta ammissibile si andrà a valutare se questo produce un incremento della funzione utilità, provvedendo in tal caso anche ad aumentare il rating (linea 12). Nel caso contrario andremo a ridurre il rating ed a eseguire comunque il consiglio dato (linea 14). Nei test condotti in ambiente simulato, si è considerando un CA costituito da 100 celle (Fig.11), dove il colore verde indica una cella QL in fase di produzione ed una cella rossa una cella in fase di addestramento. I test condotti sulla singola cella hanno dimostrato la convergenza a 800 cicli di apprendimento (Fig.10).

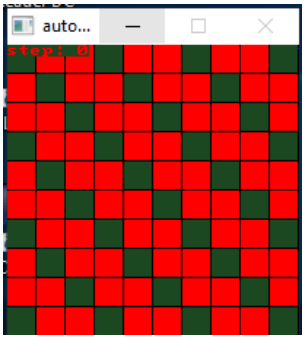

Fig. 11 – Cellular Transfer Learning

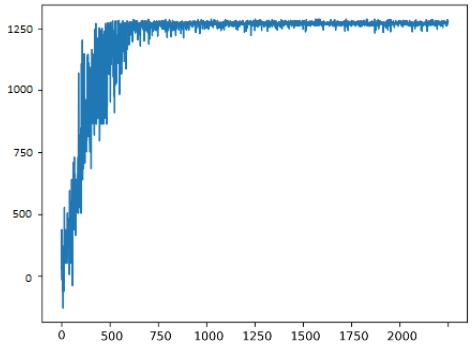

Fig. 10 – Cumulative Q-Value (Score) of single Cell

Analoghi test sono stati svolti considerando un CA composto da 100 celle identiche, operandi su copie dello stesso labirinto in modo indipendente e parallelo. In PQL, gli agenti imparano lo stesso compito in isolamento e in modo indipendente senza interagire tra di loro attraverso l'ambiente, così come avviene anche nel nostro caso. Relativamente a PQL gli agenti apprendono una politica ottima aggiornando la tabella Q condivisa. Nel nostro approccio si utilizza una condivisione diretta della politica intesa come tupla (stato, azione, utilità). I risultati dimostrano che utilizzando una politica Ɛ-greedy, con Ɛ1 = 0.5 (50% exploit) si otterrà un dimezzamento del numero di episodi necessari a trovare una politica ottimale (Fig.13). Se utilizziamo un Ɛ1 = 0.8 (80% exploit) avremo una riduzione del numero di episodi pari circa al 20% (Fig.12). Inoltre conducendo un'analisi di performance sulle singole celle è emerso che il numero medio di step necessari per raggiungere un nodo terminale a partire da uno stato di partenza si dimezza, così come la dispersione media dei learning episodes tra tutte le celle coinvolte. Ciò dimostra che l'approccio utilizzato migliora la velocità del processo di apprendimento di un fattore Ɛ1, la qualità e l'accuratezza del processo, proponendosi come valida alternativa al PQL a tabella Q condivisa.

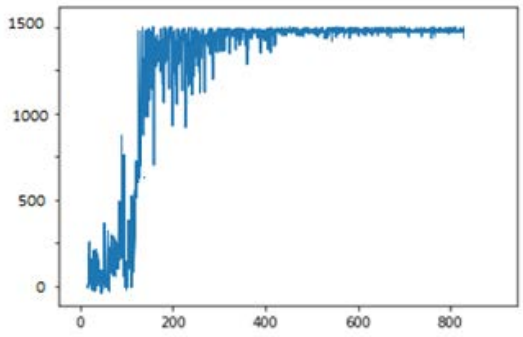

Fig. 13 – Cumulative Q_value Ɛ1 = 0.5 Ɛ2 = 0.5

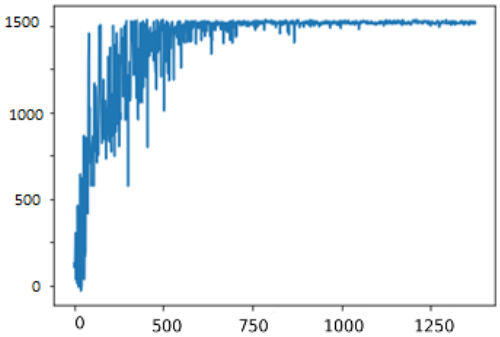

Fig. 12– Cumulative Q_value Ɛ1 = 0.8 Ɛ2 = 0.5

Tecniche come C-MARL sono state introdotte per adattare gli algoritmi RL agli ambienti in cui un team di più RL di agenti interagiscono per perseguire un obbiettivo di squadra. Gli agenti che formano un team collaborano a un obiettivo comune. Rispetto a questo approccio, nel nostro caso ogni agente adotta una politica avida. Ogni agente ha l'obbiettivo di ottimizzare le proprie performance. Questo tuttavia non crea antagonismo nel gruppo, anzi ogni agente perseguendo al meglio il proprio obiettivo diverrà più esperto del dominio e quindi sarà di maggiore aiuto alla squadra.

I test sono stati condotti, disponendo su ogni riga dell'automa un labirinto che si differisce dalla riga adiacente per il 10% di celle scambiate. Sono state introdotte delle celle con tabelle Q pre_addestrate rappresentanti agenti onniscienti che costituiscono i semi del CA. L'analisi è stata condotta inserendo prima il 33% di celle addestrate e poi il 50%. I risultati del test hanno dimostrato miglioramenti in termini di qualità, accuratezza e velocità di apprendimento, rilevando una riduzione della dispersione sui learning episodes ed una riduzione del numero di step necessari per la ricerca del nodo terminale.

Non si rilevano significative variazioni nei casi in cui si avvia l'addestramento senza semi, con il 33% o il 50% di semi. Il parametro che influisce significativamente sulla velocità di apprendimento è la percentuale di azioni scelte dall'automa provenienti dai consigli, Ɛ2. I risultati dimostrano che utilizzando una politica Ɛ-greedy, con Ɛ2 = 0.5 (50% exploit) si otterrà una significativa riduzione del numero di episodi necessari a trovare una politica ottimale, con una riduzione di circa il 15%. Rispetto ai test condotti su celle omogenee, non otterremo in questo caso un decremento del numero di episodi apri a Ɛ2, in quando aumenta la probabilità di ricevere consigli inutilizzabili dai vicini operanti su ambienti differenti, il che produce un rallentamento del processo di apprendimento. Il dimezzamento del numero di episodi si ottiene impostato un Ɛ2 = 0 (100% transfer). Questo tuttavia produce un degrado della qualità ed accuratezza del risultato, registrando una riduzione dello score ed un aumento del numero medio di step necessari per raggiungere un nodo terminale, così come la dispersione media dei learning episodes. La condizione migliore si raggiunge utilizzando un Ɛ2 = 0.8 (80% exploit) in cui avremo una riduzione del numero di episodi pari circa al 20%, con un aumento della qualità ed accuratezza dei risultati.

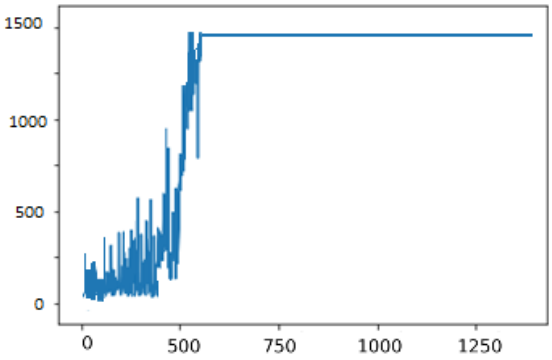

Fig. 15-Cumulative Q_value Ɛ1 = 0.5 Ɛ2 = 0.5

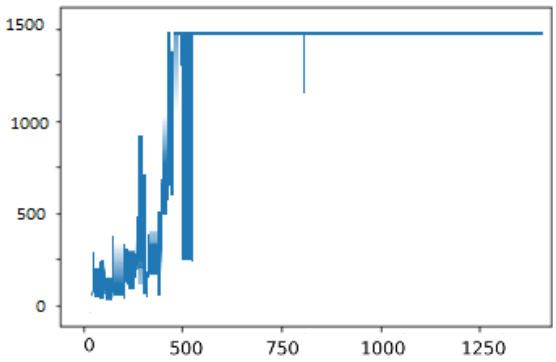

Fig. 14- Cumulative Q_value Ɛ1 = 0.5 Ɛ2 = 0.8

[1] Rogers, E. M. 2010. Diffusion of innovations. Simon and Schuster.

[2] Sukhbaatar, S.; Fergus, R.; et al. 2016. Learning multiagent communication with backpropagation. In Advances in Neural Information Processing Systems, 2244–2252.

[3] Oliehoek, F. A., and Amato, C. 2016. A concise introduction to decentralized POMDPs, volume 1. Springer.

[4] Graves, A.; Bellemare, M. G.; Menick, J.; Munos, R.; and Kavukcuoglu,K. 2017. Automated curriculum learning for neural networks. In International Conference on Machine Learning, 1311–1320.

[5] Daniele Miorandi, Sabrina Sicari, Francesco De Pellegrini, Imrich Chlamtac, Internet of things: Vision, applications and research challenges, Ad Hoc Networks, Vol. 10, issue 7, pp. 1497-1516, Elsevier, 2012.

[6] Ovidiu Vermesan and Peter Friess, "Internet of things-from research and innovation to market deployment", River Publishers Aalborg, 2014.

[7] F. Cicirelli, A. Guerrieri, G. Spezzano, A. Vinci, O. Briante, A. Iera, and G. Ruggeri, "Edge computing and social internet of things for large-scale smart environments development," IEEE Internet of Things Journal, no. 99, 2017.

[8] F. Cicirelli, A. Guerrieri, G. Spezzano, and A. Vinci, “A Cognitive Enabled, Edge-Computing Architecture for Future Generation IoT Environments,” in Proceeding of the IEEE 5th World Forum on Internet of Things, Limerick, Ireland, 2019.

[9] S. Sutton and A. G. Barto. Introduction to reinforcement learn- ing. MIT Press, 1998.

[10] Christopher Watkins and Peter Dayan, Technical Note: Q-Learning, Journal Machine Learning, May 1992, 8(3) ISSN 0885-6125, pp. 279-292.

[11] L. Busoniu, R. Babuska, and B. De Schutter, Multi-agent reinforcement learning: An overview. In Innovations in Multi-Agent Systems and Applications, pp. 183-221, 2010, Springer Berlin Heidelberg.

[12] R. Matthew Kretchmar, Parallel reinforcement learning, 6th World Conference on Systemics, Cybernetics, and Informatics, 2002.

[13] J. Tsitsiklis and B. Van Roy. An analysis of temporal-difference learning with function approximation. IEEE Transactions on Automatic Control, 42(5), pp. 674-690, 1997.

[14] Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A.K. Fidjeland, G. Ostro- vski, S. Petersen, C. Beattie,A. Sadik, I. Antonoglou, H. King, D. Kumaran, D.Wierstra, S. Legg, and D. Hassabis. Human- level control through deep reinforcement learning. Nature,518 (7540), pp. 529-533, 2015.

[15] Sebastian Thrun and Anton Schwartz, Issues in using function approximation for reinforcement learning. In Proceedings of the 1993 Connectionist Models Summer School Hillsdale, NJ.Lawrence Erlbaum. Citeseer, 1993.

[16] Peter Finnman and Max Winberg, Deep reinforcement learning compared with Q-table learning applied to backgammon, Bachelor’s Thesis in Computer Science, KTH Royal Institute of Technology, Sweden, 2016.

[17] Steven D. Whitehead, A Complexity Analysis of Cooperative Mechanism in Reinforcement Learning, Proceedings of the AAAI’91. pp. 607-613, 1991.

[18] Flache A. Hegselmann R. 2002 Do Irregular Grids make a Difference? Relaxing the Spatial Regularity Assumption in Cellular Models of Social Dynamics JASSS, Journal of Artificial Societies and Social Simulation 4 4of 26), 11/4/2002.

[19] Mitchell, M., Crutchfield, J., and Das, R. (1996). Evolving cellular automata with genetic algorithms: A review of recent work. In Proceedings of the First International Conference on Evolutionary Computation and its Applications (EvCA’96).

[20] Berenji, H. and Vengerov, D. (2000a). Advantages of cooperation between reinforcement learning agents in difficult stochastic problems. In Proceedings of 9th IEEE International Conference on Fuzzy Systems.

[21] Tan, M. (1993). Multi-agent reinforcement learning: Independent vs. cooperative learning. In Huhns, M. N. and Singh, M. P., editors, Readings in Agents, pages 487–494. Morgan Kaufmann, San Francisco, CA, USA.

[22] Abolhasani S. M. Meybodi M. R. Esnaashari M. 2007 LABER: A Learning Automata Based Energy-aware Routing Protocol for Sensor Networks IKT conference, Tehran, Iran.

[23] Beigy H. Meybodi M. R. 2004 A mathematical framework for cellular learning automata Advances on Complex Systems 295 320 7Nos. 3-4, September/December, New Jersey,.

[24] Esnaashari M. Meybodi M. R. 2007 Irregular Cellular Learning Automata and Its Application to Clustering in Sensor Networks Proceedings of 15th Conference on Electrical Engineering (15th ICEE), Tehran, Iran 15 17May.

[25] L. Panait, S. Luke Cooperative multi-agent learning: the state of the art Auton. Agents Multi-Agent Syst., 11 (3) (2005), pp. 387-434

[26] R García-Martínez, D Borrajo, P Maceri . Learning by knowledge sharing in autonomous intelligent systems Advances in Artificial Intelligence-IBERAMIA-SBIA 2006, Springer (2006), pp. 128-137

[27] M. Wiering, “Multi-agent reinforcement learning for traffic light control,” in Machine Learning: Proceedings of the Seventeenth International Conference (ICML’2000), 2000, pp. 1151–1158.

[28] S. Shalev-Shwartz, S. Shammah, and A. Shashua, “Safe, multiagent, reinforcement learning for autonomous driving,” arXiv preprint arXiv:1610.03295, 2016.

[29] C. de Vrieze, S. Barratt, D. Tsai, and A. Sahai, “Cooperative multiagent reinforcement learning for low-level wireless communication,” arXiv preprint arXiv:1801.04541, 2018.

[30] S. Q. Zhang, Q. Zhang, and J. Lin, “Efficient communication in multiagent reinforcement learning via variance based control,” in Advancesin Neural Information Processing Systems, 2019, pp. 3230–3239.

[31] T. Rashid, M. Samvelyan, C. S. De Witt, G. Farquhar, J. Foerster, and S. Whiteson, “Qmix: monotonic value function factorisation for deep multi-agent reinforcement learning,” arXiv preprint arXiv:1803.11485, 2018.

[32] Z Cheng, Q Zhao, F Wang, Y Jiang, L Xia, J Ding : Satisfaction based Q-learning for integrated lighting and blind control- Energy and Buildings, 2016 – Elsevier

[33] F. Cicirelli, A. Guerrieri, G. Spezzano, A. Vinci, O. Briante, A. Iera, and G. Ruggeri, “Edge computing and social internet of things for large-scale smart environments development,” IEEE Internet of Things Journal, no. 99, 2017.

[34] F. Cicirelli, A. Guerrieri, A. Mercuri, G. Spezzano, and A. Vinci,“Itema: A methodological approach for cognitive edge computing iot ecosystems,” Future Generation Computer Systems, vol. 92, pp. 189–197, 2019.

[35] F. Cicirelli, A. Guerrieri, G. Spezzano, and A. Vinci, “A Cognitive Enabled, Edge-Computing Architecture for Future Generation IoT Environments,” in Proceeding of the IEEE 5th World Forum on Internet of Things, Limerick, Ireland, 2019.