

Algoritmo di auto-organizzazione per STS basato su stigmergia e sui principi della democrazia di Atene

Antonio Iannotta

January 12, 2022

Abstract

Scopo di questo documento è quello di definire come agenti in un ambiente possano tra di loro interagire e coordinarsi in maniera del tutto autonoma e democratica al fine di conseguire un bene comune. Come verrà successivamente spiegato in maniera più dettagliata il concetto di bene comune può variare; quello che tuttavia non varia è il modo *democratico* con cui gli agenti si coordinano. Soprattutto sfruttando il concetto di stigmergia, il che porta a vedere agenti ed ambienti non più come elementi separati ma come un'unica entità nella quale ogni azione degli agenti lascia tracce sull'ambiente che verranno apprese, elaborate, valutate da altri agenti per le loro operazioni. L'obiettivo è quello quindi di spiegare come stigmergia e principi democratici possano tra di loro coesistere.

1 Introduzione

Il contesto nel quale ci si muove nell'ambito dei Sistemi Distribuiti richiede continue considerazioni su come tali sistemi debbano essere considerati, studiati e come debba esserne studiata l'interazione che questi sistemi hanno tra di loro e con l'ambiente. È in quest'ottica che i STS¹ consentono di fornire una visione attuale e maggiormente astratta dei sistemi, dal momento che si ragiona in ottica non solo di sistemi di computazione interconnessi ma bensì di agenti, astraendo il fatto se si tratti di elementi di computazione (cluster di macchine, sensori per IoT) oppure di esseri umani. Proprio in virtù di questa astrazione l'interesse può essere posto su come i singoli agenti di questi sistemi interagiscano gli uni con gli altri, eventualmente definendo nuovi protocolli e nuovi modi di interazione. Il tipo di relazione tra agenti che si analizza in questo documento è un insieme di tre principi:

- **auto-organizzazione:** principio che si basa sul fatto che il coordinamento tra i vari agenti per il corretto funzionamento del sistema avviene senza una vera unità centrale, ma solo tra interazioni tra le entità stesse.

¹Socio-Technical Systems

- **stigmergia:** l'interazione tra gli agenti avviene sulla base di modifiche all'ambiente nel quale queste entità si trovano.
- **principi democratici di Atene:** discutere in questo contesto tutti i principi e tutte le sfaccettature che costituivano i fondamenti della democrazia Ateniese sarebbe inopportuno. Tuttavia alcuni di quei principi possono essere applicati ai STS. In particolar modo il concetto di azioni basate su una conoscenza condivisa così come il concetto su azioni finalizzate al bene della comunità.

Obiettivo di questo documento è mostrare come tali principi possano trovare riscontro nell'ambito dei STS e, più in generale, dei sistemi distribuiti. Nel capitolo 2 viene illustrato come tali principi possano essere applicati ai STS, presentando anche un algoritmo pensato per applicarli tutti e 3. Nel capitolo 3 viene effettuata l'analisi architetturale di un eventuale sistema che potrebbe in modo ottimale usufruire dell'algoritmo presentato. Il capitolo 4 presenta un esempio di applicazione di questo algoritmo fornendo una formalizzazione basata su process algebra.

2 Auto-organizzazione di STS

In questa sezione l'attenzione è posta maggiormente sull'aspetto informatico. In particolare si vuole presentare un algoritmo che consenta l'autogestione e l'auto-organizzazione di un sistema distribuito che siano basate sul concetto di stigmergia e che abbiano come base di coordinamento i principi democratici Ateniesi. Prima di presentare l'algoritmo è più adatto tracciare sommariamente qual è l'ambiente all'interno del quale questo algoritmo dovrà essere applicato. Il sistema è un sistema distribuito. Parlando di auto-organizzazione e autogestione si intende che il sistema non deve dipendere in maniera più o meno forte da una singola architettura centrale, bensì gli agenti sono chiamati a diventare elementi attivi per fare in modo che il sistema raggiunga costantemente gli obiettivi preposti. Iniziano qui ad essere visibili i principi Ateniesi di cui prima si è accennato.

L'obiettivo di ciascun agente che decide di entrare in questo sistema distribuito deve essere il benessere del sistema del suo complesso, inteso sia come ambiente e contesto nel quale queste entità si muovono sia inteso come benessere delle entità stesse. La stigmergia è il mezzo che si utilizza per raggiungere questo scopo.

Differentemente da tipologie di comunicazione presenti nei tipici sistemi distribuiti, le quali prevedono che gli agenti interagiscano tra di loro in maniera diretta (seppur con modi e tempi differenti), mediante la stigmergia l'interazione tra i vari agenti avviene solo in corrispondenza di determinate tracce che vengono lasciate e trovate nell'ambiente/contesto all'interno del quale questi agenti si trovano ad operare. L'aspetto sicuramente più interessante della stigmergia è che consente forse di implementare al meglio il principio di auto-organizzazione e autogestione consentendo di fare la seguente valutazione: una volta lasciato

un segno nell'ambiente, non è importante quale agente presente all'interno del sistema vedrà quel segnale ed agirà di conseguenza perchè sicuramente l'agente, sulla base di quel segnale, compirà un'azione dalla quale in maniera più o meno diretta sicuramente il sistema trarrà vantaggio. L'algoritmo che si vuole presentare è un algoritmo per una gestione di risorse nell'ambito di un sistema distribuito che opera sui principi sopra-elencati. L'algoritmo è di seguito descritto:

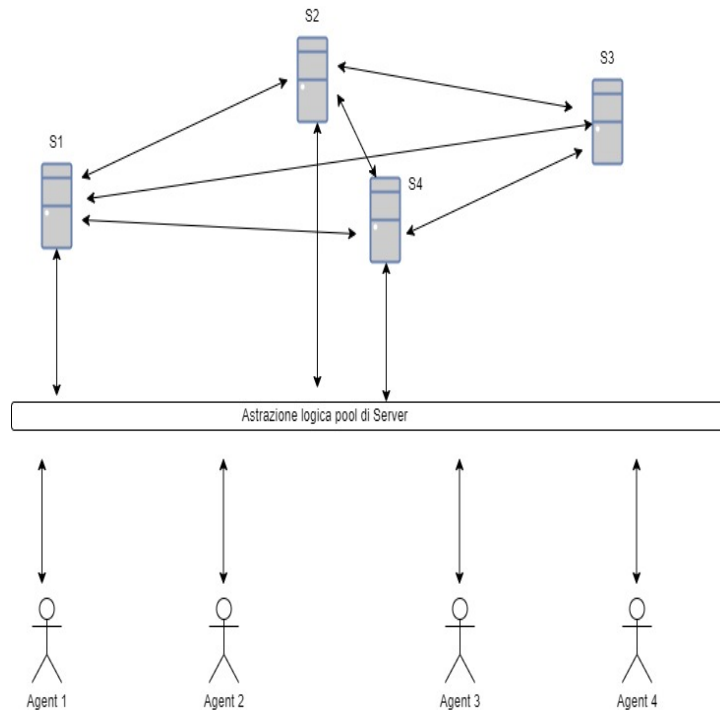
- Un agente entra in questo sistema solo se dispone di una quantità minima di risorse che può offrire al sistema stesso. Questo potrebbe risultare in contraddizione con quanto affermato prima sui concetti di autogestione ma in realtà è l'unica condizione posta sugli agenti, in un contesto che è esterno rispetto a quello nel quale gli agenti vogliono entrare. È possibile immaginare e visualizzare una netta separazione ambientale tra i due spazi.
- Una volta entrato per l'agente inizia l'interazione indiretta con gli altri agenti ed il sistema inizia la propria autogestione. In particolare, volendo scendere nel dettaglio di un sistema distribuito per la gestione di risorse, lo scenario tipico di interazione potrebbe essere quello che coinvolge un agente che ha una mancanza di risorse da fornire e che chiede aiuto semplicemente segnalando questa situazione nell'ambiente. Qui si può rimarcare quanto riferito in precedenza: non è importante chi si assumerà l'onere di aiutare l'agente perchè l'obiettivo primario è che il sistema continui a funzionare, con conseguente attenzione che non è più incentrata sul ruolo dei singoli agenti ma bensì sulle condizioni del sistema nel suo complesso. In questo caso si è potuto evidenziare, seppur in linea sommaria, il ruolo della stigmergia. Nel prossimo punto si segnala invece come un sistema di questo tipo possa adottare principi democratici.
- Si è già ripetuto come interesse primario dell'azione degli agenti sia quello della gestione del sistema nel suo complesso e il benessere del sistema stesso. Da precisare, ancora una volta, è il fatto che il sistema è l'insieme di agenti e ambiente.
Quindi è opportuno considerare la questione in cui il sistema nel suo complesso non sia più in grado di garantire un certo livello di risorse. È a questo punto che entra in gioco il processo democratico del voto, con gli agenti che esprimono una preferenza sulle risorse da incrementare, ad esempio, e con il risultato che viene fornito da una certa politica di voto (maggioranza, minor valore possibile etc..).

3 Architettura

In questa sezione quello che si vuole fornire è una vista dell'architettura corrispondente a questo sistema.

Il sistema si è già detto essere composto da agenti e ambiente all'interno del

quale tali agenti si muovono. Inoltre si è già accennato al fatto che nell'ambito dei STS gli agenti possono sia essere unità computazionali sia esseri umani. Per generalità consideriamo gli agenti astraendo la loro natura fisica e li consideriamo come entità capaci di inviare segnali e di comunicare all'interno di un ambiente (sensori, dispositivi wearable, smartphone, computer, server). Occorre a questo punto concentrarsi sull'ambiente. È evidente, sulla base di quanto descritto nell'algoritmo come sia necessaria un'architettura in grado di mettere a disposizione una risorsa, condivisa a tutti gli agenti, e che gli agenti stessi possano considerare come ambiente. L'architettura che si è pensata in questo contesto è quella di un pool di server che mantengano ognuno una copia di una coda LIFO e una coda di voto, e che forniscano queste due risorse a tutti gli altri agenti di modo che ogni modifica possa essere visualizzata con tempistiche molto vicine a quelle real-time. L'architettura che si è immaginata è quella di un'interfaccia verso un server logico con la quale gli agenti possono interagire. Questo server logico altro non è che l'astrazione fornita dal pool di server nel loro atto di scambiarsi informazioni al fine di fornire un'unica pila LIFO ed un'unica coda di voto a tutti gli agenti. Il modo con cui questi server possano adempiere a tale operazione potrebbe esser visto nel concetto di State Machine Replication, con tanti server che riportano lo stesso stato (in questo caso la risorsa LIFO) e che comunicano tramite specifici protocolli in modo da attuare al meglio la replicazione della risorsa andando a preservare una certa consistenza dei dati. Evidente come, in un sistema di questo tipo, la consistenza risulta fondamentale forse più della disponibilità. Un certo ritardo temporale è tollerabile dagli agenti a patto che le informazioni che andranno a reperire siano coerenti. Altro aspetto importante per il quale si è deciso di utilizzare un'interfaccia fornita da un pool di server ed in generale il concetto di State Machine Replication è quello della fault-tolerance. Di seguito si presenta graficamente l'architettura che si è immaginata per l'applicazione di questo algoritmo.



4 Esempio

L'esempio che qui si riporta è quello di un sistema di memorizzazione distribuita che si auto-organizza e autogestisce secondo i principi riportati sopra. L'algoritmo è il seguente:

4.1 Avvio del sistema

Ogni agente, sia esso il primo sia esso l'ultimo che vuole entrare nel sistema deve essere in possesso dei requisiti di memoria stabiliti dall'esterno. Abbiamo quindi un agente che vuole entrare nel sistema e, per farlo, deve possedere un certo quantitativo di memoria da mettere a disposizione. Se si ritrova in possesso di questo requisito, all'atto di registrazione questa viene accettata altrimenti rifiutata.

4.2 Sistema già avviato

Nel caso di sistema già avviato possono verificarsi due situazioni che comportano conseguenze:

- Un agente ha la memoria che scende sotto la soglia prefissata: a questo punto l'agente segnala questo evento sulla coda. Il primo agente che legge

questa situazione dalla pila soccorre il richiedente aiuto andando ad alleggerire la sua memoria trasferendo su di se dati in eccesso. A questo punto gli eventi che possono verificarsi sono due:

- L’agente che aiuta riesce a completare il trasferimento di memoria. In questo caso entrambi gli agenti riprendono la loro la loro attività.
 - Nel caso in cui l’operazione di alleggerimento comporti per l’aiutante una mancanza della sua memoria il ciclo si ripete iterativamente con l’aiutante in questo caso che lascia un messaggio nella coda per chiedere aiuto. Il ciclo si ripete per ogni agente per il quale dovesse verificarsi una situazione di mancanza di memoria.
- La quantità di memoria media del sistema scende sotto la soglia prefissata. A questo punto il sistema entra in uno stato di blocco, e a disposizione degli agenti entra in gioco la coda messa a disposizione dall’interfaccia del server astratto per depositare i voti. I voti altro non sono che la quantità di memoria aggiuntiva che ciascun agente può mettere a disposizione. La quantità di memoria che viene scelta è la più bassa tra quelle votate, di modo che resta sempre spazio per operazioni di recovery con future possibilità di incrementare la memoria disponibile. A questo punto nella coda LIFO compare il risultato della votazione e il primo agente che lo legge, cambia la propria memoria sulla base di quanto letto e scrive sulla coda il cambiamento effettuato. Da qui in poi chiunque legga che è stato effettuato un cambiamento di memoria sa di quanto incrementare e sa anche a quale evento corrisponde dal momento che l’unica situazione nella quale la memoria viene cambiata è immediatamente dopo la votazione.

Di seguito si formalizza il sistema utilizzando le algebre dei processi. Il primo passo è quello di definire una grammatica per la coda LIFO, che risulta essere la seguente:

- $LIFO \Rightarrow (E \cup LIFO | \langle E \rangle \cup LIFO | \emptyset)$
- $E \Rightarrow e$

La prima produzione di questa grammatica indica la coda come un evento E che è stato inserito nella coda LIFO, come un evento che è in attesa di essere inserito nella coda LIFO oppure come l’insieme vuoto. Per tale grammatica valgono i seguenti assiomi:

- $X \cup Y \equiv Y \cup X$
- $X \cup (Y \cup Z) \equiv (X \cup Y) \cup Z$
- $X \cup \emptyset \equiv X$

A questo punto si deve definire una grammatica di eventi per la coda LIFO. Prima di definire tale grammatica formalmente si elencano quelli che sono gli eventi che sono stati definiti con il loro relativo significato. Questa definizione risulterà di grande rilevanza al momento di definire l’*LTS* relativo al sistema concorrente:

- $\text{ghn}(E)^2 \rightarrow$ definisce l'evento di osservazione di un intervento di aiuto effettuato da un altro agente. Mediante tale evento un agente riconosce che c'è stato un evento di aiuto relativo ad un precedente messaggio di aiuto.
- $\text{gvr}(E)^3 \rightarrow$ definisce l'evento di osservazione del risultato della votazione per stabilire quanta memoria ciascun agente deve aggiungere.
- $\text{ghr}(E)^4 \rightarrow$ definisce l'evento di osservazione di un messaggio di carenza di memoria lasciato da un agente
- $\text{gmc}(E)^5 \rightarrow$ definisce l'evento di osservazione della quantità di memoria cambiata da un altro agente. È uno dei due modi mediante i quali un agente può comprendere il quantitativo di memoria da aggiungere.
- $\text{phn}(E)^6 \rightarrow$ definisce l'evento mediante il quale un agente lascia sulla coda il messaggio relativo ad un suo intervento nei confronti di un altro agente in difficoltà.
- $\text{phr}(E)^7 \rightarrow$ definisce l'evento mediante il quale un agente lascia sulla coda il messaggio relativo ad una propria carenza di memoria.
- $\text{pmc}(E)^8 \rightarrow$ definisce l'evento mediante il quale un agente lascia un messaggio sulla coda notificando l'avvenuto cambiamento della propria memoria.

A questo punto è possibile definire la grammatica per gli eventi relativi alla coda LIFO:

- $E\text{-LIFO} \Rightarrow !O\text{-LIFO} | ?I\text{-LIFO} | \tau$
 - $O\text{-LIFO} \Rightarrow \text{ghn}(E) | \text{ghr}(E) | \text{gvr}(E) | \text{gmc}(E)$
 - $I\text{-LIFO} \Rightarrow \text{phn}(E) | \text{phr}(E) | \text{pmc}(E)$

Il simbolo ? indica che si tratta di eventi che si verificano in input alla coda LIFO, mentre il simbolo ! denota gli eventi che si verificano in output alla coda. A questo punto è possibile definire un *LTS* per la coda LIFO con le corrispondenti regole di transizione:

- $\text{LIFO} \cup \langle e \rangle \xrightarrow{\tau} \text{LIFO} \cup e \text{ [INSERTED]}$
- $\text{LIFO} \cup e \xrightarrow{!(e)} \text{LIFO} \text{ [HELP NOTIFICATION OUTPUT]}$
- $\text{LIFO} \cup e \xrightarrow{!\text{ghr}(e)} \text{LIFO} \text{ [HELP REQUEST OUTPUT]}$

²get help notification

³get vote results

⁴get help request

⁵get memory change

⁶put help notification

⁷put help request

⁸put memory change

- $\text{LIFO} \cup e \xrightarrow{!gvr(e)} \text{LIFO} \text{LIFO}$ [VOTE RESULT OUTPUT]
- $\text{LIFO} \cup e \xrightarrow{!gmc(e)} \text{LIFO} \text{LIFO}$ [MEMORY CHANGE OUTPUT]
- $\text{LIFO} \xrightarrow{?phr(e)} \text{LIFO} \text{LIFO} \cup \langle e \rangle$ [HELP REQUEST INPUT]
- $\text{LIFO} \xrightarrow{?phn(e)} \text{LIFO} \text{LIFO} \cup \langle e \rangle$ [HELP NOTIFICATION INPUT]
- $\text{LIFO} \xrightarrow{?pmc(e)} \text{LIFO} \text{LIFO} \cup \langle e \rangle$ [MEMORY CHANGE INPUT]

Dopo aver definito la grammatica e l'*LTS* relativo alla coda LIFO è necessario eseguire la stessa operazione per gli agenti:

- $A \Rightarrow ghn(E).A | ghr(E).A | gvr(E).A | gmc(E).A | phn(E).A | phr(E).A | vote(E).A | help(E).A | pmc(E).A | 0$

È possibile notare come in questa grammatica siano presenti due operazioni aggiuntive non riportate nella coda LIFO: *help* e *vote*. Tali operazioni non sono state riportate per le seguenti ragioni:

- operazione di voto: tale operazione viene effettuata su una coda differente dalla LIFO, quello che è di interesse è soltanto il fatto che dalla coda LIFO potrà essere letto il risultato della votazione di modo di procedere al cambiamento della memoria.
- operazione di aiuto: questa operazione è l'unica che costituisce una comunicazione diretta tra più agenti. Quindi l'azione di aiuto non prevede alcuna comunicazione con la pila. Volendo ragionare in ottica temporale l'operazione di aiuto è esattamente a metà tra una richiesta di aiuto e una notifica di avvenuto intervento.

A questo punto si definisce una grammatica per gli eventi relativi agli agenti:

- $E-A \Rightarrow !O-A | ?I-A | \tau$
 - $O-A \Rightarrow phn(E) | help(E) | phr(E) | vote(E) | pmc(E)$
 - $I-A \Rightarrow ghn(E) | ghr(E) | gvr(E) | gmc(E)$

Negli eventi di Output è riportato *help(E)* l'evento E che l'operazione *help* prende come argomento è da intendersi come l'identificativo dell'agente che ha effettuato la richiesta di aiuto. Si è deciso qui di astrarre dalla logica dei messaggi della coda. Tutti i messaggi sono un evento, generico, tuttavia al momento dell'implementazione questi eventi dovranno essere discriminati sulla base delle rispettive specifiche. Si definisce ora l'*LTS* relativo agli agenti:

- $ghn(e).A \xrightarrow{?ghn(e)} A A$ [HELP NOTIFICATION INPUT]
- $ghr(e).A \xrightarrow{?ghn(e)} A A$ [HELP REQUEST INPUT]

- $\text{gvr}(e).A \xrightarrow{?_{\text{gvr}}(e)} {}_A A$ [VOTE RESULT INPUT]
- $\text{gmc}(e).A \xrightarrow{?_{\text{gmc}}(e)} {}_A A$ [MEMORY CHANGE INPUT]
- $\text{phn}(e).A \xrightarrow{!_{\text{phn}}(e)} {}_A A$ [HELP NOTIFICATION OUTPUT]
- $\text{phr}(e).A \xrightarrow{!_{\text{phr}}(e)} {}_A A$ [HELP REQUEST OUTPUT]
- $\text{vote}(e).A \xrightarrow{!_{\text{vote}}(e)} {}_A A$ [VOTE OPERATION]
- $\text{pmc}(e).A \xrightarrow{!_{\text{pmc}}(e)} {}_A A$ [MEMORY CHANGE OUTPUT]
- $\text{help}(e).A \xrightarrow{!_{\text{help}}(e)} {}_A A$ [HELP OPERATION]

Prima di procedere con la definizione della grammatica per il sistema coordinato è necessario riportare un ulteriore assioma (quello che consente una scelta delle operazioni) valido sia per la coda LIFO che per gli agenti:

- $X+Y \equiv Y+X$
- $X+(Y+Z) \equiv (X+Y)+XZ$
- $(X+Y).Z \equiv X.Z+Y.Z$

A questo punto è necessario definire la grammatica per il sistema coordinato:

- $\text{CS} \Rightarrow \text{AL} || \text{LIFO}$
- $\text{AL} \Rightarrow (A || \text{ALAL}) | 0$

L'operatore $||$ è quello che indica le operazioni parallele. In particolare in questo caso il sistema coordinato è l'esecuzione in parallelo di uno o più agenti e di una coda LIFO. Per l'operatore parallelo valgono le seguenti proprietà:

- $X || Y \equiv Y || X$
- $X || (Y || Z) \equiv (X || YX) || Z$
- $X || 0 \equiv X$

Per la definizione che segue del *LTS* relativo al sistema coordinato con A_i si fa riferimento ad un generico agente della lista di agenti che in quel momento esegue una certa operazione a fronte di certe precondizioni. Si definisce ora una grammatica degli eventi per il sistema coordinato:

- $E\text{-CS} \Rightarrow \text{phr}(E) | \text{vote}(E) | \text{pcm}(E) | \text{help}(E) | \text{phn}(E)$

Nell'ottica del sistema coordinato di seguito viene riportato il significato di tali eventi:

- $phr(E)$ indica l'evento relativo ad un messaggio di carenza di aiuto depositato sulla coda LIFO da parte di un agente della lista
- $vote(E)$ indica l'evento relativo all'operazione di voto effettuata da un agente della lista
- $pcm(E)$ indica l'operazione di cambiamento di memoria effettuato da un agente a fronte di determinate condizioni
- $phn(E)$ indica l'evento relativo alla notifica di avvenuto intervento da parte di un agente nei confronti di un altro in difficoltà
- $help(E)$ indica l'evento di aiuto di un agente verso un altro a fronte di determinate condizioni

Di seguito si riporta l' LTS relativo al sistema coordinato:

- $$\frac{ghr(e).A_i \xrightarrow{?ghr(e)} A_i \quad LIFO \cup e \xrightarrow{!ghr(e)} LIFO \quad [HELP]}{AL || help(e).A_i || AL || LIFO \xrightarrow{help(e)} CSAL || A_i || AL || LIFO}$$
- $$\frac{[HELP]}{AL || phn(e).A_i || AL || LIFO \xrightarrow{phn(e)} CSAL || A_i || AL || LIFO} \quad [HELP \text{ NOTIFICATION}]$$
- $$\frac{(gvr(e).A_i \xrightarrow{?gvr(e)} A_i \quad LIFO \cup e \xrightarrow{!gvr(e)} LIFO) + (gmc(e).A_i \xrightarrow{?gmc(e)} A_i \quad LIFO \cup e \xrightarrow{!gmc(e)} LIFO)}{AL || pmc(e).A_i || AL || LIFO \xrightarrow{pmc(e)} CSAL || A_i || AL || LIFO} \quad [MEMORY \text{ CHANGE}]$$
- $$AL || phr(e).A_i || AL || LIFO \xrightarrow{phr(e)} CSAL || A_i || AL || LIFO \quad [HELP \text{ REQUEST}]$$
- $$AL || vote(e).A_i || AL || LIFO \xrightarrow{vote(e)} CSAL || A_i || AL || LIFO \quad [VOTE \text{ OPERATION}]$$