ALMA MATER STUDIORUM – UNIVERSITA' DI BOLOGNA

SEDE DI CESENA
SECONDA FACOLTA' DI INGEGNERIA CON SEDE A CESENA
CORSO DI LAUREA IN INGEGNERIA DEI SISTEMI E DELLE TECNOLOGIE
DELL'INFORMAZIONE

---

Sistemi intelligenti Distribuiti.

# COOPERATIVE COMPUTING

Sviluppato da: Stefano Olivieri

Anno Accademico 2004 / 05

---

# Indice generale

# 1 Introduzione

Il sistema in oggetto si prefigge come obiettivo quello di realizzare un possibile modello per affrontare un generico problema seguendo l'approccio dettato dal Cooperative Computing.

In particolare, dunque, il sistema non è finalizzato alla risoluzione di una specifica classe di problemi ma vuole fornire un infrastruttura di supporto per una soluzione del problema basata sul concetto di cooperazione tra diverse entità distinte.

L'idea alla base del modello in costruzione è quella di poter scomporre un qualsiasi problema in un insieme di distinti **sottoproblemi** .

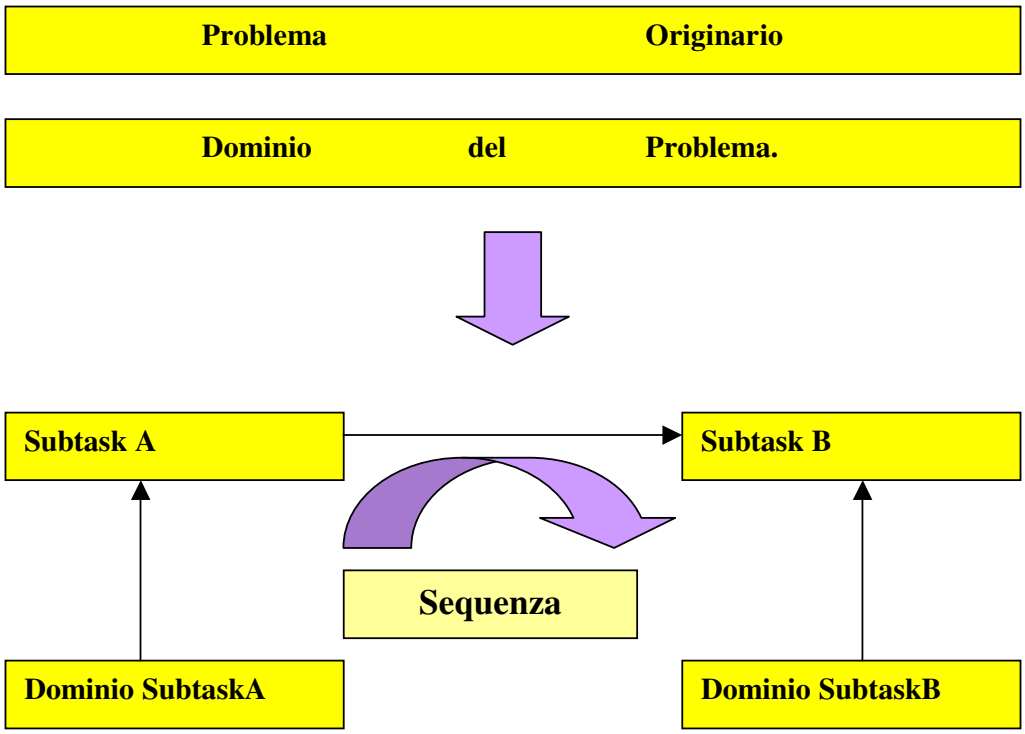
In tale scenario il sistema in oggetto ha il compito di trattare il problema specifico come un parametro variabile, mentre la metodologia con la quale viene gestita la scomposizione del problema rimane invariata e fattorizzata nel modello.

Una peculiarità fondamentale che il sistema dovrà possedere sarà la scalabilità, infatti il sistema dovrà prevedere il fatto che un numero fortemente variabile di entità possano rendersi disponibili predisponendo opportune politiche di assegnamento.

Dunque il problema fondamentale trattato nel progetto è quello della **coordinazione**, l'input del sistema è la sequenza dei task da compiere, i rispettivi domini e gli algoritmi di risoluzione; il sistema ha il compito di armonizzare e coordinare l'esecuzione di questi sottoproblemi da parte di entità esterne che, deliberatamente, decidono di prestare la loro opera mettendosi a disposizione del sistema.

Come detto l'input del sistema è la decomposizione funzionale del problema originario in subtask distinti; tuttavia occorre introdurre il concetto di sequenza per descrivere la dinamica del funzionamento del sistema stesso: a ogni subtask

viene associato un dominio composto da un insieme di elementi, si ipotizza che il problema originario consista nel compiere una certa operazione su ogni elemento del dominio; ecco che tale operazione viene decomposta in subtask e l'applicazione di tutti i subtask a un dato elemento del dominio prende il nome di sequenza.

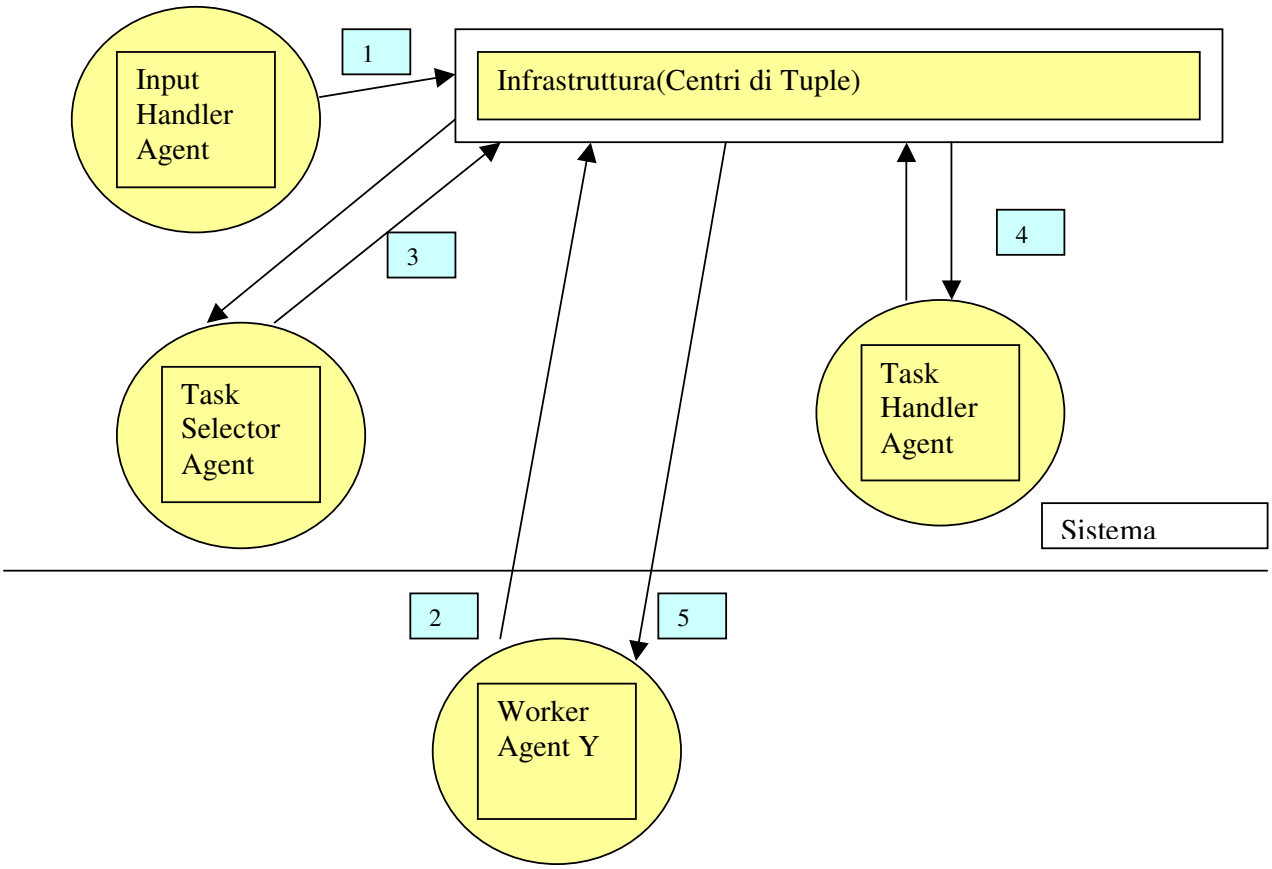


# 2 Struttura del sistema

Il sistema è formato da una comunità di agenti, in questo caso il comportamento del sistema è un comportamento **emergente**, che deriva dall'attività congiunta di tutti gli agenti che compongono il sistema.

Ogni agente è indipendente degli altri, questo non significa che gli agenti sono scorrelati nelle loro attività ma significa che ogni agente ha un proprio compito da portare a termine e nessun agente può manipolare gli obiettivi di un altro agente.

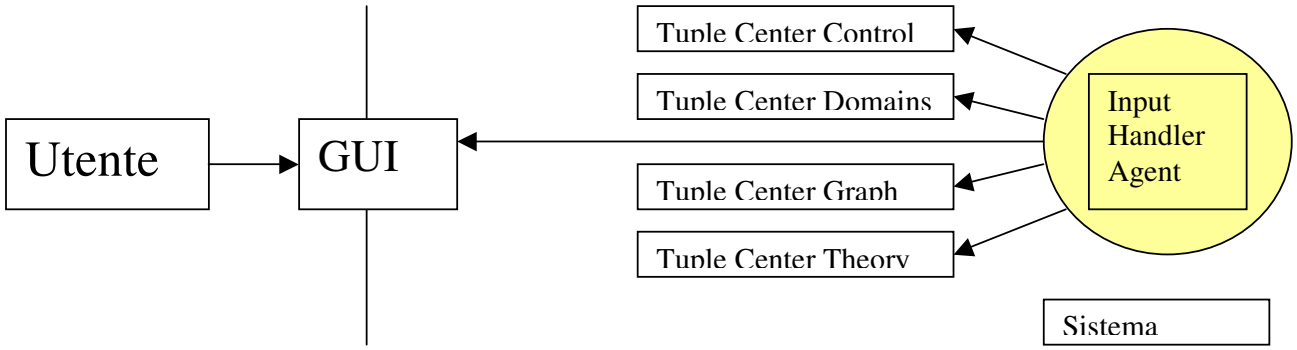
Gli agenti sono relazionati mediante i dati su cui lavorano, ogni agente ha infatti una propria funzione e svolge una determinata funzione all'interno del sistema, dunque gli agenti si relazionano mediante i dati su cui lavorano, in questo modo si impone un ordine nella loro attività.



### 2.1 InputHandlerAgent

Tale agente ha il compito di raccogliere la descrizione del problema originario in termini di sottoproblemi, i relativi domini e gli algoritmi di risoluzione.

Tali elementi vengono conservati e resi disponibili attraverso la conversione in tuple da immettere in relativi centri di tuple.




### 2.2 TaskSelectorAgent

Tale agente ha il compito di determinare quale sarà il prossimo subtask ad essere eseguito da un certo workerAgent che si trova ozioso in quel momento.

L'attività del selettore viene proprio avviata dal fatto che in quel momento esista un agente "lavoratore" che non ha un compito da portare a termine.

L'assegnazione del task da parte del selettore è un attività molto delicata, occorre tenere presente diversi fattori, primo tra tutti quello di sequenza.

Come detto il sistema concettualizza la sua attività come una successione di operazioni da compiere su un certo elemento: ecco che **l'insieme delle operazioni previste da tutti i subtask applicate a un elemento del dominio viene detto sequenza.**

Il selettore sceglie i subtask in maniera tale che le sequenze attive contemporaneamente siano non più di una; questo significa che prima di iniziare a processare un nuovo elemento del dominio il sistema tende a terminare l'insieme delle operazioni associate a un elemento già selezionato.

Questo permette di massimizzare il numero di sequenze completate in un certo istante, se infatti in un dato istante venissero a mancare gli agenti Worker ecco che

le sequenze attive e incomplete sarebbero solamente una mentre se si avviassero parallelamente diverse sequenze le sequenze incomplete sarebbero molteplici.

Dunque il selettore opera tenendo conto che:

- Occorre minimizzare il numero delle sequenze incomplete.
- Non si devono avviare subtask relativi a elementi del dominio che sono già stati eseguiti oppure che sono ora in esecuzione.
- Anche nel caso in cui una sequenza non è terminata non si può avviare un subtask che è dipendente da un subtask non ancora finito.

L'ultima condizione tiene conto delle dipendenze tra i subtask, anche se una sequenza non è terminata non è possibile avviare un subtask che dipende da subtask non ancora terminati perché, appunto,per poter essere eseguito tale subtask necessita del risultato del subtask ancora in esecuzione, ecco il concetto di dipendenza.

Il selettore avvia una nuova sequenza solo nel caso in cui tutti i subtask di una data sequenza sono terminati oppure sono in fase di esecuzione, se una data sequenza non è ancora terminata ma non è possibile avviare un subtask in quanto dipendente da altri subtask non ancora terminati allora l'agente rimane in stato ozioso sino a che il subtask termina e così è possibile assegnare quello successivo.

***Algoritmo di selezione.***

```
end_all([],IdSeq).
end_all([H|T],IdSeq) :- end(IdAgent,H,IdSeq),end_all(T,IdSeq).
parents(NODE,GEN) :-setof(X,arch(X,NODE),GEN),!.
parents(NODE,[]):-not arch(X,NODE).\n"+
start(TasktoStart,IdSeq) :-
arch(TasktoStart,Y),
not end(IdAgent,TasktoStart,IdSeq),
not doing(TasktoStart,IdSeq),
parents(TasktoStart,[]),!.
start(TasktoStart,IdSeq) :-
(arch(TasktoStart,D);arch(D,TasktoStart)),
```

```
not end(IdAgent,TasktoStart,IdSeq),
not doing(TasktoStart,IdSeq),
parents(TasktoStart,List),
end_all(List,IdSeq),!.
```

## 2.3 TaskHandlerAgent

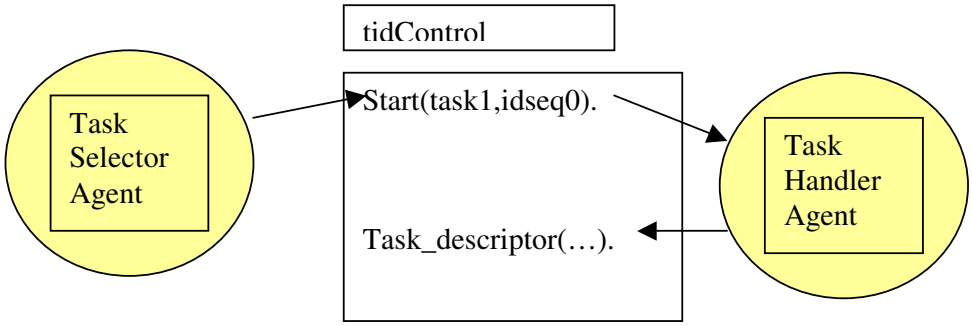
L'agente TaskHandler si occupa di mettere in pratica la scelta del selettore; il selettore, come detto, sceglie, di volta in volta il subtask da assegnare all'agente Worker in quel momento libero, tuttavia il selettore esaurisce il suo compito indicando quale dovrà essere il subtask da attivare ma non si preoccupa di gestire il come tale subtask verrà assegnato all'agente Worker; di questo si occupa il gestore.

Il gestore (TaskHandlerAgent) è in costante attesa che il selettore produca l'indicazione su quale task debba venire attivato, quando la riceve si occupa di:

- recuperare la teoria del subtask.
- recuperare un elemento del suo dominio.
- se il subtask da attivare è dipendente, allora occorre recuperare i risultati dei subtask da cui esso dipende e inserirli nel suo dominio.

Il gestore produce come risultato un **descrittore** del task da attivare, il quale contiene le seguenti informazioni:

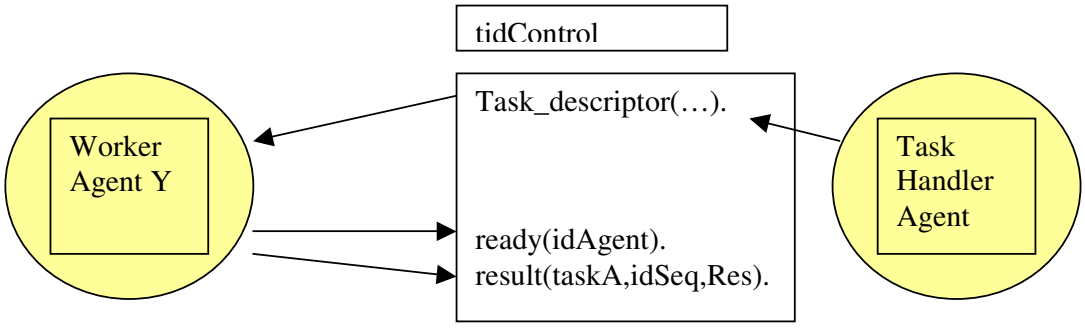
- Nome del subtask da attivare.
- Id della sequenza a cui appartiene.
- Dominio del subtask (elemento).
- Algoritmo del subtask.
- Eventuali risultati di subtask da cui quello corrente dipende e appartenenti alla stessa sequenza.




## 2.4 WorkerAgent

Gli esecutori fisici dei subtask sono gli agenti “lavoratori”: con questa espressione si intende indicare entità che accedono in maniera attiva al sistema mettendosi alle dipedenze di esso.

Il WorkerAgent manifesta esplicitamente la suà volontà a svolgere dei compiti che gli vengono impartiti, in particolare una volta che comunica al sistema la propria disponibilità si mette in attesa di un descrittore di subtask in quanto attraverso esso potrà percepire tutte le informazioni necessarie a svolgere il task.




Quando un AgentWorker termina il suo lavoro ecco che produce un risultato e ritorna pronto, cioè disponibile a eseguire nuovi compiti, questo stato

dell'AgentWorker fornisce lo stimolo per il selettore a scegliere un nuovo subtask da eseguire e il ciclo rincomincia.