

Course Timetabling Problem

PROGETTO

Laboratorio di Intelligenza Artificiale

Carolina Crespi

23 Settembre 2026

1 Course Timetabling Problem – CTP

Il *Course Timetabling Problem* (CTP) è un problema di ottimizzazione combinatoria, classificato come NP-hard, con applicazioni naturali nella pianificazione oraria di lezioni universitarie, esami, turni di lavoro e attività condivise.

Intuitivamente, il problema modella la situazione in cui, dato un insieme di eventi (lezioni) che devono essere programmati in un numero limitato di slot temporali, si vuole assegnare a ciascun evento uno slot in modo che:

- eventi *incompatibili* (perché condividono un docente, un'aula o un gruppo di studenti) ricevano slot diversi (*vincolo hard*, non violabile);
- le preferenze temporali di ciascun evento siano rispettate per quanto possibile (*vincolo soft*, la cui violazione è ammessa ma comporta una penalità).

Questa formulazione con vincoli hard e soft è tipica dei problemi di timetabling reali, in cui alcune condizioni sono strutturali (non ci possono essere due lezioni contemporanee nella stessa aula) mentre altre esprimono preferenze operative (un docente preferisce non insegnare al primo slot della mattina). Alcuni esempi applicativi:

- la costruzione dell'orario didattico di un corso di laurea, con vincoli su aule e docenti e preferenze su fasce orarie;
- la pianificazione degli esami di fine sessione, evitando sovrapposizioni per gli studenti e distribuendo il carico nel tempo;
- la gestione di turni ospedalieri, in cui alcuni vincoli sono di legge (riposi minimi) e altri di preferenza (giorni liberi richiesti).

Formalmente, siano dati:

- un insieme di n eventi $\mathcal{E} = \{E_1, \dots, E_n\}$;
- un insieme di T slot temporali $\mathcal{T} = \{1, \dots, T\}$;
- un grafo di conflitto non orientato $G = (\mathcal{E}, \mathcal{C})$, dove $(E_i, E_j) \in \mathcal{C}$ indica un conflitto hard;
- per ciascun evento E_i , un insieme di slot preferiti $P_i \subseteq \mathcal{T}$.

Introducendo le variabili $x_i \in \{1, \dots, T\}$, che rappresentano lo slot assegnato all'evento E_i , il problema si formula come:

$$\min \sum_{i=1}^n \mathbb{K}[x_i \notin P_i] \quad (1)$$

soggetta ai vincoli hard:

$$x_i \neq x_j \quad \forall (E_i, E_j) \in \mathcal{C}, \quad (2)$$

dove $\mathbb{I}[\cdot]$ è la funzione indicatrice (vale 1 se la condizione è vera, 0 altrimenti). La funzione obiettivo conta dunque il numero di eventi assegnati a uno slot non preferito, mentre i vincoli garantiscono che nessuna coppia di eventi in conflitto condivida lo stesso slot.

La difficoltà del problema deriva dalla combinazione dei due tipi di vincoli: se il grafo di conflitto ammette una colorazione con T colori (condizione di ammissibilità), esistono in generale molte soluzioni ammissibili, ma non tutte rispettano le preferenze allo stesso modo. La ricerca dell'ottimo richiede quindi di navigare uno spazio combinatorio in cui i vincoli hard restringono l'insieme delle soluzioni ammissibili e i vincoli soft ne differenziano la qualità.

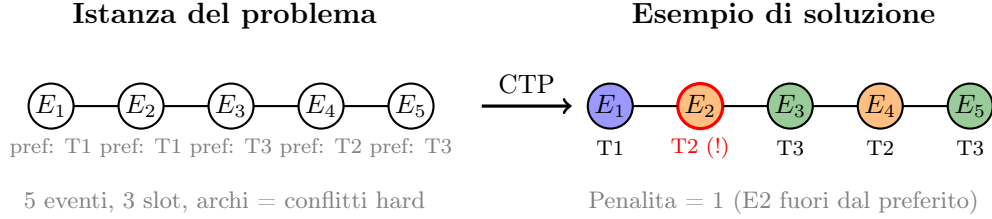


Figura 1: A sinistra: un'istanza illustrativa con 5 eventi (nodi) e archi che rappresentano i conflitti hard; sotto ogni evento è indicato lo slot preferito. A destra: un esempio di soluzione, in cui a ciascun evento è assegnato uno slot (colore diverso per ciascuno di T_1, T_2, T_3). L'evento E_2 è evidenziato in rosso perché assegnato a uno slot non preferito, generando una penalità totale pari a 1. La figura ha scopo unicamente illustrativo e non corrisponde all'istanza considerata nel progetto.

2 Istanza del Problema

Nel progetto è considerata una singola istanza del CTP composta da $n = 8$ eventi e $T = 4$ slot temporali disponibili.

I dati dell'istanza sono forniti nel file allegato alla presente traccia:

- **courses.txt**: contiene il numero di eventi n , il numero di slot T , la lista delle coppie di eventi in conflitto hard e, per ciascun evento, l'insieme degli slot preferiti.

Il valore ottimo della funzione obiettivo è noto ed è pari a $z^* = 1$: qualunque assegnazione ammissibile deve violare almeno una preferenza. Tale valore può essere utilizzato per verificare la correttezza delle soluzioni ottenute.

3 Protocollo Sperimentale

Il protocollo sperimentale varia in funzione della natura dell'algoritmo scelto.

Algoritmo deterministico (Constraint Programming)

Per il Constraint Programming è richiesta **una sola esecuzione**. Nella relazione finale devono essere riportati:

- il valore della funzione obiettivo ottenuto;
- l'assegnazione degli eventi agli slot (vettore x);
- la verifica dell'ammissibilità della soluzione (nessun conflitto hard violato);

- l'elenco delle preferenze violate;
- la strategia di branching e le tecniche di propagazione utilizzate.

Algoritmi stocastici (Local Search, Tabu Search, ACO)

Per gli algoritmi stocastici sono richieste **5 run indipendenti**, utilizzando seed diversi. Ogni run è governata dal seguente criterio di arresto:

- il raggiungimento del valore ottimo noto $z^* = 1$, oppure
- il raggiungimento di un numero massimo di **500 valutazioni** della funzione obiettivo.

Una *valutazione della funzione obiettivo* corrisponde al conteggio delle preferenze violate in una soluzione completa e ammissibile (che rispetta tutti i vincoli hard).

Nella relazione finale devono essere riportati:

- il valore della funzione obiettivo ottenuto in ciascuna run;
- l'assegnazione degli eventi agli slot della run migliore;
- la verifica dell'ammissibilità della soluzione;
- l'elenco delle preferenze violate della run migliore;
- la media dei valori finali sulle 5 run;
- un breve commento sul comportamento dell'algoritmo adottato.

4 Algoritmo da Implementare

Il candidato è invitato a scegliere e implementare uno dei seguenti algoritmi:

1. **Constraint Programming (CP)**, con formulazione del problema mediante variabili $x_i \in \{1, \dots, T\}$, vincoli di disuguaglianza per le coppie in conflitto e minimizzazione del numero di preferenze violate. Devono essere descritti la strategia di branching (scelta della variabile e del valore) e le tecniche di propagazione utilizzate.
2. **Local Search (LS)**, con soluzione iniziale generata casualmente (o costruita con una regola semplice, come l'assegnazione a uno degli slot preferiti quando possibile), vicinato basato sul riassegnamento dello slot di un singolo evento, e criterio di accettazione *best-improvement* oppure *first-improvement*, da specificare nella relazione. La gestione dei vincoli hard (restrizione del vicinato alle sole mosse ammissibili oppure penalizzazione delle infeasibility) deve essere dichiarata e motivata.
3. **Tabu Search (TS)**, con vicinato analogo a quello della Local Search, tabu list sulle mosse recenti (ad esempio, coppie evento-slot appena modificate), criterio di aspirazione e strategie di intensificazione/diversificazione a scelta del candidato. Tutti i parametri (dimensione della tabu list, durata, criterio di aspirazione) devono essere dichiarati e motivati nella relazione.
4. **Ant Colony Optimization (ACO)**, con costruzione incrementale della soluzione (ogni formica assegna gli eventi uno alla volta), matrice di feromone $\tau_{i,t}$ associata alle coppie evento-slot, informazione euristica $\eta_{i,t}$ basata sulle preferenze e sulla compatibilità con i vincoli hard, e regola di aggiornamento del feromone lasciata alla scelta del candidato. Tutti i parametri (α , β , ρ , numero di formiche) devono essere dichiarati e motivati nella relazione.

5 Note Finali al Progetto

La relazione finale deve essere redatta in $\text{\LaTeX}$, con una lunghezza minima di 4 pagine e massima di 12 pagine.

È fortemente sconsigliata l'inclusione di codice implementato, mentre è fortemente consigliato l'uso di pseudo-codice e diagrammi esplicativi.

La relazione deve illustrare in modo chiaro:

- l'algoritmo sviluppato e le sue caratteristiche principali;
- le motivazioni delle scelte progettuali effettuate;
- eventuali elementi di novità e originalità introdotti;
- un'analisi critica dei risultati ottenuti;
- considerazioni finali sul comportamento dell'algoritmo.

La valutazione finale si baserà principalmente su:

- correttezza dell'implementazione e rispetto dei vincoli;
- qualità dei risultati ottenuti;
- qualità espositiva della relazione;
- completezza e chiarezza dell'analisi.

Consegna: il progetto (codice sorgente) e la relazione finale (`.pdf` e `.tex`) devono essere inviati via email all'indirizzo `carolina.crespi@unict.it` entro le ore 14:00 del **12 ottobre 2026**. Il candidato è invitato a comunicare la scelta dell'algoritmo da sviluppare entro le ore 16:00 del **25 settembre 2026**.