Ottimizzazione dei Garanti accademici

Colli Simone[1] and Merenda Saverio Mattia[1]

[1] simone.colli@studenti.unipr.it
[2] saveriomattia.merenda@studenti.unipr.it

December 10, 2024

(MM:

- Modificare il nome del progetto su github e nel container docker da pd-project a ottimizzazione-garanti-accademici
- Scrivere le scritte dinamiche in italiano, ad esempio Figure 1, ecc

)

Abstract

Questo lavoro presenta l'analisi e l'implementazione di un sistema automatizzato per l'assegnazione dei garanti ai corsi universitari, in conformità ai requisiti ministeriali. L'obiettivo principale è garantire che ogni corso soddisfi i vincoli minimi di docenza, rispettando le regole di distribuzione tra diverse categorie di docenti e ottimizzando l'uso delle risorse disponibili.

Utilizzando la programmazione logica con Answer Set Programming (ASP) (Lifschitz, 2002), abbiamo modellato il problema attraverso fatti, regole e vincoli derivati dai dati ministeriali e universitari. Abbiamo implementato una serie di vincoli per rispettare i minimi richiesti di docenti per corso, evitando sovrapposizioni improprie tra gli incarichi dei docenti e considerando scenari realistici in cui un docente può assumere più ruoli parziali.

L'approccio è stato testato su un dataset reale contenente informazioni su corsi, SSD (Settori Scientifico Disciplinari) e docenti dell'Università degli Studi di Parma. I risultati dimostrano come il sistema possa trovare configurazioni ottimali che soddisfano i requisiti, massimizzando l'efficienza e mantenendo flessibilità nell'assegnazione dei docenti.

1 Introduzione

L'assegnazione dei garanti nei corsi universitari costituisce una questione fondamentale per la gestione ottimale delle risorse accademiche. Nell'ambito universitario, il *garante* è un docente responsabile di rappresentare e tutelare la qualità didattica di un corso, garantendo il rispetto dei requisiti disciplinari e istituzionali. I garanti possono appartenere a diverse categorie contrattuali: docenti a tempo indeterminato, docenti a tempo determinato e, in casi eccezionali, docenti a contratto.

La sfida principale consiste nel soddisfare i vincoli ministeriali relativi ai garanti, garantendo al contempo un'allocazione equilibrata e sostenibile delle risorse. Ogni corso deve essere supportato da un numero minimo di garanti, suddivisi tra le diverse fasce contrattuali, per assicurare un livello adeguato di competenza e rappresentatività. Inoltre, è indispensabile che almeno il 50% dei garanti afferisca al Settore Scientifico Disciplinare (SSD) caratterizzante del corso, al fine di garantire la coerenza tra l'offerta formativa e le competenze disciplinari.

Un'ulteriore complessità è rappresentata dall'impiego di docenti a contratto, il cui utilizzo deve essere limitato e subordinato alle sole situazioni in cui non sia possibile soddisfare i requisiti attraverso i docenti strutturati. La necessità di rispettare questi vincoli, combinata con la disponibilità limitata di personale e la necessità di bilanciare il carico di lavoro, rende questo problema una sfida organizzativa e computazionale significativa.

Un primo ostacolo affrontato in questo progetto riguarda la fase di pre-elaborazione dei dati forniti dall'università, descritta nella Sezione 2 di questo elaborato. I dati risultano disomogenei e incompleti, richiedendo una significativa pulizia e riorganizzazione prima di poter essere utilizzati efficacemente nel modello ASP. Questa fase ha richiesto lo sviluppo di strumenti dedicati per uniformare e validare i dati.

La Sezione 3 esplora il cuore del nostro approccio, ovvero la costruzione del modello ASP. Qui abbiamo implementato regole logiche per soddisfare i vincoli ministeriali e massimizzare l'efficienza dell'assegnazione dei garanti, tenendo conto delle limitazioni sulle fasce contrattuali e sull'impiego dei docenti a contratto.

Successivamente, nella Sezione 4, presentiamo i risultati ottenuti su un esempio ridotto, un test che ha permesso di validare il modello in un ambiente controllato e di analizzare la qualità delle soluzioni generate. Infine, la Sezione 5 descrive l'applicazione del modello su un dataset completo contenente tutti i corsi universitari, eseguendo un benchmark su larga scala per valutare la capacità del nostro approccio di risolvere problemi reali e complessi.

2 Preprocessing dei dati

2.1 Analisi preliminare

I dati analizzati, forniti dall'ufficio didattico dell'Università di Parma, erano organizzati in formato Excel con una struttura tabellare complessa, contenente numerosi campi. Per garantire un'analisi efficace e accurata, si è resa necessaria una fase di preprocessing, finalizzata a correggere eventuali incoerenze, migliorare la qualità complessiva del dataset e suddividerlo in sottoinsiemi più gestibili.

Il dataset fornito includeva informazioni relative ai docenti, ai ricercatori e alle coperture degli insegnamenti nei diversi corsi di laurea. Dopo un'attenta analisi preliminare, sono stati selezionati esclusivamente i campi considerati rilevanti per l'analisi. Nel file contenente i dati di docenti e ricercatori, sono state prese in considerazione solo le informazioni riguardanti la matricola, la fascia contrattuale e l'SSD. Per quanto riguarda il file sulle coperture, sono stati mantenuti i campi relativi alla matricola del docente, al codice del tipo di corso, al codice del di laurea e all'SSD.

Questa selezione dei dati è stata essenziale per ridurre la complessità e concentrarsi sugli elementi chiave necessari per costruire il modello ASP e verificare il rispetto dei vincoli ministeriali. Tutto questo è stato realizzato utilizzando diversi script in Python, supportati dai moduli `pandas`[1] e `openpyxl`[2], essenziali per la manipolazione e la gestione dei dataset.

La maggior parte delle operazioni di preprocessing ha riguardato il file contenente le coperture degli insegnamenti, articolandosi in diverse attività principali. In primo luogo, abbiamo gestito le righe contenenti una o più colonne vuote, eliminando quelle completamente prive di informazioni, in quanto non rilevanti per l'analisi. Le righe in cui mancavano i dati relativi al docente (i.e., matricola, nome e cognome) sono state invece isolate in un file separato. Questa scelta è stata necessaria per identificare e analizzare in modo specifico gli insegnamenti non ancora assegnati ad alcun docente.

Un ulteriore passo fondamentale ha riguardato la scomposizione del dataset originale, suddividendolo in sottoinsiemi più piccoli e gestibili. Questa operazione ha permesso di affrontare con maggiore precisione i singoli aspetti del problema, semplificando il successivo trattamento dei dati. Infine, sono stati identificati e gestiti in modo univoco e automatizzato i casi particolari, consentendo di creare un dataset uniforme e privo di anomalie, adatto all'analisi e all'implementazione del modello ASP.

Per migliorare ulteriormente la gestibilità dei dati, abbiamo proceduto con la creazione di sottoinsiemi minimali e distinti. Il primo passo è consistito nella separazione delle righe relative agli insegnamenti tenuti da docenti o ricercatori da quelle tenute dai docenti a contratto.

Questa suddivisione è stata effettuata utilizzando il file dei docenti fornito, sfruttando la colonna contenente la matricola come criterio discriminante. In particolare, le righe associate ai docenti a contratto sono state estratte e archiviate in un file separato, al fine di garantire un maggiore ordine e facilitare eventuali ispezioni manuali. Parallelamente, le righe relative ai docenti a tempo indeterminato e determinato sono state salvate anch'esse in un altro file, per gli stessi identici motivi.

Una volta completata la suddivisione, è stata apportata una modifica ai valori nella colonna relativa al codice del tipo di corso di laurea. In particolare, tutte le occorrenze del valore `L` sono state sostituite con `LT`, così da esplicitare i corsi di laurea triennale e migliorare la chiarezza dei dati. Successivamente, abbiamo proceduto con l'identificazione dei casi particolari utilizzando un controllo incrociato tra due colonne specifiche. Questo processo ha permesso di individuare e gestire in modo univoco i corsi di laurea che presentano requisiti particolari, come illustrato nella Tabella I. Per implementare questa logica, abbiamo adattato il processo di modifica dei dati al

[1] `https://pandas.pydata.org/`
[2] `https://openpyxl.readthedocs.io/en/stable/`

corso specifico da verificare. Ad esempio, per la Laurea Triennale in Servizio Sociale, abbiamo sostituito il codice del tipo di corso precedentemente identificato come `LT` con `LTSS`.

Corsi di studio	Docenza di riferimento (**minimo**)	Prof. a tempo indeterm. (**minimo**)	Ricercatori (**massimo**)	Docenti a contratto (**massimo**)
LT	9	5	4	2
LM	6	4	2	1
LMCU 5 anni	15	8	7	3
LMCU 6 anni	18	10	8	4
LT Servizio Sociale LT Scienze Motorie	5	3	2	1
LT Prof. sanitarie LT a orient. profess. LM Servizio Sociale LM Scienze Motorie	4	2	2	1
LM Infermieristica	3	1	2	1

Table I: Tabella che riassume i numeri di docenti per i corsi di studio.

2.2 Analisi post incontro con l'ufficio

(MM: fare passaggio) Venerdì 6 dicembre 2024 abbiamo svolto un incontro con l'ufficio incaricato della gestione dei dati accademici, durante il quale sono stati affrontati alcuni punti critici emersi nella fase di analisi preliminare. Questo confronto ci ha consentito di ottenere chiarimenti fondamentali e di affinare ulteriormente il processo di preprocessing dei dati sopra descritto.

Un argomento importante di cui abbiamo discusso, riguarda la corretta gestione di situazioni in cui la numerosità delle immatricolazioni ai corsi di laurea risultano superiori alla numerosità massima teorica.

Un aspetto cruciale emerso riguarda la definizione delle preferenze relative ai docenti incaricati del ruolo di garante. È stata delineata una chiara categorizzazione ed preferenza per la valutazione, includendo in ordine: i docenti con copertura al 100% per un singolo corso, docenti a contratto con copertura esclusivamente al 100% per un singolo corso, e docenti a contratto con copertura al 100% per due corsi distinti.

Relativamente alla scelta dei garanti è stata esposta una preferenza secondaria, che implica la copertura del ruolo di garante per un corso da parte del presidente di quel corso di laurea.

Questi chiarimenti e criteri definiti durante l'incontro sono stati integrati nel processo di preprocessing, consentendo di riorganizzare i dati in modo coerente con le preferenze e i vincoli discussi. Sono stati implementati controlli aggiuntivi per garantire la consistenza dei dati elaborati, migliorando così l'efficacia del modello ASP nella fase di assegnazione.

3 Answer Set Programming

In questa sezione approfondiamo l'organizzazione del progetto, evidenziando come sia stato strutturato per garantire chiarezza e modularità. Per facilitare l'ordine visivo e semplificare eventuali ispezioni manuali, il progetto è stato suddiviso in più file sorgenti, ciascuno dedicato a un aspetto specifico del dataset e delle regole.

Abbiamo separato i fatti e le regole in due file principali: uno dedicato ai docenti (Sezione 3.1) e un altro focalizzato sulle coperture (Sezione 3.2). Per maggiore praticità, è stato creato un file specifico per i docenti a contratto (Sezione 3.3) e un altro per i limiti ministeriali (Sezione 3.4).

Il file principale, il main, integra tutte le regole relative ai garanti, inclusi i vincoli e le priorità definite per l'ottimizzazione del sistema (Sezione 3.5).

È importante sottolineare che tutti questi file vengono generati dinamicamente dal programma di preprocessing scritto in Python. Questa fase, gestita dal file main del preprocessing, automatizza la creazione delle regole e dei fatti ASP in base ai dati forniti, garantendo così una configurazione personalizzata e facilmente aggiornabile.

3.1 Docenti

Il file `docenti.lp` contiene una rappresentazione strutturata delle informazioni relative ai docenti, organizzate in diverse sezioni per garantire chiarezza e modularità. Questo file serve come base per definire le caratteristiche dei docenti e il loro SSD.

La prima parte del file definisce gli SSD disponibili, ognuno rappresentato da una coppia di valori: la disciplina e il livello associato (e.g., INF/01, MAT/03). Gli SSD sono utilizzati per verificare la compatibilità tra i docenti e i corsi di laurea.

Successivamente, vengono dichiarate le fasce contrattuali dei docenti (`td` per tempo determinato, `ti` per tempo indeterminato). Queste fasce sono utilizzate per distinguere i docenti in base alla loro tipologia di contratto, (SC: Aggiungerei un "distinguendo i professori associati e ordinari dai ricercatori") aspetto rilevante per rispettare i requisiti ministeriali.

La sezione relativa ai docenti include l'elenco completo dei docenti, ciascuno identificato da una matricola unica. Per ogni docente, vengono indicate la fascia contrattuale e il settore scientifico disciplinare caratterizzante, che ne definisce il dominio di competenza. Queste informazioni sono formalizzate tramite il predicato `docente/4`, che associa la matricola del docente, la fascia, il settore disciplinare e il livello SSD.

Di seguito, riportiamo un esempio per illustrare la struttura dei dati contenuti nel file.

```
%  SEZIONE: SSD
ssd(inf, 1).

%  SEZIONE: FASCIE
fascia(td).
fascia(ti).

%  SEZIONE: Docenti
%  Rossi Mario (1234), SSD caratterizzante: inf/1
matricola_docente(1234).

%  SEZIONE: SSD caratterizzante dei docenti
%  Rossi Mario (1234), SSD caratterizzante: inf/1
docente(1234, td, inf, 1) :- matricola_docente(1234), fascia(td), ssd(inf, 1).
```

Listing 1: Esempio struttura dati di `docenti.lp`.

3.2 Coperture

Il file `coperture.lp` è fondamentale per rappresentare la struttura dei corsi universitari, le informazioni sui docenti assegnati e le caratteristiche associate a ciascun corso. Questo file contiene diverse sezioni organizzate per garantire chiarezza e modularità nella definizione delle informazioni.

La prima parte del file introduce i tipi di corso (`laurea/1`) e i relativi SSD utilizzati per caratterizzare i corsi e i docenti. Successivamente, vengono definiti i TAF (Tipologia Attività Formativa), che classificano ulteriormente gli insegnamenti.

Le informazioni sui corsi sono strutturate in due livelli: i codici identificativi di ciascun corso (`codice_corso/1`) e la loro descrizione dettagliata tramite il predicato `corso/4`. Quest'ultimo associa il codice del corso, il tipo di laurea, il SSD caratterizzante e il livello del settore disciplinare.

Una parte cruciale del file riguarda la relazione tra corsi e docenti, formalizzata tramite il predicato `cattedra/4`. Questo predicato associa un docente (identificato tramite la matricola) a un corso specifico, includendo il tipo di laurea e il TAF relativo.

Di seguito, un esempio rappresentativo del contenuto del file.

```
%  SEZIONE: Tipi di Corso
laurea(lt).
laurea(lm).

%  SEZIONE: TAF
taf(b).
taf(a).
taf(c).

%  SEZIONE: Corsi
%  INFORMATICA (3027)
codice_corso(3027).

%  SEZIONE: Informazioni Corsi
corso(3027, lt, inf, 1) :- codice_corso(3027), laurea(lt), ssd(inf, 1).

%  SEZIONE: Relazioni Corsi-Docenti
```

```
%  Corso: 3027, Docente: Rossi Mario
cattedra(3027, 1234, lt, b) :- codice_corso(3027),
                               matricola_docente(1234),
                               laurea(lt), taf(b).
```

Listing 2: Esempio struttura dati di `coperture.lp`.

3.3 Docenti a contratto

Il file `docenti_a_contratto.lp` contiene le informazioni relative ai docenti a contratto, una categoria di docenti utilizzata come risorsa di emergenza nei casi in cui non è possibile soddisfare i vincoli ministeriali con i docenti a tempo determinato o indeterminato. Questi docenti sono identificati tramite la fascia `c` e vengono impiegati come *jolly*, ovvero come soluzioni di riserva per garantire la copertura minima dei corsi universitari.

All'interno del file, ogni docente a contratto è identificato tramite la sua matricola ed è marcato con il predicato `jolly/1`. Questo predicato segnala che il docente può essere assegnato a qualsiasi corso, indipendentemente dall'SSD. I docenti a contratto non sono associati a uno specifico corso o SSD come i docenti a tempo determinato o indeterminato, ma possono essere impiegati in modo flessibile per *"tappare i buchi"* e assicurare che tutti i corsi soddisfino i requisiti minimi.

L'utilizzo dei docenti a contratto nel modello ASP è strettamente regolato. Essi sono inclusi come possibili garanti nei corsi, ma il loro utilizzo è penalizzato nel processo di ottimizzazione, al fine di privilegiare i docenti delle altre fasce. Questa scelta garantisce che i docenti a contratto siano usati solo come ultima risorsa, minimizzando il loro impatto sulla soluzione complessiva. Approfondiremo meglio questa questione nella Sezione 3.6.

Di seguito è riportato un esempio della struttura del file:

```
%  SEZIONE: FASCIE
fascia(c).

%  SEZIONE: Docenti
%  Rossi Mario (1234), docente a contratto
matricola_docente(1234).
jolly(1234).
```

Listing 3: Esempio struttura dati di `docenti_a_contratto.lp`.

3.4 Limiti ministeriali

Il file `ministeriale.lp` contiene i dati relativi ai requisiti ministeriali per ogni corso di laurea, con particolare riferimento al numero minimo di garanti richiesti per ciascun corso. Ogni corso ha associato un insieme di valori che determinano i vincoli di assegnazione dei docenti, inclusi i docenti a tempo indeterminato e tempo determinato, e il numero massimo di docenti a contratto.

Le regole `ministeriale/5` sono calcolate dinamicamente durante la fase di preprocessing. In particolare, questi valori sono determinati sulla base del numero di studenti iscritti a ciascun corso e grazie all'applicazione della formula della W, come illustrato nella Tabella II, che tiene conto delle specifiche necessità di copertura di ciascun corso. Nello specifico, la W viene calcolata nel seguente modo:

$$W = \frac{\text{Studenti iscritti al corso}}{\text{Massimo teorico di iscritti al corso}} - 1$$

Di seguito un esempio della struttura del file per il corso di Informatica, per il quale sono richiesti almeno 9 garanti, di cui 5 a tempo indeterminato, 4 a tempo determinato, e un massimo di 2 docenti a contratto.

```
%  SEZIONE: Garanti minimi per corso (codice_corso, minimo_complessivo,
%                                     docenti_ti, docenti_td,
%                                     max_docenti_contratto)
ministeriale(3027, 9, 5, 4, 2).
```

Listing 4: Esempio struttura dati di `ministeriale.lp`.

3.5 Vincoli

(MM: sistemare e aggiungere i vincoli) (MM: scrivere la roba dei pesi)

Nel modello proposto abbiamo introdotto una serie di vincoli deboli per avere maggiore flessibilità nella gestione di situazioni in cui non è possibile rispettare rigorosamente i requisiti ministeriali.

Qualifica	**# base**	**# effettivo**
Docenza di riferimento	9	$\lfloor 9 \times (1 + w) \rfloor$
Docenti a tempo indeterminato	5	$\lfloor 5 \times (1 + w) \rfloor$
Docenti non appartenenti all'ateneo	3	$\lfloor 3 \times (1 + w) \rfloor$

Table II: Formule per il calcolo del numero di docenti di riferimento in base alla numerosità degli studenti.

```
% Vincolo: il numero di docenti a tempo indeterminato per ogni corso deve
% essere almeno pari al minimo ministeriale richiesto.
% Se il numero e' inferiore al minimo, la soluzione viene scartata.
% ! weak constraint
:~ conta_docenti_indeterminato(Corso, Numero),
        ministeriale(Corso, Minimo, Minimo_ind, _, _),
        codice_corso(Corso),
        Numero < Minimo_ind.
        [Numero * 10@1]

% Vincolo: il numero di docenti a tempo determinato per ogni corso non deve
% essere superiore al massimo ministeriale richiesto.
% Se il numero e' superiore al massismo, la soluzione viene scartata.
% Penalizzazione per ricercatori che superano il limite consentito
% ! weak constraint
:~ conta_docenti_determinato(Corso, Numero),
        ministeriale(Corso, _, _, Massimo_det, _),
        Numero > Massimo_det,
        Penalita = Numero - Massimo_det.
        [Penalita * 50@1]
% Vincolo: il numero di docenti che afferiscono al SSD del corso deve rappresentare
    almeno il 50%
% del numero totale di docenti di riferimento.
% Se la condizione non e' soddisfatta, la soluzione viene scartata.
% ! weak constraint
:~ numero_docenti_che_afferiscono(Aff),
        numero_docenti_di_riferimento(Tot),
        2 * Aff <= Tot.
        [Tot - 4 * Aff@1]
```

Listing 5: Vincoli di flessibilità

```
% Vincolo debole per minimizzare il numero di garanti effettivi rispetto al minimo
    richiesto.
% Penalizza le soluzioni in cui il numero di garanti effettivi
% (calcolato da `garanti_per_corso`) non coincide con il numero minimo
% stabilito dai requisiti ministeriali per il corso.
:~ garanti_per_corso(Corso, Numero),
        ministeriale(Corso, Minimo, _, _, _),
        Numero != Minimo.
        [Numero * Minimo@1]
```

Listing 6: Minimizzazione dei garanti per corso di laurea.

3.6 Gestione delle priorità

La gestione delle priorità è stata un aspetto fondamentale nel nostro progetto, poiché miravamo a rispettare scrupolosamente i vincoli ministeriali, garantendo al contempo flessibilità per gestire situazioni particolari in cui tali vincoli non fossero pienamente soddisfacibili.

Per organizzare le priorità dei garanti, abbiamo deciso di basarci, in seguito ad un colloquio con l'ufficio didattico, principalmente sulla loro tipologia contrattuale. Il nostro obiettivo era quello di massimizzare l'impiego di docenti a tempo determinato (i.e., ricercatori) e a tempo indeterminato (i.e., professori associati e ordinari), prima di ricorrere ai docenti a contratto, che sono utilizzati solo quando strettamente necessario. Per ottimizzare la gestione, abbiamo preferito che i docenti assumessero il ruolo di garanti per un solo corso, evitando situazioni in cui un docente fosse impegnato come garante per più corsi contemporaneamente al 50%.

Abbiamo implementato queste priorità nel nostro modello con l'ausilio di regole di massimizzazione e minimizzazione, come mostrato di seguito:

```
% Docenti a tempo determinato (ricercatori)
#maximize { 50 : garante(_, _, _, td) }.

% Docenti a tempo indeterminato
#maximize { 40 : garante(_, _, _, ti) }.

% Docenti a contratto
#maximize { 32 : garante(_, _, _, c) }.

% Docenti con peso 10
#maximize { 25 : garante(_, _, Peso, _), Peso = 10 }.

% Minimizzare i docenti con peso 5
#minimize { 100, Docente : garante(Docente, _, Peso, _), Peso = 5 }.
```

Listing 7: Gestione delle priorità dei docenti.

Inoltre, abbiamo dato priorità ai docenti che insegnano corsi fondamentali per i corsi di laurea, distinguendo tra gli insegnamenti di base, identificati dal TAF A, e gli insegnamenti caratterizzanti, identificati dal TAF B.

```
% Garanti che hanno insegnamenti di base
#maximize { 25 : garante(Docente, Corso, _, _),
                 cattedra(Corso, Docente, _, TAF),
                 TAF = a}.

% Garanti che hanno insegnamenti caratterizzanti
#maximize { 18 : garante(Docente, Corso, _, _),
                 cattedra(Corso, Docente, _, TAF),
                 TAF = b}.

% Garanti che hanno insegnamenti affini
#maximize { 10 : garante(Docente, Corso, _, _),
                 cattedra(Corso, Docente, _, TAF),
                 TAF = c}.
```

Listing 8: Gestione delle priorità del TAF.

Per ottimizzare l'assegnazione dei garanti in base alla loro competenza, sono stati privilegiati i docenti il cui l'SSD corrisponde a quello caratterizzante del corso. In questo modo, ci siamo assicurati che i garanti siano adeguatamente allineati con le competenze richieste dai corsi.

```
% Ottimizzare i garanti con SSD caratterizzante
#maximize { 20 : garante(Docente, Corso, _, _),
                 docente(Docente, _, SettoreSSD, _),
                 corso(Corso, _, SettoreSSD, _) }.
```

Listing 9: Preferenza dei garanti con macrosettore coerente a quello del corso.

Infine, per mantenere l'equilibrio tra la qualità e l'efficacia del sistema, abbiamo minimizzato il numero di garanti assegnati a ciascun corso, penalizzando le situazioni in cui il numero di garanti supera il minimo richiesto. Questo vincolo ci ha permesso di ottimizzare le risorse, evitando l'assegnazione di più docenti di quanto fosse necessario.

```
% Minimizzo il numero di garanti per ogni corso
#minimize { Penalita : garanti_per_corso(Corso, Numero),
                       Penalita = Numero * 10 }.
```

Listing 10: Minimizzazione dei garanti per corso di laurea.

3.7 Ottimizzazioni

4 Esempio giocattolo

(MM: magari cambiare nome della sezione in "Validazione della soluzione proposta", lo rende più professionale e accattivante)

Per validare il modello sviluppato e testare l'efficacia dell'approccio proposto, abbiamo costruito un esempio giocattolo sfruttando il corso di laurea triennale di informatica ed il corso di laurea magistrale in Scienze Informatiche.

Il dataset generato ci ha accompagnato nel processo di testing del codice ASP proposto; permettendoci di verificare se fosse in grado di gestire correttamente la generazione dei termini, i vincoli e le preferenze specificate.

I risultati ottenuti sono stati utilizzati per ottimizzare ulteriormente il codice, permettendoci di identificare eventuali aree di miglioramento relative alla gestione dei vincoli e all'efficienza della generazione dei termini.

(MM: spiegare i comandi utilizzati per generare il dataset (il comando python))
(MM: specificare il comando utilizzato per lanciare l'analisi con clingo)
(MM: stampare l'output andando a commentare il risultato)

5 Valutazione sperimentale

5.1 Dataset piccolo (dipartimento SMFI)

5.2 Dataset grande (tutti i dipartimenti)

6 Conclusioni

Acknowledgments
Questo progetto è stato realizzato nel corso di Programmazione Dichiarativa (a.a. 2024-25), presso l'Università degli Studi di Parma.

References

Lifschitz, Vladimir (2002). "Answer set programming and plan generation", *Artificial Intelligence*, Vol. 138 No. 1. Knowledge Representation and Logic Programming, pp. 39–54. ISSN: 0004-3702. DOI: https://doi.org/10.1016/S0004-3702(02)00186-8. **available at**: https://www.sciencedirect.com/science/article/pii/S0004370202001868.