Università degli Studi di Padova

Università degli Studi di Padova

Dipartimento di Ingegneria dell'Informazione

Corso di Laurea Triennale in
Ingegneria Informatica

Un approccio MIP per la risoluzione del problema del Minimal Seed Set

Relatore:
Prof. Domenico Salvagnin

Laureando:
Alberto Basaglia
2008803

Anno Accademico 2022/2023

Abstract

Questa tesi studia un interessante problema legato al mondo della biologia. Il problema del *Minimal Seed Set* è cosí definito: data una descrizione delle molecole che compongono un organismo e delle reazioni chimiche che possono avvenire al suo interno, determinare il più piccolo sottoinsieme di molecole che, attraverso le reazioni, possa essere combinato fino a raggiungere l'intero insieme.

Viene mostrato come una formulazione MIP del problema non solo è possibile, ma scala bene anche per i sistemi più complessi e permette di risolvere tutte le istanze prese in esame in tempi brevi.

Saranno presentate due possibili formulazioni per il problema e ne verranno confrontati i risultati.

Indice

Elenco delle tabelle

Elenco delle figure

Capitolo 1

Introduzione

In un paper del 2008 di *Borenstein et al* che tratta il problema del *Minimal Seed Set* viene riportato che una strategia basata sulla programmazione lineare intera non è scalabile ad istanze reali del problema. Si ricorre quindi ad un'approssimazione che possa essere risolta in tempi ragionevoli.[2]

Nel 2011 *Gefen e Brafman* provano un approccio basato sul planning che risulta però inconclusivo. Una soluzione ottima è trovata grazie ad una variante dell'algoritmo A* che permette loro la risoluzione di tutte le istanze prese in esame.[7]

In nessuna delle ricerche precedenti compare la modellazione lineare intera del problema utilizzata, si potrá quindi cercare di scrivere un modello più efficiente e scalabile di quello utilizzato dagli autori.

Inoltre, da quando le due ricerche sono state pubblicate, i risolutori disponibili sono migliorati, rendendo possibile la risoluzione di problemi più grandi e complessi. [9] [1] [11]

L'obiettivo di questa tesi è dunque quello di ricavare una modellazione del problema e risolverlo con moderni risolutori allo stato dell'arte. Verranno poi discusse le istanze reali che sono state risolte e le prestazioni raggiunte.

1.1 Programmazione Lineare Intera

Un problema di programmazione lineare consiste nel trovare il minimo (o il massimo) valore di una funzione lineare soggetta ad un numero finito di vincoli lineari.

Si parla di programmazione lineare intera quando, in aggiunta alla definizione precedente, è permessa anche l'aggiunta del vincolo di interezza ad un sottoinsieme delle variabili del problema.

Viene presentata ora la formulazione generica di un problema di programmazione lineare intera. 1.1

$$\begin{array}{lll} \min & f(x) & \\ \text{t.c.} & a_i x \sim b_i & i = 1, \ldots, m \\ & l_j \leq x_j \leq u_j & j = 1, \ldots, n = M \\ & x_j \in \mathbb{Z} & \forall j \in J \subseteq N = \{1, \ldots, n\} \end{array} \tag{1.1}$$

È importante notare che la programmazione lineare puó essere vista come un caso particolare della programmazione lineare intera. Infatti, prendendo $J = \emptyset$, il vincolo di interezza non verrebbe applicato a nessuna delle variabili prese in esame. Si parlerebbe invece di programmazione lineare intera pura nel caso di $J = N$. Negli altri casi invece si parla sempre di *programmazione lineare intera mista*, a cui ci si riferisce con *MIP* (o a volte *MILP*).

Nel corso del capitolo verranno trattati problemi di minimo, nel caso di problemi di massimo valgono gli stessi principi con dei piccoli accorgimenti (banalmente un cambio di verso delle disequazioni).

1.2 Principali tecniche di risoluzione

1.2.1 Concetti generali

Restrizione

Dato un problema di ottimizzazione P, creo un altro problema P' aggiungendo dei vincoli. Si avrá sicuramente:

$$F(P') \subseteq F(P)$$

ovvero l'insieme delle soluzioni del problema con vincoli aggiunti sará un sottoinsieme di quello del problema originale.

Chiaramente questo (se usato senza una particolare accortezza) non è utile: Se viene trovata una soluzione ottima essa lo sará soltanto per la restrizione e

non globalmente. Se non si riesce a trovare nessuna soluzione (quindi $F(P') = \emptyset$) nulla potrá essere detto per P.

Quello che si puó fare è prendere un insieme di restrizioni che copra tutto l'insieme di partenza:

$$P_1, \cdots, P_k | \bigcup_{i=1}^{k} F(P_i) = P$$

In questo caso si parlerá di restrizioni *esaustive*.

Avendo un insieme di restrizioni che copre tutto l'insieme di partenza, nel caso in cui tutte le restrizioni fossero impossibili, anche il problema di partenza sarebbe impossibile. Se più restrizioni sono risolte all'ottimo, la soluzione ottima del problema di partenza è la migliore di quelle trovate (la più piccola nel caso di problemi di minimo, la più grande nei problemi di massimo).

Rilassamento

Un rilassamento R di un problema di ottimizzazione è un altro problema ottenuto da P

- Togliendo alcuni vincoli: $F(P) \subseteq F(R)$.
- Sostituendo la funzione obiettivo $f(x)$ con una sua approssimazione inferiore $g(x)$: $g(x) \leq f(x)\ \forall x \in F(P)$

Importante è notare che la funzione obiettivo del rilassamento deve approssimare inferiormente la funzione del problema soltanto nella regione ammissibile, $F(P)$.

Un rilassamento ha un comportamento complementare a quello delle restrizioni in questo modo descritto:

1. $F(R) = \emptyset \implies F(P) = \emptyset$: se non viene trovata nessuna soluzione ammissibile nel rilassamento, sicuramente non ne sarebbe stata trovata una neanche nel problema originale.
2. se $x^\star$ è soluzione ottima di R allora $x^\star$ è un **lower bound** di P: se il rilassamento (il cui spazio delle soluzioni comprende quello del problema originale) ha come soluzione ottima $x^\star$, andando a ottimizzare il problema originale si potrá sperare di fare meglio; in questo modo si ottiene un lower bound.

3.
$$\begin{cases} x^\star \text{ è soluzione ottima di } R \\ x^\star \in F(P) \\ g(x^\star) = f(x^\star) \end{cases} \implies x^\star \text{ è soluzione ottima di } P$$

Dopo aver trovato (e risolto) un rilassamento del problema di ottimizzazione, ci si troverá sicuramente in uno dei 3 casi descritti precedentemente.

I casi 1 o 3 forniscono informazioni importanti sul problema di partenza P e in entrambe le situazioni la mia elaborazione è finita: è stato trovato (e dimostrato) che il problema è impossibile nel primo caso oppure viene trovata una soluzione (e la dimostrazione che essa è tale) nel terzo caso.

Si puó dire che questi casi "dipendano dalla fortuna", diverso invece è il comportamento per il secondo caso: esso ci fornisce sempre un *lower bound*.

Questa proprietá sará fondamentale per l'algoritmo branch&bound.

1.2.2 Branch & Bound

L'algoritmo di branch&bound è una tecnica di tipo *divide-and-conquer* per risolvere problemi di ottimizzazione lineari. La tecnica si basa sul cercare di visitare lo spazio di ricerca in maniera intelligente, eliminando le parti in cui non puó essere presente nessuna soluzione utile: ora segue una breve spiegazione del funzionamento dell'algoritmo.

Dato F l'insieme delle soluzioni di un problema P, $c : F \to \mathbb{R}$ la sua funzione obiettivo e $\bar{x}$ una sua soluzione (determinata ad esempio euristicamente). Sia $z = c(\bar{x})$, il costo della soluzione nota, *incumbent*. Questo valore verrá poi utile per scartare alcuni rami dall'albero di ricerca.

Per prima cosa viene effettuata la fase di *bounding*: il problema P viene rilassato (ad esempio, nel caso di un problema di ottimizzazione lineare intera, potrebbe essere preso il suo rilassamento lineare) e poi risolto. Prendendo, ad esempio, F come insieme delle soluzioni di P, P' avrá come soluzione $G \supseteq F$. Se la soluzione ottima trovata appartiene anche ad F oppure coincide con z, si ha finito. Negli altri casi invece si passa alla dase del *branching*.

Si genera quindi una separazione $F^\star$ di F, ovvero un insieme finito di restrizioni che coincidono con l'insieme di partenza.

$$\bigcup_{F_i \in F^\star} F_i = F$$

Non è detto, anche se molto spesso è questo il caso, che $F^\star$ sia una partizione di F. I "figli" di F vengono ora aggiunti alla coda di problemi da processare, $\mathcal{Q}$.

Si estrae ora un problema dalla coda $\mathcal{Q}$ e se ne risolve un suo rilassamento, si ricadrá quindi sicuramente in una delle seguenti situazioni:

1. Se non viene trovata alcuna soluzione (il problema è impossibile) il problema, e tutti i suoi possibili figli, vengono scartati. In questo caso si parla di *pruning by infeasibility.*

2. Se viene trovata una soluzione *ammissibile* migliore dell'incumbent, esso viene aggiornato con la soluzione appena trovata.

3. Se la soluzione trovata è peggiore dell'incumbent, si puó scartare il sottoproblema. È possibile farlo perché, nel caso migliore in cui la soluzione del rilassamento fosse anche soluzione del problema, comunque non darebbe risultati migliori di quelli già ottenuti. In questo caso si parla di *pruning by optimality.*

4. Se non è stato possibile scartare il sottoproblema sará necessario fare branching e aggiungere i figli del sottoproblema corrente alla coda $\mathcal{Q}$.

Si continuano poi ad eseguire questi passaggi fino a quando la coda $\mathcal{Q}$ si svuota. Al termine dell'esecuzione, la soluzione al problema è l'*incumbent* corrente.

1.2.3 Branch & Cut

Per migliorare le prestazioni dell'algoritmo precedente, esso puó essere combinato con la tecnica dei piani di taglio. L'idea è quella di aggiungere delle disequazioni ad ogni nodo dell'albero decisionale in modo da "rafforzare" la formulazione del sottoproblema.

Per poter applicare questa tecnica in modo efficace, sará necessario avere dei metodi efficienti per risolvere il problema separazione.

Esso consiste nel, data una soluzione $x^\star$ che si vuole scartare, derivare una disuguaglianza valida del tipo $\alpha^T x \leq \alpha_0$ che "tagli" fuori $x^\star$ dal problema. Dovrá quindi valere $\alpha^T x^\star > \alpha_0$.

Cosí facendo si avrá una formulazione rafforzata del problema di partenza, da cui è stata però rimossa $x^\star$, che non potrá quindi più essere trovata come soluzione.

1.2.4 Gap

Le tecniche di Branch&Bound e Branch&Cut permettono inoltre di calcolare il gap della risoluzione. Questo valore, chiamato *gap assoluto*, è calcolato come la differenza tra la miglior soluzione trovata (z) e il miglior rilassamento. Nella maggior parte dei casi si utilizza un *gap relativo* ottenuto dividendo il valore calcolato in precedenza per la miglior soluzione.

La risoluzione, una volta portata a termine, avrá gap 0. Se la risoluzione fosse interrotta, ad esempio per time limit, sará possibile comunque derivare da essa delle informazioni utili attraverso la miglior soluzione trovata e il gap rimanente.

Il gap ci fornisce un'indicazione sul progresso dell'algoritmo di risoluzione. Ad esempio, un valore grande del gap sta ad indicare che il processo di ottimizzazione è ancora lontano dall'arrivare alla soluzione ottima.

Capitolo 2

Il problema

Il problema del *Minimal Seed Set* viene definito [2] come segue:

Sia C un insieme di molecole associato ad un determinato organismo. Una reazione biochimica è una coppia ordinata di insiemi, $r = (X, Y)$. $X \subseteq C$ viene chiamato *substrate* ed è l'insieme delle molecole che sono necessarie per soddisfare la reazione. $Y \subseteq C$ è l'insieme dei prodotti della reazione e contiene tutte le molecole risultanti dal completamento di una reazione. Spesso le reazioni sono bidirezionali, in questo caso verranno quindi rappresentate da r_1 ed r_2 tali che $r_1 = (X, Y)$, $r_2 = (Y, X)$. In questo modo una reazione reversibile viene modellata come due reazioni separate e opposte.

Il *metabolic network* R di un organismo è definito quindi come l'insieme delle reazioni che possono avvenire al suo interno. Viene utilizzato C per rappresentare tutte le molecole che appaiono nelle reazioni.

Un insieme di molecole C è detto raggiungibile da un suo sottoinsieme S se esiste una sequenza di reazioni che, dopo essere stata applicata a S, ha prodotto tutti i componenti di C.

Un sottoinsieme di C è definito *seed set* se ci permettere di raggiungere l'insieme di partenza attraverso le reazioni.

Infine, il *Minimal Seed Set* è il più piccolo seed set di C. Esso esiste sempre (nel caso "peggiore" in cui nessuna reazione puó essere applicata corrisponde a C) e potrebbe non essere unico.

Ogni molecola dovrá quindi far parte del sottoinsieme di partenza oppure venir generata da almeno una reazione. È importante notare che in questo modello di organismo, il completamento di una reazione non rimuove le molecole di partenza.

La naturale rappresentazione di questo problema è attraverso l'utilizzo di un grafo bipartito. I due tipi di vertici sono la molecola e la reazione.

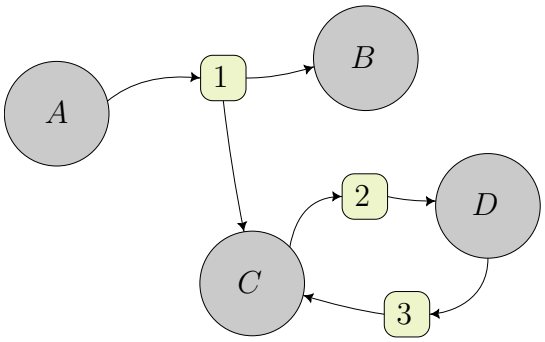


Figura 2.1: Esempio di grafo relativo ad un organismo

Gli archi del grafo sono orientati e hanno il seguente significato: un arco da una reazione ad una molecola indica che essa viene prodotta dalla reazione; un arco da una molecola ad una reazione significa che la molecola fa parte dell'insieme di reagenti necessari per il completamento di essa.

In figura 2.1 un esempio di rappresentazione di un organismo. L'insieme delle molecole che compongono l'organismo è $\{A, B, C, D\}$ e le reazioni sono reazioni $A \rightarrow B + C$, $C \leftrightarrow D$. Come si puó vedere, la seconda reazione, reversibile, è stata modellata come due reazioni separate. In questo caso, il *minimal seed set* sarebbe composto dalla molecola A: essa non viene prodotta da nessuna reazione e deve di conseguenza essere aggiunta all'insieme di partenza. Attraverso la reazione 1, a partire da A, vengono prodotte le molecole B e C. Quest'ultima, grazie alla reazione 2, ci permette di ottenere anche la molecola D.

Capitolo 3

Modellazione MIP

Vengono presentate due possibili formulazioni lineari intere per la risoluzione del problema. L'idea alla base di entrambi i modelli è quella di avere variabili che rappresentino l'appartenenza o meno di una molecola al *Minimal Seed Set*. È necessario anche aggiungere un insieme di variabili che esprimano il soddisfacimento (ovvero la presenza di tutti i reagenti richiesti) delle reazioni. Si dovrà poi fare in modo che le molecole prodotte diventino disponibili per completare altre reazioni.

È inoltre necessario tenere conto della possibile presenza di cicli all'interno del grafo per risolvere correttamente il problema. Per fare ciò si deve modellare, oltre alla disponiblitá o no di una molecola, anche l'istante temporale in cui essa diventa utilizzabile dalle reazioni. Se questo non venisse fatto sarebbe possibile avere, ad esempio, due molecole che si producono a vicenda senza che nessuna di esse venga aggiunta al *Minimal Seed Set*. Nello specifico, se un organismo formato soltanto dalle molecole A e B avesse solamente una reazione reversibile, $A \leftrightarrow B$, una molecola tra A e B dovrá essere per forza aggiunta al Minimal Seed Set.

È quindi richiesto un vincolo che imponga alle reazioni di poter essere completate solamente dopo che tutte le molecole richieste sono diventate disponibili, spezzando il ciclo di dipendenza e risolvendo il problema.

La principale differenza tra i due modelli di seguito descritti consiste nella tecnica utilizzata per rappresentare la successione degli eventi all'interno dell'organismo.

3.1 Prima formulazione

Utilizzo come insiemi

1. I: insieme delle molecole.
2. J: insieme delle reazioni.

come parametri

1. $r_{ij} \in \{0,1\}, i \in I, j \in J$: 1 se la reazione j richiede la molecola i; 0 altrimenti.
2. $p_{ij} \in \{0,1\}, i \in I, j \in J$: 1 se la reazione j produce la molecola i; 0 altrimenti.

e come variabili

1. $x_i \in \{0,1\}, i \in I$: 1 se la molecola i fa parte del *Minimal Seed Set*.
2. $u_{ij} \in \{0,1\}, i \in I, j \in J$: 1 se la reazione j è utilizzata per produrre i.
3. $t_i \in \mathbb{Z}^+, i \in I$: istante di prima disponiblitá di una molecola.

Visto che l'obiettivo è quello di trovare il sottoinsieme più piccolo, l'obiettivo del modello sará quindi quello di minimizzare

$$\sum_{i}^{I} x_i \tag{3.1}$$

Ogni molecola dovrá essere prodotta da almeno una reazione oppure essere stata aggiunta al minimal seed set

$$x_i + \sum_{j}^{\mathrm{p}(i)} u_{ij} \geq 1 \quad \forall i \in I \tag{3.2}$$

Bisogna poi fare in modo che le molecole prodotte da una reazione diventino disponibili nell'istante successivo al completamento di essa

$$\begin{gathered} t_a + 1 \leq t_b + M(x_a) + M(1 - u_{bj}) \\ \forall a \in I \quad \forall j \in J \text{ t.c. } p_{aj} = 1 \quad \forall b \in I \text{ t.c. } r_{bj} = 1 \end{gathered} \tag{3.3}$$

L'interpretazione del vincolo è la seguente: per ogni molecola a, per ogni reazione j che produce a, per ogni molecola b richiesta dalla reazione j, l'istante di disponibilitá di b deve essere successivo a quello di a. Il vincolo puó essere "disattivato" in due modi:

1. $x_a = 1$: la molecola di partenza è nel minimal seed set e quindi automaticamente presente in ogni istante.

2. $u_{bj} = 0$: la reazione non viene utilizzata e quindi nessun vincolo deve essere rispettato.

Infine è presente un vincolo per mantenere tutti gli istanti di tempo minori di M

$$t_i \leq M \quad \forall i \in I \tag{3.4}$$

Il modello completo è quindi

$$\begin{aligned}
\min \quad & \sum_{i}^{I} x_i & & \\
\text{t.c.} \quad & x_i + \sum_{j}^{\mathrm{p}(i)} u_{ij} \geq 1 & & \forall i \in I \\
& t_a + 1 \leq t_b + M(x_a) + M(1 - u_{bj}) & & \forall a \in I \\
& & & \forall j \in J \text{ t.c. } p_{aj} = 1 \\
& & & \forall b \in I \text{ t.c. } r_{bj} = 1 \\
& t_i \leq M & & \forall i \in I
\end{aligned} \tag{3.5}$$

3.2 Seconda formulazione

A differenza della precedente formulazione, viene aggiunto un nuovo insieme T per rappresentare gli istanti di tempo in cui possono avvenire le reazioni.

Agli insiemi verrá aggiunto quindi

1. $T = \{0, 1, 2, \ldots, M\}$: insieme degli istanti di tempo.

i parametri sono i medesimi della formulazione precedente. Le variabili diventano invece

1. $x_i \in \{0, 1\}, i \in I$: 1 se la molecola i fa parte del *Minimal Seed Set* (come nella prima formulazione).

2. $d_{it} \in \{0, 1\}, i \in I, t \in T$: 1 se la molecola i è disponibile all'istante t.

3. $s_{jt} \in \{0,1\}, j \in J, t \in T$: 1 se la reazione j è soddisfatta all'istante t.

Come in precedenza, l'obiettivo è quello di trovare il sottoinsieme più piccolo, si vuole quindi minimizzare

$$\sum_{i}^{I} x_i \tag{3.6}$$

La disponibilitá di una molecola in un determinato istante di tempo è condizionata dalla sua disponibilitá all'istante di tempo precedente. Inoltre, nel caso in cui ci siano reazioni che lo producono, esso puó diventare disponibile in seguito al soddisfacimento di una reazione all'istante antecedente

$$d_{it} \leq d_{i(t-1)} + \sum_{j}^{J} p_{ij} s_{j(t-1)} \quad \forall i \in I \quad \forall t \in T \setminus \{0\} \tag{3.7}$$

Per poter essere soddisfatta, una reazione deve rispettare la disponibilitá di tutti gli elementi necessari

$$d_{it} \geq s_{jt} \quad \forall i \in I \quad \forall j \in J \quad \forall t \in T \quad \text{t.c. } r_{ij} = 1 \tag{3.8}$$

Le molecole che fanno parte del *Minimal Seed Set* devono essere disponibili dal primo istante. Viene aggiunto quindi un vincolo per legare la disponibilitá di ogni elemento all'istante 0 alla sua appartenenza o meno al set minimale

$$d_{i0} = x_i \quad \forall i \in I \tag{3.9}$$

L'obiettivo del modello è quello di trovare un set che ci permetta di raggiungere l'intero insieme delle molecole. Viene quindi aggiunto un vincolo che, per essere soddisfatto, richiede la disponibilitá di ogni elemento all'istante finale

$$d_{iM} = 1 \quad \forall i \in I \tag{3.10}$$

Il modello completo MIP risulta quindi essere:

$$
\begin{aligned}
\min \quad & \sum_{i}^{I} x_i \\
\text{t.c.} \quad & d_{it} \leq d_{i(t-1)} + \sum_{j}^{J} p_{ij} s_{j(t-1)} && \forall i \in I \quad \forall t \in T \setminus \{0\} \\
& d_{it} \geq s_{jt} && \forall i \in I \quad \forall j \in J \quad \forall t \in T \quad \text{t.c. } r_{ij} = 1 \\
& d_{i0} = x_i && \forall i \in I \\
& d_{iM} = 1 && \forall i \in I
\end{aligned}
\tag{3.11}
$$

3.3 Scelta dei parametri M e T

È stato mostrato come entrambi i modelli abbiano dei parametri relativi agli istanti di tempo in qualche modo simili, anche se di natura diversa.

Il primo modello utilizza dei vincoli *BigM* per mantenere un ordine temporale nel completamento delle reazioni. Il secondo modello utilizza invece una dimensione aggiuntiva rappresentata dall'insieme degli istanti di tempo T.

Nonostante i modelli abbiano un approccio diverso, essi ottengono lo stesso comportamento a livello temporale ponendo $|T| = M + 2$.

Che valore assegnare ora ad M? Bisogna fare in modo che tutte le reazioni "abbiano il tempo" di essere completate per riuscire a trovare una soluzione ottima al problema. Un approccio potrebbe essere quello di prendere un valore più alto della somma del numero di reazioni e del numero di molecole, in modo da assicurarsi che ci siano abbastanza istanti di tempo anche nel caso peggiore.

Si pensi ora ad un modello in cui le reazioni sono in numero minore delle molecole. Il caso peggiore (quello che richiede il maggior numero di istanti di tempo) si verifica se per ogni t puó avvenire solamente una reazione, andando cosí a richiedere $|J|$ istanti. Prendendo dunque $M = |J|$ la soluzione trovata sará ottima.

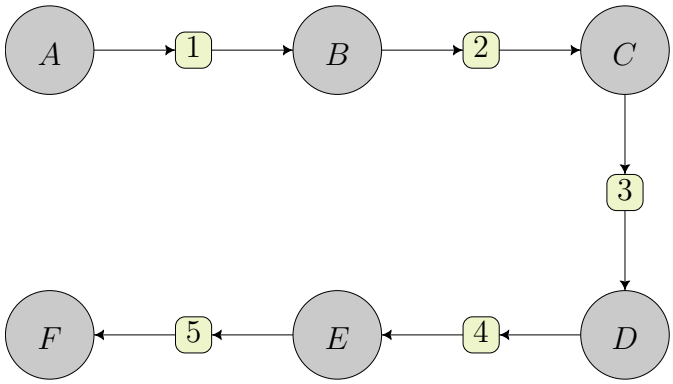


Figura 3.1: Catena di reazioni

Ad esempio nel caso in figura 3.1, il modello ha bisogno di almeno 5 istanti temporali per trovare la soluzione ottima con un solo elemento (in questo caso A) nel minimal seed set. Se fossero disponibili meno istanti di tempo verrebbe trovata una soluzione in cui un altro elemento è stato aggiunto (e quindi non la soluzione ottima cercata).

Se ci si trovasse invece in un caso in cui le reazioni fossero in numero maggiore delle molecole dell'organismo, potremmo prendere $M = |I|$. Infatti, anche nel caso peggiore in cui ogni ad ogni istante di tempo viene prodotta solamente una nuova molecola, si potrá essere sicuri di avere abbastanza istanti per valutare tutti i percorsi possibili.

Si conclude quindi dicendo che una buona scelta è assegnare

$$M = \min\{|I|, |J|\} \tag{3.12}$$

Si vedrá poi che, negli organismi presi in esame, le soluzioni ottime hanno bisogno di un numero molto minore di istanti temporali per coprire tutto il set.

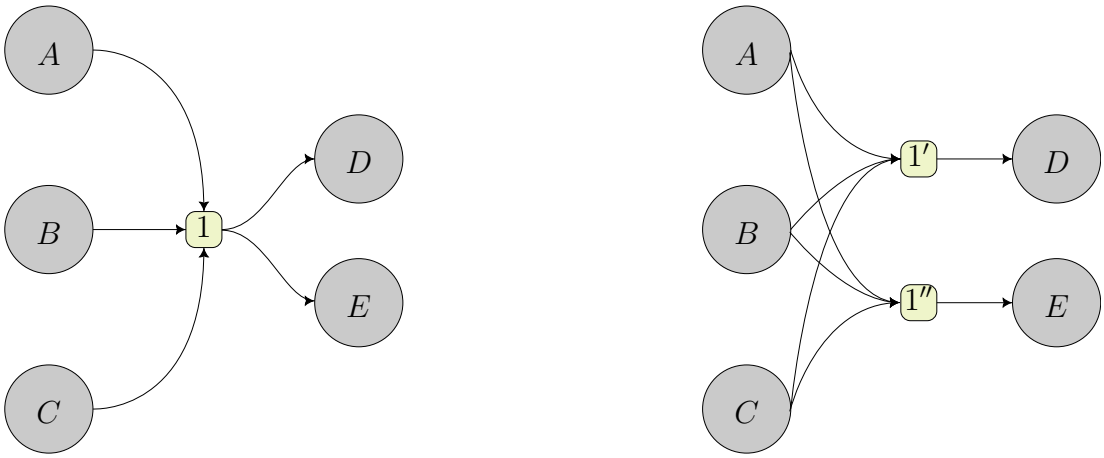
3.4 Preprocessing

In generale, in un determinato organismo, sono presenti reazioni che producono più di un elemento. Un approccio per provare a snellire il lavoro del risolutore è quello di separare le reazioni con più di un prodotto in un insieme di reazioni, con gli stessi reagenti richiesti, ma con solamente un elemento risultante. Un esempio è visibile in figura 3.2.

Oltre a questo, nel problema di partenza, potrebbero essere presenti reazioni duplicate: un passaggio di *preprocessing* le elimina per ridurre il numero di variabili del problema.

In aggiunta un altro passaggio controlla la presenza, ed in caso positivo effettua la rimozione, delle reazioni *dominate*. Con reazione dominata si intende una reazione che richiede lo stesso insieme di reagenti di un'altra reazione ma produce solo un sottoinsieme delle molecole. Naturalmente ha senso rimuovere le reazioni dominate solamente nel caso in cui non siano state divise con il passaggio di *pre processing* precedente (in cui una reazione con più di un prodotto viene divisa in più reazioni con un prodotto solo) visto che gli insiemi di molecole prodotte saranno sempre composti da solamente un elemento.

L'utilizzo di queste strategie e il loro impatto sulla risoluzione del problema verranno valutati nel capitolo seguente.



(a) Prima (b) Dopo

Figura 3.2: Esempio separazione reazioni

Capitolo 4

Istanze reali

In questo capitolo verranno presentate le procedure e le istanze utilizzate per collezionare dei risultati empirici.

4.1 Strumenti utilizzati

4.1.1 Database KEGG

Il modello è stato poi provato su delle istanze reali ottenute dal database KEGG [10]. KEGG (Kyoto Encyclopedia of Genes and Genomes) è un database bioinformatico utilizzato nel mondo della ricerca. Esso contiene informazioni sulle sequenze genetiche, le funzioni biologiche e le interazioni molecolari di una vasta gamma di organismi.

Il database contiene una collezione di "mappe metaboliche", ovvero rappresentazioni grafiche dei processi chimici che avvengono all'interno degli organismi. Queste mappe illustrano le interconnessioni tra le molecole e le reazioni chimiche che le riguardano.

Per utilizzare gli stessi dati utilizzati da *Gefen e Brafman* nel loro paper, è stato necessario estrarli dai file *PDDL* rilasciati direttamente degli autori in un repository GitHub.[6]

4.1.2 Macchine utilizzate

Tutti i test sono stati effettuati su macchine con processore *Intel Xeon CPU E3-1220 V2 @ 3.10GHz* e *16 GB* di ram. Come sistema operativo è stato utilizzato *Fedora 37* con kernel `6.1.18-200.fc37.x86_64`.

Per garantire la coerenza dei risultati, il risolutore è stato sempre l'unico processo in esecuzione sulla macchina.[1].

4.1.3 Generazione del modello

I modelli che verranno poi passati al risolutore sono stati generati con un programma scritto nel linguaggio Rust[12], utilizzando un modulo[3] per esportare file di tipo *.lp* compatibili con *CPLEX*. Il codice sorgente del programma è consultabile all'indirizzo https://github.com/albertobasaglia/minseedset-tools.

4.1.4 CPLEX

CPLEX è un software commerciale di IBM utilizzato per risolvere problemi di ottimizzazione matematica, in particolare problemi di programmazione lineare (LP) e programmazione lineare intera mista (MIP). CPLEX utilizza una varietà di algoritmi, tra cui branch&bound e branch&cut, per trovare la soluzione ottima al problema. Il software è ampiamente utilizzato in ambito accademico e industriale per la risoluzione di problemi di ottimizzazione relativi a diversi settori di studio.[8]

La versione utilizzata è la 22.1.0.

4.1.5 Dati estratti

Come anticipato nella sezione 4.1.1, i dati sono stati estratti da dei file in formato *PDDL* resi disponibili direttamente dagli autori del paper. Per ottenere la struttura originale dell'organismo in modo da generare il modello lineare intero, è stata aggiunta al programma, di cui si parla nella sezione 4.1.3, la possibilità di leggere i dati direttamente da un file in formato *PDDL*.

4.1.6 Verifica della soluzione

Un modo per verificare se la soluzione è ammissibile, ovvero se il minimal set che è stato trovato ci permette effettivamente di raggiungere l'intero set, è il seguente:

1. Aggiungere gli elementi della soluzione trovata all'insieme $\mathcal{S}$. Inserire nell'insieme $\mathcal{C}$ tutti gli elementi dell'organismo.

[1] In aggiunta naturalmente ai processi necessari per il corretto funzionamento del sistema operativo.

2. Sia ora $\mathcal{S}' = \mathcal{S}$. Per ogni reazione possibile, controllare se tutti i reagenti necessari per il suo completamento sono contenuti in $\mathcal{S}$. Se essi sono presenti, aggiungere a $\mathcal{S}'$ tutte le molecole prodotte dalla reazione.

3. Se, dopo aver "applicato" tutte le reazioni possibili, sono stati aggiunti dei nuovi elementi all'insieme (ovvero $\mathcal{S}' \supset \mathcal{S}$), ripetere il passaggio 2 espandendo $\mathcal{S} = \mathcal{S} \cup \mathcal{S}'$. Se invece $\mathcal{S}' = \mathcal{S}$, si possono verificare le due seguenti condizioni:

   (a) Se $\mathcal{S} = \mathcal{C}$, la soluzione di partenza è ammissibile.

   (b) Altrimenti, la soluzione di partenza non è ammissibile.

Oltre a verificare se una soluzione è ammissibile o meno, l'algoritmo visto in precedenza permette di ottenere il numero di istanti temporali necessari all'insieme di partenza per venir espanso all'insieme completo. Per farlo sará necessario contare il numero di esecuzioni del punto 2.

4.1.7 Performance variability

I moderni risolutori (come ad esempio CPLEX) utilizzano tecniche basate anche su fattori casuali. Nonostante gli algoritmi utilizzati, come ad esempio branch&bound e branch&cut, siano deterministici, spesso è necessario ricorrere alla generazione di numeri randomici per scegliere tra più alternative ritenute ugualmente "promettenti". Questo avviene ad esempio nel caso di euristiche o nella scelta delle variabili da utilizzare per fare il branching.[5]

Per poter quindi valutare le performance della risoluzione e confrontare vari approcci senza dipendere troppo dalla variabilitá intrinseca del risolutore, si è scelto di risolvere ogni istanza k volte e di utilizzare il tempo medio di risoluzione come metrica. Prima di iniziare la risoluzione di una determinata istanza verrá quindi generato un numero casuale che sará poi utilizzato come seed per il risolutore. CPLEX ci permette di impostare il seed tramite un parametro.

4.2 Istanze di test

Avendo a disposizione due formulazioni per lo stesso problema, verranno presentate delle istanze di test relative ad entrambe. Seguirá poi una comparazione tra le prestazioni delle due.

4.2.1 Prima formulazione

Come prima cosa è stata risolta un' istanza modellata con la prima formulazione. Sono stati utilizzati entrambi i passaggi di *preprocessing* (separazione delle reazioni e rimozione dei duplicati). Come valore di M si è scelto di prendere

$$M = \min\{|I|, |J|\}$$

Questo valore è naturalmente diverso per ognuno degli organismi e viene automaticamente calcolato e utilizzato nel modello. Il time limit è stato impostato a 60 minuti.

I risultati della risoluzione sono presentati nella tabella 4.1.

Analizzando la tabella si puó notare come la maggior parte delle istanze venga risolta in un tempo ragionevole. Per l'organismo *HSA*, uno dei piú complessi, 1 delle 20 istanze non riesce ad essere risolta entro il time limit.

È possibile poi vedere, in figura 4.1, la distribuzione dei tempi di risolzione per ogni istanza.

Il grafico a violino, oltre a rappresentare il valore minimo e il valore massimo, permette di avere una rappresentazione di come sono distributi i dati all'interno dell'intervallo.

Organismo	Tempo medio (s)	Risolti	Timelimit	Gap medio
aae	3.86	20	0	0.00%
ath	12.81	20	0	0.00%
avn	0.20	20	0	0.00%
ayw	0.25	20	0	0.00%
bmu	12.05	20	0	0.00%
bra	7.45	20	0	0.00%
bxe	18.88	20	0	0.00%
cme	5.04	20	0	0.00%
cre	0.14	20	0	0.00%
dme	15.14	20	0	0.00%
dre	41.37	20	0	0.00%
ecc	28.27	20	0	0.00%
eco	38.14	20	0	0.00%
ecp	21.57	20	0	0.00%
ecv	32.25	20	0	0.00%
ecx	104.25	20	0	0.00%
gga	23.16	20	0	0.00%
hsa	331.91	19	1	0.36%
mmu	36.17	20	0	0.00%
rha	8.70	20	0	0.00%
sce	10.12	20	0	0.00%
xla	6.38	20	0	0.00%

Tabella 4.1: Risoluzione prima formulazione con $M = \min\{|I|, |J|\}$ e preprocessing

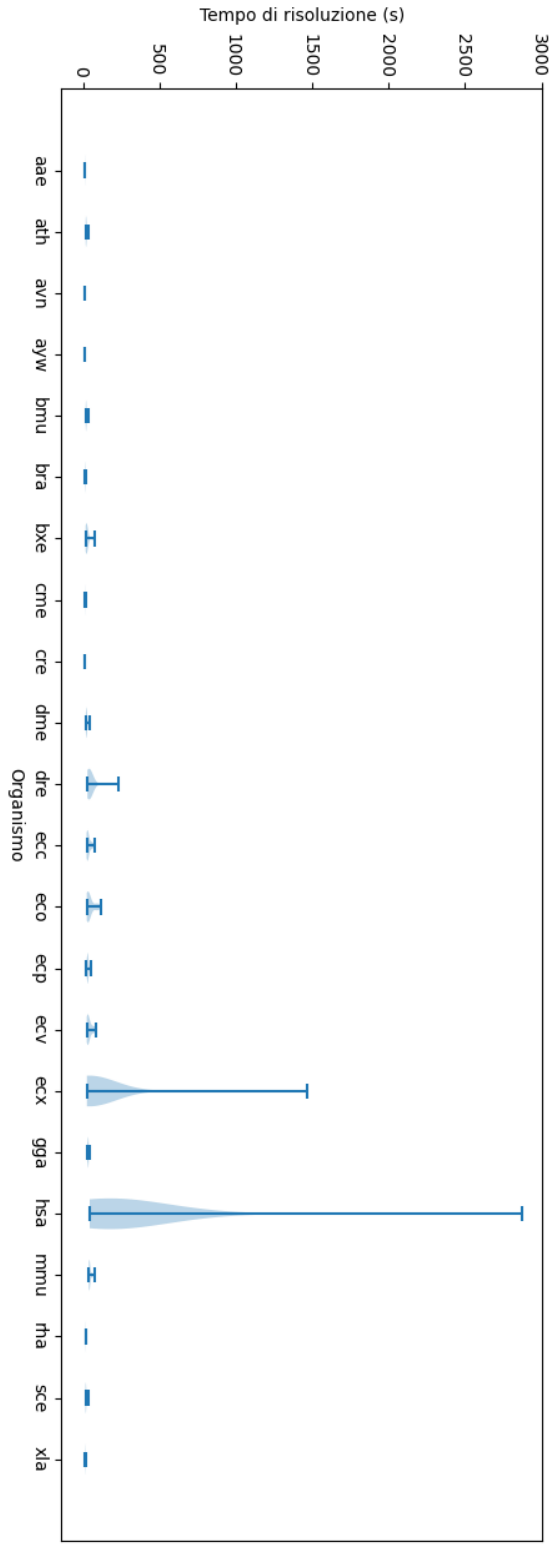


Figura 4.1: Distribuzione risultati prima formulazione con $M = \min\{|I|, |J|\}$ e preprocessing

Dopo aver visto che risultati si riescono ad ottenere applicando i passaggi di *preprocessing*, viene fatto lo stesso test senza utilizzarli.

Tutti i parametri vengono mantenuti uguali ($M = \min\{|I|, |J|\}$ e time limit di 60 minuti) ma non vengono né separate le reazioni né rimossi i duplicati.

I risultati sono mostrati nella tabella 4.2.

Si puó notare che un numero piú grande di istanze non viene risolto entro il time limit prefissato. Tuttavia la maggior parte continuano ad essere risolte.

Come in precedenza, in figura 4.2 é visibile, in un grafico a violino, la rappresentazione della distribuzione dei tempi di risoluzione.

Organismo	Tempo medio (s)	Risolti	Timelimit	Gap medio
aae	4.05	20	0	0.00%
ath	23.89	20	0	0.00%
avn	0.21	20	0	0.00%
ayw	0.24	20	0	0.00%
bmu	18.00	20	0	0.00%
bra	13.22	20	0	0.00%
bxe	28.29	20	0	0.00%
cme	7.03	20	0	0.00%
cre	0.22	20	0	0.00%
dme	27.66	20	0	0.00%
dre	69.77	17	3	0.37%
ecc	36.24	20	0	0.00%
eco	37.04	20	0	0.00%
ecp	51.22	20	0	0.00%
ecv	48.57	20	0	0.00%
ecx	36.83	20	0	0.00%
gga	178.02	18	2	0.34%
hsa	727.42	15	5	0.36%
mmu	417.52	19	1	0.36%
rha	13.47	20	0	0.00%
sce	15.32	20	0	0.00%
xla	11.65	20	0	0.00%

Tabella 4.2: Risoluzione prima formulazione con $M = \min\{|I|, |J|\}$ senza preprocessing

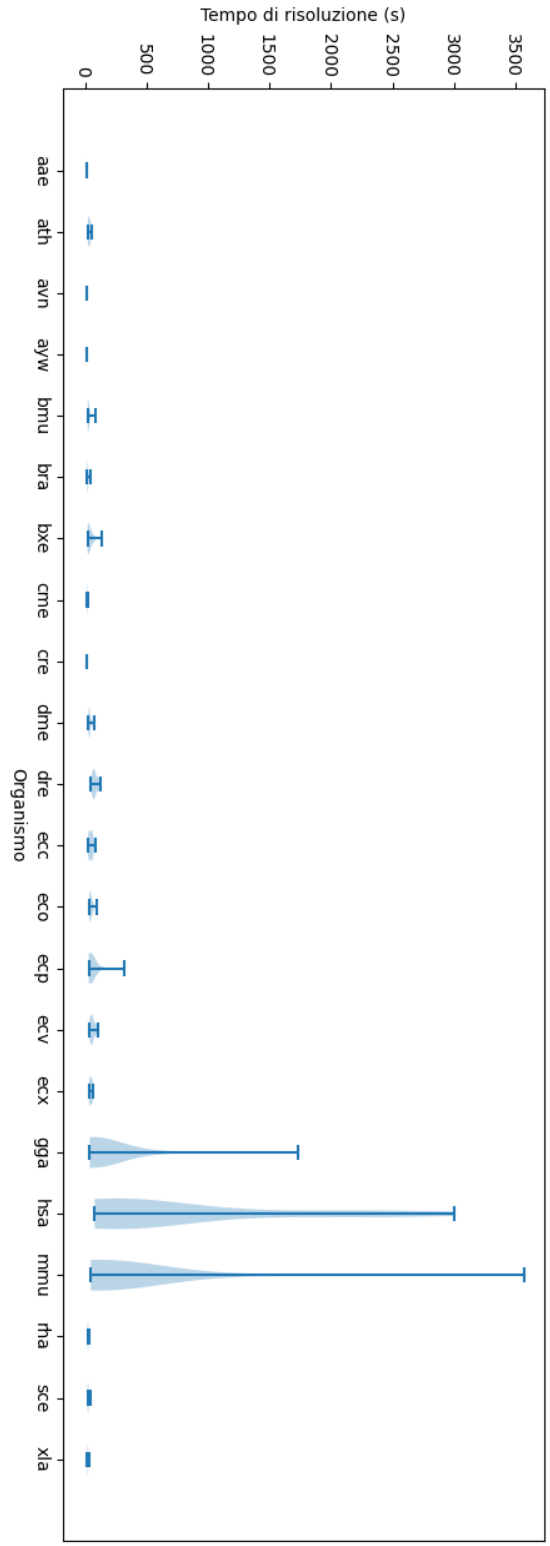


Figura 4.2: Distribuzione risultati prima formulazione con $M = \min\{|I|, |J|\}$ senza preprocessing

Viene presentata ora una comparazione tra i risultati precedenti in modo da poter trarre delle conclusioni sull'impatto dell'applicazione del preprocessing sulle prestazioni della risoluzione.

Si puó notare come l'applicazione del preprocessing abbia come effetto sia il miglioramento del tempo di risoluzione sia una riduzione delle istanze che non vengono risolte entro il timelimit prefissato.

	Istanze risolte		Tempo medio (s)	
Organismo	Applicato	Non applicato	Applicato	Non applicato
aae	20/20	20/20	3.87	4.05
ath	20/20	20/20	12.82	23.89
avn	20/20	20/20	0.21	0.21
ayw	20/20	20/20	0.25	0.24
bmu	20/20	20/20	12.06	18.01
bra	20/20	20/20	7.45	13.22
bxe	20/20	20/20	18.88	28.29
cme	20/20	20/20	5.04	7.04
cre	20/20	20/20	0.15	0.22
dme	20/20	20/20	15.15	27.66
dre	20/20	17/20	41.37	69.77
ecc	20/20	20/20	28.27	36.24
eco	20/20	20/20	38.15	37.05
ecp	20/20	20/20	21.57	51.22
ecv	20/20	20/20	32.25	48.57
ecx	20/20	20/20	104.25	36.83
gga	20/20	18/20	23.17	178.02
hsa	19/20	15/20	331.91	727.43
mmu	20/20	19/20	36.17	417.52
rha	20/20	20/20	8.71	13.47
sce	20/20	20/20	10.12	15.32
xla	20/20	20/20	6.39	11.65

Tabella 4.3: Comparazione effetti preprocessing sulla prima formulazione

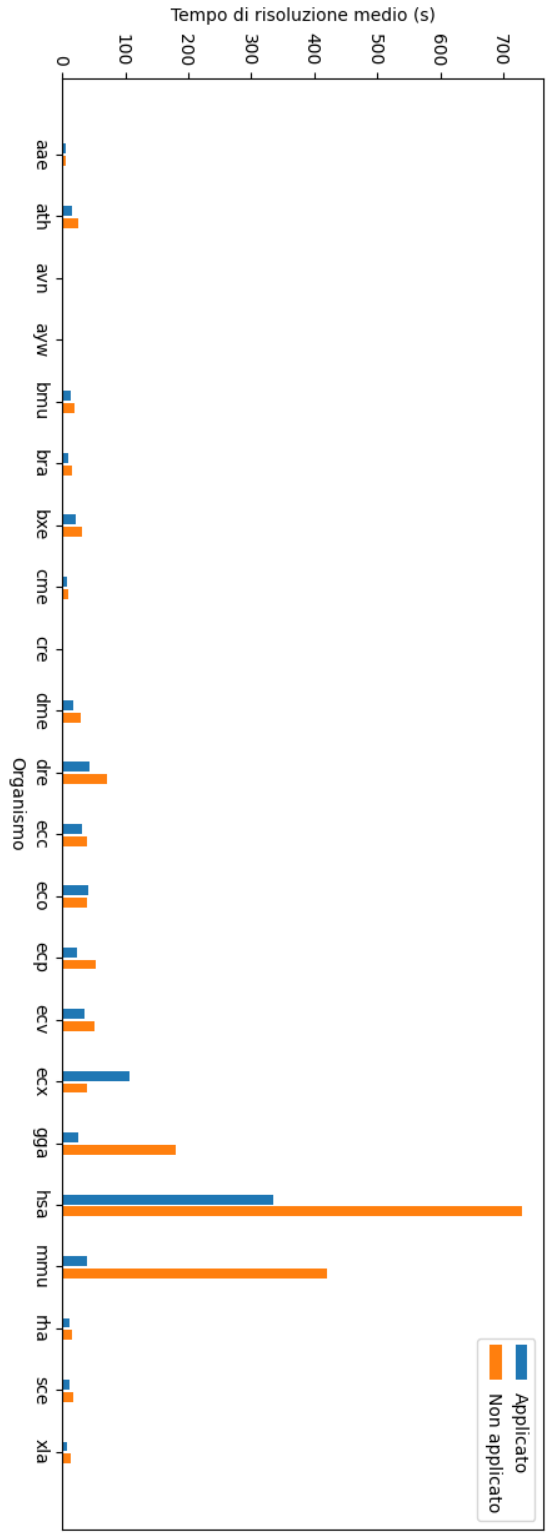


Figura 4.3: Comparazione effetti preprocessing sulla prima formulazione

Come prova aggiuntiva, dopo aver risolto le istanze con i metodi precedenti ed aver calcolato il numero di iterazioni necessarie, è possibile provare a risolvere qualche istanza con un valore di M piú piccolo.

Si fa notare che il modello non minimizza il numero di iterazioni necessarie per raggiungere l'insieme completo ma bensí il numero di molecole da aggiungere al set. Da questo se ne deriva che non necessariamente il numero di iterazioni trovato è il minimo possibile per un determinato organismo. L'unica cosa che puó essere detta è che esiste sicuramente una soluzione ottima (con numero di elementi minimo nel *seed set*) con quel determinato numero di passaggi necessari a raggiungere l'insieme completo visto che è stata trovata.

Sono state utilizzate le soluzioni ottime trovate dalla risoluzione con

$$M = \min\{|I|, |J|\}$$

e passaggi di preprocessing (presentata all'inizio della sezione).

È stato poi applicato l'algoritmo presentato nella sezione 4.1.6 per calcolare il numero di iterazioni necessarie ad espandere il *seed set* all'insieme completo.

Nella tabella 4.4 sono visibili i risultati ottenuti.

Organismo	Iterazioni
aae	18
ath	36
avn	22
ayw	6
bmu	13
bra	15
bxe	27
cme	20
cre	5
dme	57
dre	58
ecc	19
eco	16
ecp	15
ecv	16
ecx	22
gga	58
hsa	58
mmu	58
rha	27
sce	29
xla	35

Tabella 4.4: Numero di iterazioni necessarie

Viene scelto quindi $M = 60$ in modo da avere un valore di M che vada bene per risolvere tutte le istanze. Vengono risolte ora 20 istanze per ogni organismo utilizzando questo valore di M. Vengono applicati i passaggi di preprocessing e viene impostato un time limit di 60 minuti (anche se, come si potrá vedere, esso non viene mai raggiunto). Nella tabella 4.5 sono visibili i risultati che sono stati ottenuti.

Organismo	Tempo medio (s)	Risolti	Timelimit	Gap medio
aae	2.77	20	0	0.00%
ath	6.34	20	0	0.00%
avn	0.21	20	0	0.00%
ayw	0.23	20	0	0.00%
bmu	6.14	20	0	0.00%
bra	4.87	20	0	0.00%
bxe	8.40	20	0	0.00%
cme	3.64	20	0	0.00%
cre	0.14	20	0	0.00%
dme	9.46	20	0	0.00%
dre	17.20	20	0	0.00%
ecc	12.69	20	0	0.00%
eco	14.23	20	0	0.00%
ecp	12.46	20	0	0.00%
ecv	17.30	20	0	0.00%
ecx	15.20	20	0	0.00%
gga	16.64	20	0	0.00%
hsa	55.51	20	0	0.00%
mmu	25.38	20	0	0.00%
rha	5.09	20	0	0.00%
sce	6.76	20	0	0.00%
xla	4.15	20	0	0.00%

Tabella 4.5: Risoluzione prima formulazione con $M = 60$ e preprocessing

Come si puó vedere tutte le istanze vengono risolte e i tempi sono migliori di quelli ottenuti in precedenza. Tuttavia questo test é stato possibile solo dopo aver risolto le istanze precedenti con cui abbiamo trovato il valore di $M = 60$ e quindi non sarebbe stato possibile considerarlo valido a priori.

4.2.2 Seconda formulazione

Viene risolta ora un'istanza per organismo utilizzando la seconda formulazione. Il valore di M é

$$M = \min\{|I|, |J|\}$$

come per la formulazione precedente. Non sono stati applicati i passaggi di preprocessing.

Organismo	Gap
aae	11.03%
ath	OOM
avn	14.71%
ayw	0.00%
bmu	OOM
bra	OOM
bxe	OOM
cme	5.17%
cre	4.55%
dme	2.20%
dre	100.00%
ecc	6.10%
eco	OOM
ecp	OOM
ecv	5.79%
ecx	5.71%
gga	100.00%
hsa	100.00%
mmu	100.00%
rha	2.16%
sce	4.89%
xla	5.96%

Tabella 4.6: Risoluzione seconda formulazione con $M = \min\{|I|, |J|\}$ senza preprocessing

Come é possibile vedere, quasi nessuna delle istanze viene risolta e, al termine della risoluzione, il gap rimanente è considerevole. Nei casi in cui appare *OOM* il risolutore ha esaurito la memoria a sua disposizione durante l'esecuzione. Nei casi in cui invece il gap ammonta al 100%, dopo i 60 minuti di time limit il risolutore si trovava ancora in una fase di pre-risoluzione e quindi non era ancora stato trovato un bound al problema.

Viene eseguito ora lo stesso test, questa volta applicando i passaggi di preprocessing.

Organismo	Gap
aae	11.03%
ath	OOM
avn	14.71%
ayw	0.00%
bmu	6.03%
bra	100.00%
bxe	100.00%
cme	4.76%
cre	6.82%
dme	2.20%
dre	100.00%
ecc	6.10%
eco	OOM
ecp	OOM
ecv	5.79%
ecx	5.71%
gga	100.00%
hsa	100.00%
mmu	100.00%
rha	27.43%
sce	4.89%
xla	5.96%

Tabella 4.7: Risoluzione seconda formulazione con $M = \min\{|I|, |J|\}$ e preprocessing

Anche in questo caso solamente un organismo viene risolto, il gap rimanente è molto alto e molte istanze finiscono la memoria disponibile.

Prendendo ora in considerazione i risultati ottenuti con la prima formulazione, si prova a risolvere un'istanza per ogni organismo utilizzando un valore di $M = 60$. Questa informazione è derivata dai dati disponibili nella tabella 4.4. Come già detto in precedenza, si puó prendere questo valore di M solamente perché le istanze sono state risolte in precedenza e si conosce almeno una soluzione ottima raggiungibile con tale valore.

Vengono applicati i passaggi di preprocessing e viene impostato un time limit pari a 60 minuti. I risultati sono visibili nella tabella 4.8.

Organismo	Gap
aae	11.03%
ath	3.51%
avn	8.82%
ayw	0.00%
bmu	6.03%
bra	3.49%
bxe	4.58%
cme	4.33%
cre	2.27%
dme	1.10%
dre	2.21%
ecc	6.10%
eco	4.71%
ecp	5.71%
ecv	5.79%
ecx	4.90%
gga	2.07%
hsa	2.54%
mmu	2.90%
rha	1.08%
sce	4.44%
xla	1.79%

Tabella 4.8: Risoluzione seconda formulazione con $M = 60$ e preprocessing

Si puó notare che, nonostante i gap rimanenti siano piú bassi, anche per questo valore di M solamente un'istanza viene risolta.

La prima formulazione è quindi sicuramente superiore in quanto consente di risolvere il maggior numero di istanze di ogni organismo preso in esame.

4.2.3 Dimensione del problema

Per provare a quantificare questa differenza, sono stati raccolti i dati relativi al numero di colonne, numero di righe e numero di coefficienti diversi da zero di ogni problema. Vengono confrontati quindi questi valori per la prima e per la seconda formulazione.

Per alcune risoluzioni con la seconda formulazione, i valori non sono riportati perché la fase di presolve non è terminata entro il time limit.

	Righe		Colonne		Non zero	
org	1	2	1	2	1	2
aae	2475	1 211 358	2354	1 130 288	6910	3 103 090
ath	4907	4 832 374	4728	4 588 576	13 476	12 396 276
avn	677	86 698	612	75 648	1976	219 872
ayw	578	70 582	541	64 181	1485	175 221
bmu	4237	3 586 189	4083	3 406 471	11 711	9 215 211
bra	5407	5 922 262	5240	5 667 754	14 692	15 187 510
bxe	5458	5 941 498	5269	5 657 998	15 117	15 288 725
cme	3283	2 151 847	3141	2 023 621	9065	5 510 571
cre	808	139 936	775	131 719	2031	349 313
dme	4386	3 840 672	4217	3 636 689	12 121	9 859 381
dre	5130	5 200 414	4949	4 949 005	14 373	13 418 341
ecc	4315	3 604 495	4125	3 389 605	12 353	9 317 571
eco	4472	3 868 982	4276	3 639 466	12 812	10 004 255
ecp	4354	3 656 968	4163	3 440 183	12 515	9 464 001
ecv	4330	3 619 388	4137	3 401 298	12 433	9 361 297
ecx	4404	3 737 104	4209	3 513 634	12 673	9 672 609
gga	5000	5 017 618	4842	4 798 314	13 754	12 904 940
hsa	5451	5 864 011	5275	5 604 763	15 311	15 160 727
mmu	5425	5 803 294	5241	5 533 918	15 245	14 994 897
rha	5403	5 854 182	5222	5 583 044	14 866	15 045 178
sce	3655	2 631 214	3493	2 471 806	10 257	6 764 453
xla	4252	3 652 964	4109	3 482 365	11 589	9 363 321

Tabella 4.9: Dimensione problema con preprocessing prima del presolve

	Righe		Colonne		Non zero	
org	1	2	1	2	1	2
aae	1087	-	1336	-	4181	-
ath	1780	220 290	2118	217 396	6847	755 007
avn	333	18 554	380	15 944	1278	58 650
ayw	78	12 389	103	11 348	304	41 938
bmu	1591	184 751	1930	177 660	6136	641 628
bra	1751	-	2114	-	6720	-
bxe	1995	-	2388	-	7643	-
cme	1190	136 345	1441	132 649	4614	468 333
cre	82	31 615	104	27 653	314	101 216
dme	1509	266 689	1792	247 328	5841	914 463
dre	1928	430 499	2314	406 799	7455	1 488 330
ecc	1968	195 788	2357	184 414	7636	676 471
eco	2044	182 914	2441	173 489	7885	632 884
ecp	1970	184 111	2359	174 950	7651	637 245
ecv	2008	190 176	2411	181 056	7802	658 541
ecx	2041	179 051	2441	169 830	7877	619 538
gga	1559	464 807	1876	438 347	6030	1 611 902
hsa	2077	-	2512	-	8018	-
mmu	1916	-	2288	-	7414	-
rha	1858	214 167	2241	197 690	7128	699 332
sce	1461	151 504	1732	144 583	5631	534 977
xla	1278	325 535	1533	305 189	4944	1 111 066

Tabella 4.10: Dimensione problema con preprocessing dopo il presolve

Come si puó vedere, le dimensioni delle istanze ottenute con la seconda formulazione sono ordini di grandezza piú grandi. Questo conferma quindi le conclusioni che erano state tratte dai risultati sperimentali ottenuti con la risoluzione dei problemi.

Oltre alle dimensioni delle istanze, sono stati estratti dal risolutore anche i valori del primo rilassamento lineare. Questo valore risulta essere uguale per entrambe le formulazioni ed è riportato in tabella 4.11. In tutti e due i casi è stato applicato il preprocessing.

Organismo	1	2
aae	668	668
ath	1360	1360
avn	168	168
ayw	171	171
bmu	1165	1165
bra	1522	1522
bxe	1498	1498
cme	901	901
cre	247	247
dme	1205	1205
dre	1387	1387
ecc	1129	1129
eco	1169	1169
ecp	1133	1133
ecv	1128	1128
ecx	1144	1144
gga	1386	1386
hsa	1471	1471
mmu	1462	1462
rha	1496	1496
sce	982	982
xla	1191	1191

Tabella 4.11: Valore del rilassamento iniziale per le due formulazioni

4.2.4 Preprocessing

Puó essere interessante ora andare a confrontare, per la prima formulazione, le dimensioni dei problemi con o senza preprocessing.

	Righe		Colonne		Non zero	
Organismo	No	Sì	No	Sì	No	Sì
aae	2531	2475	2408	2354	7189	6910
ath	5076	4907	4893	4728	14 319	13 476
avn	683	677	618	612	2006	1976
ayw	625	578	588	541	1720	1485
bmu	4366	4237	4208	4083	12 354	11 711
bra	5581	5407	5409	5240	15 559	14 692
bxe	5640	5458	5447	5269	16 025	15 117
cme	3425	3283	3280	3141	9773	9065
cre	865	808	830	775	2315	2031
dme	4604	4386	4432	4217	13 209	12 121
dre	5418	5130	5234	4949	15 811	14 373
ecc	4496	4315	4303	4125	13 256	12 353
eco	4653	4472	4454	4276	13 715	12 812
ecp	4537	4354	4343	4163	13 428	12 515
ecv	4513	4330	4317	4137	13 346	12 433
ecx	4580	4404	4382	4209	13 551	12 673
gga	5260	5000	5100	4842	15 053	13 754
hsa	5778	5451	5598	5275	16 943	15 311
mmu	5734	5425	5546	5241	16 787	15 245
rha	5598	5403	5413	5222	15 839	14 866
sce	3827	3655	3663	3493	11 116	10 257
xla	4469	4252	4323	4109	12 672	11 589

Tabella 4.12: Confronto dimensione problema prima del presolve con o senza preprocessing

	Righe		Colonne		Non zero	
Organismo	No	Sì	No	Sì	No	Sì
aae	1126	1087	1375	1336	4370	4181
ath	1890	1780	2228	2118	7374	6847
avn	333	333	380	380	1278	1278
ayw	114	78	139	103	478	304
bmu	1680	1591	2019	1930	6566	6136
bra	1862	1751	2224	2114	7250	6720
bxe	2117	1995	2511	2388	8218	7643
cme	1321	1190	1571	1441	5229	4614
cre	95	82	116	104	369	314
dme	1659	1509	1941	1792	6549	5841
dre	2212	1928	2611	2314	8739	7455
ecc	2135	1968	2524	2357	8433	7636
eco	2214	2044	2611	2441	8694	7885
ecp	2135	1970	2524	2359	8432	7651
ecv	2180	2008	2583	2411	8616	7802
ecx	2206	2041	2606	2441	8661	7877
gga	1747	1559	2065	1876	6923	6030
hsa	2463	2077	2930	2512	9710	8018
mmu	2275	1916	2680	2288	9002	7414
rha	1977	1858	2360	2241	7693	7128
sce	1605	1461	1876	1732	6297	5631
xla	1431	1278	1685	1533	5664	4944

Tabella 4.13: Confronto dimensione problema dopo il presolve con o senza preprocessing

Si puó notare come l'applicazione dei passaggi di preprocessing permetta una leggera riduzione della dimensione del problema.

4.2.5 Risultati

Vengono riportati ora i valori ottimi della funzione obiettivo, ovvero le dimensioni dei *Minimal Seed Set*. Questi risultati sono stati ottenuti grazie alle istanze risolte con la prima formulazione che permette una risoluzione completa di tutti gli organismi.

Organismo	Valore ottimo
aae	145
ath	313
avn	34
ayw	64
bmu	282
bra	372
bxe	349
cme	231
cre	88
dme	273
dre	272
ecc	246
eco	255
ecp	245
ecv	242
ecx	245
gga	290
hsa	276
mmu	276
rha	371
sce	225
xla	280

Tabella 4.14: Valori ottimi funzione obiettivo

Capitolo 5

Conclusione

Dopo aver visto i risultati che si possono ottenere grazie alla prima formulazione, presentata nella sezione 3.1, si puó concludere che è sicuramente possibile scrivere una formulazione *MIP* che risolva il problema e che scali bene anche per gli organismi più complessi (come ad esempio HSA[1]).

Infatti, come è stato mostrato nella sezione 4.2.1, con un valore di M pari al minimo tra il numero di reazioni e il numero di molecole è giá possibile risolvere un buon numero di istanze per ogni organismo. Eseguendo poi dei semplici passaggi di *preprocessing* si riescono a migliorare ulteriormente le prestazioni.

È stata anche data la dimostrazione di come sia possibile scrivere una formulazione simile a quella funzionante senza però avere successo. La seconda modellazione del problema non permette la risoluzione all'ottimo di nessuna istanza. Nonostante ció è stata inclusa nella tesi perché utilizzata come bound dell'altra formulazione.

Pur non conoscendo la modellazione che gli autori del paper[2] hanno utilizzato senza successo, è possibile affermare che, avendo a disposizione una formulazione adeguata, si riesce a risolvere il problema con un approccio MIP.

[1] Homo Sapiens

Bibliografia

[1] Tobias Achterberg and Roland Wunderling. Mixed integer programming: Analyzing 12 years of progress. *Facets of Combinatorial Optimization: Festschrift for Martin Grötschel*, pages 449–481, 2013.

[2] Elhanan Borenstein, Martin Kupiec, Marcus Feldman, and Eytan Ruppin. Large-scale reconstruction and phylogenetic analysis of metabolic environments. *Proceedings of the National Academy of Sciences of the United States of America*, 105:14482–7, 10 2008.

[3] The Rust OR Developers. Rust Linear Programming Modeler. https://github.com/rust-or/rust-lp-modeler, 2023. Visitato: April 3, 2023.

[4] Kanehisa et al. From genomics to chemical genomics: new developments in kegg. *Nucleic acids research*, 34(suppl_1):D354–D357, 2006.

[5] Matteo Fischetti and Michele Monaci. Exploiting erraticism in search. *Operations Research*, 62(1):114–122, 2014.

[6] Amitai Gefen. Minimal seed set. https://github.com/gefena/minimal_seed_set, 2023. Visitato: April 3, 2023.

[7] Avitan Gefen and Ronen Brafman. The minimal seed set problem. *Proceedings of the International Conference on Automated Planning and Scheduling*, 21(1):319–322, Mar. 2011.

[8] IBM Corporation. *IBM ILOG CPLEX Optimization Studio Documentation*. IBM Corporation, Armonk, NY, 22.1.0 edition, 2023.

[9] Josef Jablonskỳ et al. Benchmarks for current linear and mixed integer optimization solvers. *Acta Universitatis Agriculturae et Silviculturae Mendelianae Brunensis*, 63(6):1923–1928, 2015.

[10] Kyoto Encyclopedia of Genes and Genomes (KEGG). Kegg pathway. https://www.kegg.jp/, 2023. Visitato: April 3, 2023.

[11] Hans D Mittelmann. Benchmarking optimization software-a (hi) story. In *SN operations research forum*, volume 1, page 2. Springer, 2020.

[12] The Rust Project Developers. The rust programming language. https://www.rust-lang.org, 2023. Visitato: April 3, 2023.