UNIVERSITÁ POLITECNICA DELLE MARCHE

FACOLTA DI INGEGNERIA‘
Corso di Laurea in Ingegneria Informatica e dell’ Automazione

Progettazione e realizzazione di un algoritmo per la scelta ottimale della blockchain integrata nel sistema IntegrityKEY

Design and implementation of an algorithm for the optimal choice of the blockchain integrated into the IntegrityKEY system

Relatore:
Chiar.mo prof.
Simone Fiori

Tesi di Laurea di:
Gian Marco Troni

anno accademico 2023 / 2024

Indice

1 introduzione **1**
1.1 Fasi di progettazione . . . 1
1.2 Predisposizione Progetto . . . 1
1.2.1 virtual env . . . 1
1.2.2 Web framework Djngo . . . 2
1.2.3 architettura . . . 2
1.2.4 Architettura utilizzata da Django . . . 3

2 block chain **4**
2.1 Ledger . . . 4
2.2 Definizioni blockchain . . . 4
2.2.1 Blockchain come database . . . 5
2.2.2 Rete decentralizzata di tipo peer-to-peer . . . 6
2.2.3 Asset Digitali Unici . . . 6
2.3 Componenti della blockchain . . . 6
2.3.1 L'albero di Merkle . . . 8
2.3.2 Merkle proofs . . . 9
2.4 Le Distributed Ledger Technology . . . 9
2.5 Algoritmi di consenso . . . 10
2.5.1 Proof of work (PoW) . . . 10
2.5.2 proof of stake (PoS) . . . 11
2.6 Soluzioni di scalabilità blockchain . . . 11
2.6.1 channel state . . . 11
2.6.2 sidechain . . . 12
2.7 Permissionless o Permissioned Ledger . . . 12
2.8 Fork . . . 13
2.8.1 Soft Fork . . . 13
2.8.2 Hard Fork . . . 13
2.9 token . . . 13
2.9.1 Classificazione in base ai diritti . . . 14
2.9.2 smart contract . . . 14

2.9.2.1 NFT 15

3 ricerca operativa **16**
3.1 introduzione 16
3.2 problemi 16
3.2.1 problema di ottimizzazione 17
3.2.2 soluzioni problema di ottimizzazione 17
3.2.3 problema matematico 17
3.3 Algoritmo 18
3.3.1 Approssimazione 18
3.3.2 Rilassamenti 18
3.4 Programmazione Lineare 19
3.4.1 Tipi di variabili decisionali 19
3.5 Algoritmi enumerativi 20
3.5.1 Branch and Buond 20
3.5.2 Branch and cut 22
3.5.3 Branch and price and cut 23
3.6 La modellazione del problema 25
3.6.1 Problema dello zaino (ksnap) 25
3.6.2 Risoluzione 26

4 data base **28**
4.1 verifica dei dati 28
4.2 glossario 28
4.3 diagramma 29
4.4 tabella entità 29

5 algoritmo **32**
5.1 obiettivi algoritmo 32
5.2 settaggio algoritmo 32
5.3 implementazione librerie nell'algoritmo 32
5.4 algoritmo 32
5.4.1 implementazione in django 34

6 test algoritmo **42**
6.1 casi trattati test 42
6.2 test 42
6.3 risultati algoritmo 43

7 conclusioni **45**
7.1 conclusioni finali 45
7.2 esperienza 46
7.3 Futuro 46
documenetazione 47

block chain 47
ricerca operativa 48

Elenco delle figure

1.1 Model-Visual-Controller . . . 3
1.2 Model-Visual-Template . . . 3

2.1 traslazione . . . 7
2.2 blockchain . . . 7

3.1 algoritomo Branch End Bound . . . 20
3.2 algoritomo CBS MIPS resolver . . . 23
3.3 schema a blocchi Branch and price . . . 24

4.1 digramma delle entità . . . 29

5.1 algoritmo generale . . . 33
5.2 use case algoritmo . . . 33
5.3 presa dati da db: "get_data_db_in" . . . 34
5.4 presa dati da db: "set_objects_get" . . . 34
5.5 classe inserimento: "_take_querryset" . . . 35
5.6 classe inserimento: "get_colunms" . . . 35
5.7 classe inserimento: "___elem_map" . . . 35
5.8 classe inserimento: "___field_get" . . . 35
5.9 classe inserimento: "___querrySet_get . . . 35
5.10 classe inserimento: "___check_param" . . . 36
5.11 classe inserimento: "conv_list" . . . 36
5.12 classe : "___init___" . . . 37
5.13 classe : "var_config" . . . 37
5.14 classe : "constraints_model" . . . 37
5.15 classe : "___call___" e "_objecttive_model" e "___resolver_model" . . . 37
5.16 classe : "___init___" . . . 38
5.17 classe : "var_constraints_model" . . . 38
5.18 classe : "resolver_model" e "___resolver_model" . . . 38
5.19 classe base dell'algoritmo: "___init___" e "___get_data" . . . 39
5.20 classe base dell'algoritmo: "_f_group_var_sol" . . . 40

5.21 classe base dell'algoritmo:"___call___" nelle classi derivate "PuLPModel" e "ortoolsModel" 40
5.22 implementato la libreria pulp: "___solve_problem" 41
5.23 implementato la libreria or-tools: "___solve_problem" 41

Elenco delle tabelle

3.2 db tabella entità 29

5.1 funzioni creazione matrici e vettori 36
5.3 funzioni base dell'algoritimo 39

6.1 tabella risultati per tipo di blockchin 44
6.2 tempi prove: tempi risolutori 44
6.3 tabella risultati: blockchin scelte 44

Sommario

Questa trattato è Esplorazione dell'idea di progetto della realizzazione di un algoritmo per la scelta ottimale della blockchain integrata nel sistema IntegrityKEY.
In questo trattato si utilizzerà spesso queste due tecnologie correlate la "blockchain" e il "database".
Il database è una collezione di dati organizzati a parametri immagazzinata e accessibile per via elettronica.
La blockchain è una sottofamiglia di tecnologie in cui il registro è strutturato come una catena di blocchi contenenti le transazioni e la cui validazione è affidata a un meccanismo di consenso, distribuito su tutti i nodi della rete. Le principali caratteristiche delle tecnologie blockchain sono l'immutabilità del registro, la trasparenza, tracciabilità delle transazioni e la sicurezza basata su tecniche crittografiche.
Per lo sviluppo dell'algoritmo una delle tecnologie utilizzate è quella del "database" per immagazzinare e modellare i dati basati sui concetti "blockchain", creandone la sua una "immagine" immessa in un database di "immagini" di blockchain.
L'immagine dell'blockchain è la schematizzazione dei parametri di una blockchain all'interno del database, che contiene parametri comuni usate nelle blockchain. L'immissione di essa è andando a selezionare i parametri di una singola blockchain per creare la sua "immagine" come se fosse una sua istantanea.
Infine, per scegliere la blockchain ottimale utilizzo metodi di risoluzione basati sulla ricerca operativa. Con l'utilizzo di ricerca operativa setto il problema PL (programmazione lineare) a numeri misti (interi e a virgola mobile) per ogni blockchain.
I singoli problemi creati selezionando i parametri presenti nel database "dell'immagini" di blockchain. Per risolvere il problema di ricerca operativa utilizzo i metodi basiti sul B&B (Branch and Buond), questi metodi sono implementati in varie librerie. le varie librerie utilizzano le finzioni di risoluzione per risolvere il problema di ricerca operativa.
Le funzioni di risoluzione possono essere di massimo e minimo, avendone anche più di una. Le funzioni di risoluzione utilizzano i parametri valori puntuali o/e massimi o/e minimi o/e medi. di risoluzione. Nei test fatti utilizzo parametri come numero di elementi o/e velocità o/e costi di traslazioni.
Alla fine del trattato vado a vedere la motivazione di come sono stati eseguiti i test e la scelta i dati dammy (che simulino un caso di utilizzo reale) la precisione del risultato, andando a valutare la velocità di calcolo del risultato ed altre caratteristiche.

Capitolo 1: introduzione

Nel progetto esplorativo descritto in questo trattato, prima iniziare a esporre andiamo a introdurre le varie fasi di esso.

1.1 Fasi di progettazione

Nel sistema IntegrityKEY si utilizza Blockchain, all' interno impiega smart contact come asset digitale di un servizio di tracciamento del prodotto. In questa situazione bisognerebbe trovare un algoritmo per ottimizzare uso della blockchain ideale per questo sevizio.
Lo sviluppo dell'algoritmo trova la blockchain ottimale, per essere integrata nel sistema IntegrityKEY per essere utilizzato per migliorare il servizio. In tal caso in questo trattato si ha una perlustrazione di idea di progetto di questo algoritmo.
Il progetto esplorativo è stato sviluppato in quattro fasi, dove in ognuna di esse viene descritta nei seguenti capitoli:

1. *fase I* : Predisporre django sulla repository su github. Per la quale mi è stato consigliato l'utilizzo pycharm versione per studenti che predispone l'intero progetto all'utilizzo di django virtual env effettuando la prima commint. *Questa fase è descritta in questo capitolo nella sezione (Predisposizione Progetto).*
2. *fase II* : Studio preliminare dataset necessario per l'algoritmo europa finder e progettazione data base da utilizzare su django (identificazione dei key factor di distinzione tra blockchain e generare su di loro un db compatibile con django). Questa fase è descritta nei capitoli *Blockchain*, *Database*.
3. *fase III* : Generazione dei valori pseudo casuali simili al caso reale per la creazione del dataset (attraverso il completamento del db di django). Questa fase è descritta nel *capitolo Database*.
4. *fase IV* : Identificazione librerie necessarie per lo sviluppo del progetto (per ricerca operativa o anche per gli altri settori dello sviluppo) e lo sviluppo di un algoritmo testarolo in vare configurazioni dei dati caricati alla fase III. Questa fase è descritta nei capitoli *ricerca operativa*, *algoritmo*,*test*.

1.2 Predisposizione Progetto

Per la predisposizione del progetto programmato in python con un virtual environment, dove utilizzando il web framework django. Esso messo su un repository su github per tracciare il lavoro fatto durante il tirocinio con IntegrityKEY.

1.2.1 virtual env

In python il vitual eviroment è uno strumento che contiene una particolare installazione di python con la sua libreria standard e moduli aggiuntivi che dipendo da tipo di progetto, con generalmente un gestore di pacchetti (il più comune 'pip'). Si utilizza questo strumento per la compatibilità con diversi sistemi operativi e/o per isolare i

pacchetti dall'ambente base. Nell'IDE (integrated development environment) pycharm è una della suit JetBrains nella versione per studenti un'applicazione che fornisce vari strumenti per lo sviluppo software per linguaggio python. Il virtual environment è usato da pycharm nella configurazione del progetto come progetto 'django' e in più si possono aggiungere anche altre librerie in python, in django si possono utilizzare anche plugin interni di terze parti per espandergli le funzionalità.

1.2.2 Web framework Djngo

Django è un web-framework con architettura MTV (model-template-visual), utilizzato per sviluppare in modo agile applicazione web. All'interno del framework[1] utilizza vari tipi di Database chiamati 'engine', dove quello di default utilizza il database in SQLite. In questo progetto sto utilizzando database di default sia per i dati pseudo-casuali, che per quello di test che è una copia specchiata del database usato.
La sua struttura interna del progetto è formata:

- - 'manage.py' viene utilizzato per le task amministrative per il controllo del server di django come per esempio creare applicazioni, avviare il progetto, funzioni di utilità per il database o fare avviare i test all'interno del progetto django.
- Nella cartella con nome del progetto django ci sono:
  - Un file dei settaggi del progetto di django 'setting.py' e 'url.py' dove sono esposti gli url del progetto djngo.
  - 'wsgi.py' che è getaway dell'interfaccia del web server per le proprie applicazioni, invece il file 'asgi.py' è la stessa cosa dell'altro tranne che in un tread asincrono.
- Nelle cartelle delle applicazioni ci si costruisce un sito o un server per il back-end dividendolo in parti specializzate come ad esempio una parte per database ed una per il processo e una per la sicurezza e infine per le chiamate 'http' o parte di codice html. nelle applicazioni si ha i file:
  - 'models.py' per il modelli fatti in postgresql o in altri framework per il database
  - 'admins.py' per l'accesso in amministrazione al frame-work.
  - 'test.py' dove si possono creare test e infine.
  - 'view.py' per le chiamate alla propria applicazione.
  - 'url.py' dove sono esposti gli url della aplicazione.
  - 'form.py' utilizzato per la creazione web form.

1.2.3 architettura

Djngo utilizza il pattern di tipo *MVC* , ma nella documentazione ufficiale viene descritto MVT[6] per evidenziare il tipo di implementazione del livello di presentazione dei dati. Se andiamo ad approfondire la differenza tra i due pattern:

Model-Visual-Controller (MVC)

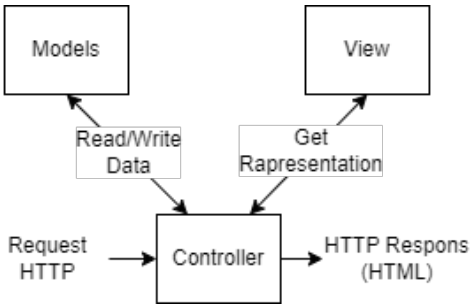


Figura 1.1: Model-Visual-Controller

- Model = rappresenta i dati e la logica interagito con un database;
- View = ricopre la funzione di rappresentare i dati;
- Controller = gestisce il flusso di richieste, agendo come intermediario tra Model e View;

Model-Visual-Template (MVT[6])

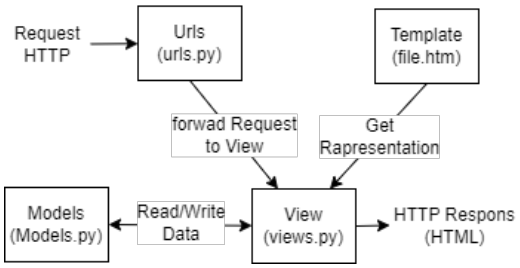


Figura 1.2: Model-Visual-Template

- Model = similmente al MVC rappresenta i dati e la logica interagato con un database;
- View = agisce più come un controller, gestendo le richieste e inviando risposte oltre quello di rappresentare i dati;
- Template = si occupa solamente di rappresentare i dati;
- Urls = controlla che al rischista 'http' se è presente nella 'mappa' al 'urls' disponibili;

1.2.4 Architettura utilizzata da Django

Django utilizza l'architettura MVT[6], in breve spiego il processo per i vari passaggi del funzionale della sua architettura.
Prima viene fatto rischista 'http' se è presente nella 'mappa' al 'urls' come risorsa. Dopo la richiesta viene passata 'View', dove viene gestita producendo una risposta 'http' o tramite un oggetto strutturato che contiene i dati o infine risposta 'http' della cui formattazione si occuperanno i template (DTL=Django Template Linguage). Nel frattempo, la logica per accedere e inserire ai dati in tabelle nel database se ne occupa il 'model', grazie utilizzo della tecnica del ORM[2] integrato.
ORM nel caso dell'utilizzo in 'django' per modellare il database utilizzando in python le classi, senza scrivere querry in SQL e addirittura creare relazioni tra tabelle.

[1]Framework = In informatica e specificamente nello sviluppo software, un framework è un'architettura logica di supporto (spesso un'implementazione logica di un particolare design pattern) sulla quale un software può essere progettato e realizzato, spesso facilitandone lo sviluppo da parte del programmatore.

[2]ORM = l'Object-Relational Mapping relazionale (ORM, O/RM e strumento di mappatura O/R) è una tecnica di programmazione che favorisce favorisce l'integrazione di sistemi software connessi al paradigma della programmazione orientata agli oggetti (OOP).

Capitolo 2: block chain

In questo capitolo approfondiremo dove si basa, il concetto, la sua struttura, e i suoi componenti della tecnologia della *blockchain*. La blockchain è una sottofamiglia della tecnologia di registro distribuito (DTL o Distributed Ledger Technology), essa rappresenta (anche) una soluzione per l'implementazione asset digitali unici. In casi reali è utilizzato per gestione creazione transizione o servizi o asset all'interno della blockchain. in questo capitolo la maggior parte delle informazioni sulla blockchain è tratta BC-def-funziona-applicazioni-oggi[1].

2.1 Ledger

Si parte dal concetto ledger (Libro Mastro o registro) è realizzato con un database, può essere di diverso tipo come: *Centralized Ledger* (Libro Mastro centralizzato o registro centralizzato), *Decentralized Ledger Centralized* Ledger (Libro Mastro decentralizzato o registro decentralizzato), *Distributed Ledger* (Libro Mastro distribuito o registro Distribuito) La blockchain è una delle realizzazioni del Distributed Ledger, come evoluzione dal Centralized Ledger e Decentralized Ledger.

Centralized Ledger
Il Centralized Ledger è rappresentato dalla logica centralizzata con il rapporto rigido centralizzato Uno-A-Tanti, dove viene gestito tutto da una struttura o autorità o sistema centralizzato di riferimento. In questo modello la fiducia si basa su autorità e nell'autorevolezza del soggetto o sistema che rappresenta il Centro dell'organizzazione.

Decentralized Ledger
Il Decentralized Ledger rappresenta la logica della centralizzazione a livello di nodi locali con satelliti. Ogni nodo locale viene gestito sulla base del modello Uno-A-Tanti, mantenendo una struttura centralizzata su scala ridotta. In questo modello la fiducia è delegata ad il soggetto centrale più vicino mantenendo centralizzato. Le organizzazioni basate su Decentralized Ledger stabiliscono una governance che delibera su strutture organizzative di natura centralizzata.

Distributed Ledger
Il Distributed Ledger è differente dagli altri, utilizzando una logica distribuita senza autorità centrale, dove la logica della governance è strutturata con la fiducia tra tutti i soggetti. Le decisioni sono prese attraverso un meccanismo di consenso, che richiede l'accordo di tutti i partecipanti e nessun soggetto si può imporre. Questa parte è spiegata più in dettaglio in "un sistema DLT di tipo blockchain".

2.2 Definizioni blockchain

La blockchain è una sottofamiglia della tecnologia registro distribuito (DTL o Distributed Ledger Technology) ed è organizzato come una catena di blocchi contenenti le transazioni e la cui validazione è incaricata a un meccanismo di

consenso. Il registro viene distribuito su tutti i nodi della rete. La blockchain può diversi tipi di blockchain Permissionless Ledger o Permissioned Ledge, con possibilità di avare una versione ibrida tra le due.
La blockchain andando a considerare la sua parte funzionale permette di gestire un database in modo distribuito. Invece, la parte operativa permette di gestire in termini di verifica e di autorizzazione senza che sia necessaria una autorità centrale. La gestione dei dati passa tramite queste operazioni:

- L'aggiornamento dei dati con la collaborazione dei partecipanti alla Rete.
- Con la possibilità di avere dati condivisi, accessibili, distribuiti che dipende dal tipo di blockchain.

Le principali caratteristiche delle tecnologie blockchain sono l'immutabilità del registro, la trasparenza, tracciabilità delle transazioni (marca temporale) e la sicurezza basata su tecniche crittografiche.
caratteristiche principali blockchain sono:

- Affidabilità: la blockchain è affidabile e sicura visto che non è governata da un'entità centrale, ma è un sistema distribuito.
- Trasparenza: Le transazioni effettuate sulla blockchain sono visibili a qualunque dei partecipanti, garantendo trasparenza nelle operazioni.
- Convenienza: Le transazioni sulla blockchain sono più convenienti rispetto a quelle tradizionali, perché in assenza di intermediari.
- Solidità: le informazioni contenute nella blockchain sono tutte più solide e credibili in quanto rimangono immutate dalla loro prima inclusione.
- Irrevocabilità: le transazioni nella blockchain sono irrevocabili, poiché assicura che non si ha possibilità di modifica o annullamento.
- Digitalità: La blockchain è una tecnologia digitale, la quale è possibile trasformare in formato digitale diversi ambiti applicativi.

2.2.1 Blockchain come database

Blockchain come database di transazioni
La Blockchain è una tecnologia che consente la creazione e gestione di un grande database distribuito per la gestione di transazioni condivisibili tra più nodi di una rete, in modo più specifico è un database strutturato in Blocchi che racchiudono più transazioni alloro volta collegati tra loro in una rete cosicché che ogni transazione avviata sulla rete debba essere validata dalla rete stessa mediazione il consenso distribuito.
Ciascun blocco è per l'appunto anche un archivio per tutte le transazioni, possono essere modificate solo con l'approvazione dei nodi della rete. Le transazioni possono essere considerate immutabili (salvo attraverso la riproposta e la ri-autorizzazione delle stesse dell'intera rete), per questo motivo il concetto di immutabilità.
I nodi della rete verificano, raggruppano e trasmettono le transazioni, dove in alcuni casi minano per cercare hash unici per creare blocchi dove caricano le transazioni. Quando c'è consenso, i nodi aggiornano le loro copie locali del libro mastro e inseriscono il blocco nella Blockchain. *Il consenso distribuito*, un processo in cui i nodi di una rete peer-to-peer raggiungono un accordo su una particolare informazione,

consente il livello di consenso.

La blockchain come database per transazioni crittografate
La tecnologia Blockchain si basa su una rete Peer-to-Peer (P2P) di computer per elaborare e registrare le transazioni crittografate in un libro mastro condiviso (database distribuito), che permette di ridefinire e reimpostare il modo in cui creiamo, otteniamo e scambiamo valore.

2.2.2 Rete decentralizzata di tipo peer-to-peer

L'elaborazione o la rete peer-to-peer (P2P) è un'architettura di applicazioni distribuita che partiziona attività o carichi di lavoro tra peer. Dove sono tutti sullo stesso livello, creando una rete di nodi peer-to-peer. I peer condividono le proprie risorse, come potenza di elaborazione, spazio di archiviazione e larghezza di banda di rete, con gli altri peer.

2.2.3 Asset Digitali Unici

l'asset digitale è semplicemente un qualunque contenuto archiviato digitalmente, in questo caso egli possono essere transazioni, servizi e prodotti tramite NFT.
Il valore dell'asset digitale è raffigurato dai prodotti o servizi o dalle transazioni se egli resterà unico e non sarà possibile duplicarlo, bensì il digitale permette di gestire in modo molto più efficiente scambi e transazioni e anche i servizi.
L'importanza degli Asset Digitali Unici è solo ed esclusivamente se si garantisce la capacità di evitare duplicazioni, ovvero solo se si garantire **l'unicità del asset**, Similmente come accade nel modo reale.

2.3 Componenti della blockchain

La blockchain è un protocollo di comunicazione basto su un database distribuito, dove i dati sono salvati su più macchine chiamati nodi.
La blockchain strutturalmente la si definisce, come una sequenza di blocchi che registrano un insieme di transazioni validate e correlate da un Marcatore Temporale (Timestamp). Qualsiasi blocco include l'hash che identifica il blocco univocamente e che consente il collegamento con il blocco precedente grazie la sua identità univoca.
La transazione contiene invece informazioni relative all'indirizzo mittente e del ricevete, le caratteristiche della transazione e la firma crittografica che garantisce sicurezza e autenticità della transazione.
I componenti fondamentali della blockchain: Nodo, Transazione, Blocco, blockchain (Ledger), Hash.

Traslazione

Componenti della transazione di una blockchain sono:

- Sender, cioè la persona che avvia il processo di transazione.
- Transaction, ossia la transazione come processo.
- Receiver, la persona che ottiene la transazione.

Le transazioni possono essere a modello de tipo Microtransazioni composte da una o più sub transazioni per ogni persona, che può essere di entrate o uscita (transazioni

bitcoin[9]) , o a Transazioni interne composte da una o più transazioni intermedie di vario tipo (o versioni)(transazioni ethereum[10]).
I Sender/ Receiver, cioè le persone che partecipano alla transizione diversi tipi di configurazione della transazione:

- Singolo Sender/ Receiver con singolo Receiver/ Sender;
- Singolo Sender/ Receiver con multipli Receiver/ Sender;
- Multipli Sender e Receiver;

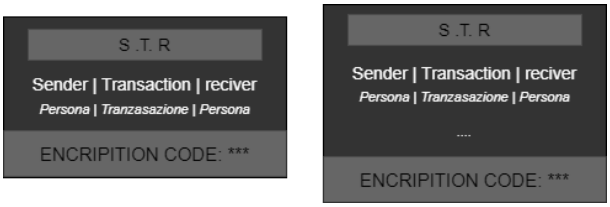


Figura 2.1: traslazione

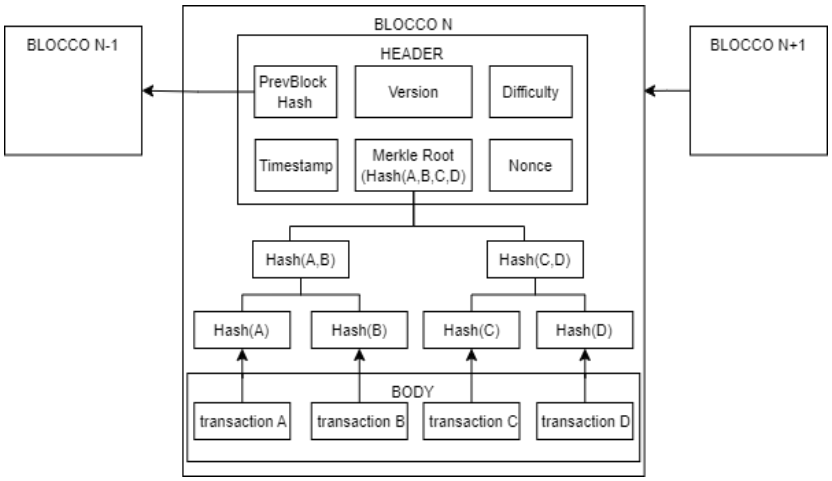


Figura 2.2: blockchain

Il Blocco
Blocco di transazioni: Il Blocco è un registro di transazioni verificate (blocchi bitcoin[3] , blocchi ethereum[2], ...), dove è costituito da intenzione del blocco e corpo del blocco (descritto in su151813401[7]). Il blocco header include generalmente:

- Version: utilizzato per mantenere traccia degli aggiornamenti software/protocollo.
- PrevBlock Hash: registra il valore di hash del blocco precedente che è stato generato in modo univoco e irreversibile per garantire il collegamento tra i blocchi.
- Merkle Root: registra il valore hash della radice dell'albero Merkle di questo blocco.
- Timestamp: viene registrato il timestamp di creazione del blocco per garantire l'ordine cronologico dei blocchi e consentendo la tracciabilità delle transazioni.

- Difficulty: il coefficiente di difficoltà da risolvere per il blocco corrente.
- Nonce: il valore calcolato dal nodo in base alla potenza di calcolo, generalmente è inferiore al target.

Il corpo del blocco contiene il registro di transazioni verificate. Ogni dato sulla transazione è caratterizzato con una firma digitale, tale meccanismo è usufruito per garantire la sicurezza dei dati nel blocco. Il body del blocco include generalmente quanto segue:

- Num TransactionsBytes: registra il numero di byte occupati dalle transazioni.
- Num Transactions: registra il numero di transazioni nel blocco. Se utilizza il modello a microtransazioni ha il numero di transazioni sia in entrata che in uscita.
- Transactions: registro di più transazione nel blocco.

BlockChain
Blockchain (Ledger) è il registro dove vengono registrate pubblico in modo immutabile tutte le transazioni effettuate in modo ordinato e sequenziale. Il Ledger è costituito dal complesso dei blocchi che sono collegati tra di loro tramite una funzione di crittografia e grazie all'uso di hash.

Nodo
I nodi sono i partecipanti alla blockchain e sono costituiti fisicamente dai server di ciascun partecipante.

Hash
Hash è una operazione Non Invertibile che consente di mappare una stringa di testo e/o numerica di lunghezza variabile in una stringa unica ed univoca di lunghezza determinata.
L'hash viene identificato in maniera univoca e sicura ciascun blocco, che deve avere bassissime probabilità di risalire al testo che lo ha generato. Le funzioni hash possono essere come MD5, BLAKE2, SHA-1 o SHA-256.

2.3.1 L'albero di Merkle

L'albero di Merkle (parte del Merkle-proofs[5]) è una struttura di dati adoperata nella Blockchain per organizzare e verificare le transazioni all'interno di un blocco. Ogni transazione viene convertita in un hash univoco usufruendo di funzioni di hash, che poi vengono disposti in un albero binario, noto anche come albero Hash o albero di Merkle. L'albero di Merkle ha una struttura gerarchica con i nodi figli rappresentanti gli hash delle transazioni, i nodi intermedi rappresentanti gli hash delle coppie di nodi figlio. La radice dell'albero rappresenta l'hash dell'intero blocco chiamata radice di Merkle, essa è presente in ogni blocco nella Blockchain.
Ci sono svariati vantaggi nel servirsi di un albero di Merkle nella Blockchain:

1. L'albero di Merkle permette di verificare facilmente l'integrità dei dati: Qualsiasi modifica apportata a una transazione influirà l'hash del nodo superiore e di conseguenza, la radice di Merkle che rende visibile se le transazioni sono state manipolate.

2. L'albero di Merkle semplifica e riduce la verifica delle transazioni:
   (a) Gli utenti possono verificare se una transazione nota è incluso in un blocco senza dover scaricare e convalidare tutte le transazioni.
   (b) Basta seguire il percorso dell'albero di Merkle dalla radice alla transazione cercata, calcolando gli hash intermedi necessari lungo il percorso.

2.3.2 Merkle proofs

Merkle-proofs[5] consentono la verifica di una specifica transazione/contenuto in un grande struttura di dati. Nei sistemi con struttura di tipo Blockchain, ci sono due tipi di nodi:

1. (i) I nodi che contengono una storia completa di Blockchain sono chiamati "nodi completi".
2. (ii) E altri nodi sono chiamati "nodi leggeri" dove include solo un elenco parziale di Blockchain, può solo convalidare le transazioni comprese in un blocco senza la necessità di scaricare l'intera Blockchain.

2.4 Le Distributed Ledger Technology

la criptografia e lo sviluppo di algoritmi di controllo e verifica dei dati che aprono le porte a quelli che diventano i Distributed Ledger Technology.

> Con i Distributed Ledger Technology si entra nell'ambito dei Database Distribuiti, ovvero di Ledger (Libri Mastro) che possono essere aggiornati, gestiti, controllati e coordinati appunto non più solo a livello centrale, ma in modo distribuito, da parte di tutti gli attori.

La blockchin è una sottofamiglia della tecnologia di registro distribuito (Distributed Ledger Technology – DLT), costruito in modo che sia immutabile su tutte le transazioni e con operazioni non replicabili grazie all'utilizzo della Marca Temporale. Essa utilizza rete Algoritmi e rete peer-to-peer alla base delle DLT, per il raggiungimento del consenso distribuito delle DLT per il processo di validazione delle transazioni.

Il Grande Libro Mastro della blockchain

La catena di Blocchi è la (il libro mastro o registro della) blockchain che è posseduto da tutti i nodi che partecipano. Ogni nuova transizione che vien registrata assieme a tante altre viene aggiunta ad un blocco, che si giunge ad una catena.

L'immutabilità della blockchain

L'immutabilità è l'altro importante valore della blockchain che naturalmente riguarda sicurezza dei dati. L'immutabilità della blockchain è data dall'utilizzo dal registro distribuito (Distributed Ledger), essendo che le transizioni possono essere modificate solo con l'approvazione dei nodi della rete. L'immutabilità varia per tipi di blockchain: Permissionless o pubbliche , Permissioned o private.

Marca Temporale
la marca temporale o Timestamp è uno degli attributi più importanti della blockchain, mantiene la sicurezza impedisce che l'operazione una volta compiuta, non venga modificata o annullata. Essa consiste da una sequenza specifica di caratteri che identificano in modo univoco, indelebile e immutabile una data e/o un orario per fissare e accertare l'effettivo avvenimento di un certo evento.

Algoritmi e rete peer-to-peer alla base delle DLT
Le Distributed Ledger Technology (DLT) richiedono di una rete peer-to-peer e gli algoritmi in grado di gestire la raccolta del consenso e la approvazione di operazioni basate appunto sul raggiungimento di un consenso. La differenza tra Distributed Ledger Technology (DLT) di tipo Pubblico e di tipo Privato è data dai modelli di gestione del Consenso.

il Consenso Distribuito
Il processo di validazione della blockchain prevede una fase di verifica e di approvazione basata su risorse di calcolo che vengono messe a disposizione dai partecipanti alla blockchain e che sono finalizzate alla risoluzione di problemi complessi o puzzle crittografici e che permettono di disporre di un Consenso Distribuito e non più di un consenso basato su un intermediario terzo.

2.5 Algoritmi di consenso

I meccanismi di consenso che sono parte del processo di validazione e creazione blocchi sono citati alcuni dei principali proof of work, proof of stake preso da su151813401[7]. Nella blockchain le entità di accodamento sono denominate minatori o validatori.

2.5.1 Proof of work (PoW)

l Proof-of-Work (PoW) è un algoritmo di consenso utilizzato nelle blockchain per garantire la sicurezza e la coerenza dei dati attraverso una competizione basata sulla potenza computazionale. I nodi della rete, chiamati "miners", cercano di risolvere un complesso problema matematico generando un valore casuale, detto "nonce", e calcolando il valore di hash del blocco. Il primo miner a trovare la soluzione corretta che, ottiene il diritto di creare un nuovo blocco di transazioni e aggiungerlo alla blockchain. la soluzione corretta è Se il valore di hash ottenuto è inferiore a un valore target predefinito.
Il livello di difficoltà del problema matematico viene automaticamente regolato per mantenere un ritmo di creazione dei blocchi stabile, prevendo sia la concentrazione della potenza di calcolo in poche mani e che difficoltà che disincentiverebbe i miner.
Contro = Richiede un grande consumo di energia e potenza di calcolo; quindi, Limita la scalabilità a causa di basso volume di transazioni verificate e latenza.
Pro = In cambio del loro contributo, i miners vengono ricompensati unità di criptovaluta o token, incentivando così la loro partecipazione e contribuendo alla sicurezza complessiva della blockchain.

2.5.2 proof of stake (PoS)

Il proof of stake (PoS) è un tipo di algoritmo di consenso in cui il blocco successivo viene scelto in base alla stake del miner. i nodi della rete possono diventare candidati alla validazione di nuovi blocchi puntando una certa quantità stake(unità di criptovaluta o token). La selezione dei i nodi per la validazione dei blocchi in base a diversi fattori, tra cui l'ammontare dello stake, al tempo di detenzione e un elemento di randomizzazione per garantire equità. Più a lungo un nodo è stato un validatore, maggiori sono le possibilità di essere selezionato come nuovo validatore. Dopo aver ottenuto il diritto di validazione, lo stake del nodo viene azzerato, e inizia un nuovo ciclo di accumulo.
Contro = Nodi con maggiori quantità di stake hanno maggiori probabilità di essere selezionati come validatori, aumentando il rischio di centralizzazione.
Pro = Il PoS offre vantaggi significativi rispetto al PoW, come la riduzione del tempo di blocco e del consumo di risorse, migliorando l'efficienza del consenso.

2.6 Soluzioni di scalabilità blockchain

le "blockchain livello 1" sono è usato la rete di base per andare a scalare la blockchain utilizzando gli algoritmi di consenso (protocolli di consenso). Quando la blockchain di "blockchain livello 1" è molto grande in volume aumenta le tariffe e i tempi di elaborazione estesi sono l'effetto dell'aumento della potenza di calcolo necessaria per risolvere e aggiungere blocchi a una blockchain.
Una soluzione risolvere questo problema è sovrapponendo assieme "blockchain livello 1" e il "blockchain livello 2" per aumentare volume delle transizioni (e smart contract) gestite della Rete blockchain. Questo risultato a un altro effetto positivo migliora la sua sicurezza e per avere un controllo sulle traslazioni. in questa parte sono spiegate alcune metodologie per implementarle.
Mainchain detta anche blockchain principale "blockchain livello 1" nelle metodologie spiegate nella *channel state* dove "off-chain" è la "blockchain livello 2", nella sidechain dove *sidechain* è la "blockchain livello 2".

2.6.1 channel state

l primo tipo di soluzione di scalabilità è il *channel state*[4], che permettono di creare un canale peer-to-peer tra due parti. Il primo tipo di soluzione di ridimensionamento sono i canali, che concedono di creare un canale peer-to-peer tra due parti, dove possono scambiare un numero illimitato di transazioni (sul sul off-chain) inviando solo due transazioni della mainchain .
La prima transazione che apre la connessione, alla fine un'altra transazione memorizzata sul Mainchain è la transazione che chiude la connessione tra il Mainchain e il off-chain.
Togliendo la maggior parte delle transazioni dal Mainchain (transazioni off-chain), i canali del off-chain ottimizzano la velocità delle transazioni, tale che riducono il traffico della rete ed i costi e ritardi delle transazioni.
Benefici: Togliendo le transazioni dal Mainchain, il off-chain decongestiona la rete del Mainchain e in seguito aumenta la velocità della transazione riducendo i costi di transazione, con la eventualità di fare micropagamenti.
Limitazioni: I partecipanti devono rimanere regolarmente online e accedere alle

proprie chiavi private per le transazioni e bloccare un certo importo dei loro fondi in un portafoglio multi-firma per poter aprire il canale di pagamento.

2.6.2 sidechain

Le sidechain[4] sono Blockchain separate che sono collegate alla Blockchain principale tramite un peg a due vie per aiutare a elaborare alcuni dei dati dalla Blockchain principale.
Ognuna di queste catene ha il proprio insieme di regole, funzionalità e scopi. Le sidechain sono responsabili della propria sicurezza e meccanismi di consenso che bloccano i parametri, essi hanno anche bisogno dei propri nodi per convalidare le transazioni e creare un blocco.

1. Two-way peg (2WP): Il 2WP funge da intermediario per facilitare il trasferimento di asset o transazioni dalla mainchain alla sidechain e viceversa.
2. Presenza della terza parte: la terza parte/autorità è responsabile delle funzioni di blocco e rilascio tra sidechain e mainchain.
3. Prove SPV: La prova SPV (Simple Payment Verification) è un modo per dimostrare crittograficamente che le transazioni sono bloccate sulla catena principale per il loro utilizzo sulla catena laterale.

Benefici: Le sidechain aumentano il throughput delle transazioni portando via la maggior parte delle transazioni nella loro sidechain ed elaborando le transazioni a una velocità molto più elevata. Le blockchain pubbliche sono trasparenti e aperte alla verifica da qualsiasi partecipante, ma non utili per le aziende per il rischio di rivelare molte informazioni specifiche dell'azienda.
Sfide: La maggior parte delle sidechain fornisce un equilibrio tra velocità e sicurezza perché sono un po' più centralizzate rispetto alla mainchain.

2.7 Permissionless o Permissioned Ledger

Esistono due tipi di blockchain principali: Blockchain Permissionless o Permissionless Ledger (blockchain Pubbliche) e Blockchain Permissioned o Permissioned Ledger (blockchain Private). Ma se si considera via di mezzo tra le due, si può aggiungere un'altra Blockchain IBRIDE.

Blockchain Permissionless
La Blockchain Permissionless si adatta a scopi open source in tal caso le regolo di accesso al network è totalmente libero per chiunque intenda partecipare. A ciascuno è permesso di contribuire all'aggiornamento dei dati sul Ledger e di usufruire per le copie immutabili ciascuna operazione in qualità di partecipante, tutto ciò quanto viene approvato grazie al consenso.
Il controllo è di tipo distribuito e riguarda tutti i partecipanti, si può indurre che tutti sono chiamati a validare le transazioni. Tale modello di blockchain impedisce qualsiasi forma di censura per il motivo che, se una transazione ha avuto il consenso necessario tra tutti i nodi partecipanti è viene registrata in modo permanente.

Blockchain Permissioned
La Blockchain Permissioned sono sicura, affidabile, adatta per quelle applicazioni aziendali o organizzazioni che necessitano di garantire l'autenticazione dei partecipanti.
Regolo di accesso e controllo Concentrato, quindi L'accesso al network e le attività di controllo e validazione sono limitati un gruppo ristretto di partecipanti che seguono delle specifiche linee guida (Governance della Blockchain).
Le Permissioned Ledger possono definire speciali regole per l'accesso e la visibilità di tutti i dati, in tal modo introducendo un concetto di Governance e di definizione di regole di comportamento (Governance della Blockchain).

Blockchain IBRIDE
La Blockchain IBRIDE si adatta a scopi aziendali e pubblici in tal caso le regolo di accesso al network è limitato a un gruppo ristretto di partecipanti che sono riconosciuti autenticati. Ma utilizza il controllo Distribuito, La validazione affidata tutti i nodi e tutti concorrono alla sicurezza complessiva della Blockchain.

2.8 Fork

I Fork sono strumenti usufruiti dalle network blockchain per migliorare le prestazioni della blockchain e/o per gestire il protocollo, che si dividono in Soft Fork e Hard Fork.

2.8.1 Soft Fork

Il Soft Fork si produce una versione aggiornata del protocollo blockchain compatibile con le versioni precedenti per una trazione reversibile, per cui consentire la partecipazione alla rete blockchain pure per quei nodi che per varie ragioni optano di non effettuare l'aggiornamento.

2.8.2 Hard Fork

L'Hard Fork stabilire un cambiamento irreversibile ed esige ai partecipanti della blockchain di effettuare necessariamente l'aggiornamento, in tal modo sono prodotte nuove blockchain. Gli Hard Fork possono essere di diverso di tipo:

- Planned: ovvero pianificati e programmati con il consenso dai partecipanti alla community;
- Contentious: ovvero che non riescono a trovare il consenso della community;

Hard Fork Planned
Nel caso di Hard Fork Planned è pianificato il cambiamento del protocollo con consenso dai partecipanti alla community, la quale non porta alla divisione della blockchain dove le regole vengono aggiornate in forma di continuità.

Hard Fork Contentious
Nel caso degli Hard Fork Contentious non trova un accordo il cambiamento proposto al protocollo all'interno della community sviluppandosi una scissione della blockchain, portano alla creazione di una nuova blockchain.

2.9 token

Un token è un asset digitale basato sulla blockchain che può essere scambiato tra due parti senza che sia necessaria l'azione di un intermediario. Funzionando come un

insieme di informazioni digitali che è in grado di conferire un diritto di proprietà su di sé, il possessore registrato su una blockchain possa trasferirlo tramite un protocollo.
Il token può eventualmente incorporare anche altri diritti addizionali che nel caso sono governati da un sistema di smart contracts.

2.9.1 Classificazione in base ai diritti

In particolare, è importante focalizzare l'attenzione su tre diverse tipologie di token determinati dal tipo di diritti gestiti dai token stessi

1. Token di classe 1 – il token ha l'aspetto di una moneta reale, non fornisce alcuna controparte e può essere trasferito tramite transazioni su blockchain.
2. Token di classe 2 – i token attribuiscono ai proprietari dei diritti che possono essere esercitati nei confronti dell'emittente dei token o di terze parti. Si tratta di token che sono in grado di permettere l'esercizio di diritti verso controparti specifiche
   (a) Token per smart contract relativi alla gestione di pagamenti futuri – Tali token conferiscono il diritto a ricevere pagamenti futuri basati su condizioni contrattuali predefinite che vengono gestite automaticamente dal token;
   (b) Token come asset: Tali token rappresentano diritti di proprietà su un determinato asset (sia materiale sia immateriale) in caso, possono rappresentare quote di partecipazione in un'entità giuridica emittente o in entità terze;
   (c) Token utilizzati per pagamenti "standardizzati": dove una persona possiede il diritto di ricevere un pagamento per un importo specifico preciso;
   (d) Token per la gestione di prestazione di servizi: l titolare di tali token ha il diritto di ricevere una prestazione specifica o anche un bene da parte dell'emittente dei token o di un terzo, per cui è stato stipulato un accordo commerciale;
3. Token di classe 3 – i token possono avere una funzione mista. Sono token che ritraggono diritti di comproprietà, oppure che rappresentano una proprietà ma conferiscono anche diritti diversi;

2.9.2 smart contract

Uno smart contract è un codice informatico o un protocollo di transazione integrato nella blockchain per semplificare, verificare e negoziare accordi contrattuali e ridurre le eccezioni fraudolente e accidentali. Questi contratti svolgono seguendo una sequenza di condizioni accettate dagli utenti, quando tali condizioni vengono adempite infine vengono eseguiti automaticamente.

2.9.2.1NFT

gli NFT (tratto da borsa-italiana NFT[8]) sono un acronimo di Non-Fungible Token (token non fungibili), dove è una delle applicazioni della finanza decentralizzata[1].
Acquistare un NFT non implica l'acquisizione della proprietà dell'opera, anzi la possibilità di attestare un diritto su quell'opera, tramite uno smart contract che effettua automaticamente un contratto che viene registrato permanentemente sulla blockchain.

[1]Finanza decentralizzata (in inglese, Decentralised Finance o DeFi): Un insieme di servizi e processi che vengono automatizzati grazie all'ausilio di contratti intelligenti (smart contract) e senza la presenza di intermediari nella gestione del diritto di proprietà.

Capitolo 3: ricerca operativa

Il problema della scelta ottimale della blockchain in questo trattato è utilizzata la Ricerca Operativa, che ha come oggetto lo studio e la messa a punto di metodologie per la soluzione di problemi matematici applicati al modo reale. Nel caso trattato utilizzammo una su tipologia di problemi, quella dei decisionali. In questo capitolo nella fino alla sezione la modellazione del problema utilizza tale risorsa AppRCunipi[22], ma per il problema matematico, B&B ed variabili decisionali utilizza AppRCuniroma1[26].

3.1 introduzione

Un processo decisionale può, in modo schematico, essere decomposto nelle seguenti fasi:

(1) Individuazione del problema: esame della situazione reale e raccolta delle informazioni;
(2) Analisi della realtà e raccolta dei dati: individuando le variabili controllabili e non e la scelta della funzione;
(3) Costruzione del modello (analitico): mediante il formalismo matematico adeguato;
(4) Determinazione di una o più soluzioni: si determina una o più soluzioni del modello;
(5) Analisi dei risultati ottenuti: tramite analisi e la verifica dei risultati ottenuti;

Nella trattazione il punto (*1*) la parte del "*esame della situazione reale*" mi è stato fornito, per la quale è l'argomento della tesi. Invece nell'esperienza fatta nel tirocinio ho trattato nella la seconda parte del punto (*1*), cioè "*raccolta delle informazioni*".
La concettualizzazione fatta del 2 capitolo (blockchain) e la creazione del database nel capitolo 3 (database), vado a trattare il una parte del (*2*) punto l'altra parte venne fatta in questo capitolo nella risoluzione del ksnap(problema dello zaino) nei vari test.
Nella trattazione del problema utilizziamo alcune metodologie messe a punto nell'ambito della Ricerca Operativa, che una rigorosa trattazione matematica sui punti (*3*) e (*4*). Il modello descrive in generale, un mezzo di strumenti di tipo logico-matematico ai fini del processo decisionale.
La costruzione di un modello (analitico) di un sistema reale è utile nello studio di sistemi organizzativi e gestionali e per ottimizzazione in tante discipline, oltre a quella di progettazione industriale e sistemi altamente complessi quali quelli informatici.
Infine, il (*6*) punto viene visto nell'ultimo capitolo (risultati).

3.2 problemi

In ricerca operativa esistono varie branchie classificate, in questa trattazione vado ad utilizzare una tra le più importanti quella della ottimizzazione. la quale è la tipologia appartiene il nostro caso di tipo di problma, zaino multidimensionale (multi kpsnap) e in un caso anche multiplo .

Nello specifico il nostro caso che tratto è un problema di ottimizzazione, che possono essere classificati come: la classe dei problemi polinomiali (P) e con la classe dei problemi NP-ardui (NP-completi quando si parli di problemi decisionali).

3.2.1 problema di ottimizzazione

Problema di ottimizzazione lo possiamo definire come un *problema matematico*, nei quali si desidera minimizzare o massimizzare una funzione con un certo numero di *parametri* e *variabili* che sono vincolate da un *numero finito di disequazioni.*
In un problema di ottimizzazione, sull'insieme ammissibile S viene definita una funzione obiettivo che fornisce il costo o il beneficio associato ad ogni soluzione.
La Funzione $c : \mathbb{R}^n \to \mathbb{R}$ dove $S \subseteq \mathbb{R}^n$ il problema di ottimizzazione può essere messo in forma:

Funzione di minimo
Generico problema

$$(\mathrm{P}) \begin{cases} \min c(x) \\ x \in S \end{cases} \qquad z(P) = \min\{c(x) : x \in S\} \tag{3.1}$$

Generico problema equivalente

$$(P_l) \begin{cases} -\max c(x) \\ x \in S \end{cases} \quad \text{dove } z(P) = z(P_l) \tag{3.2}$$

Funzione di massimo
Generico problema

$$(\mathrm{P}) \begin{cases} \max c(x) \\ x \in S \end{cases} \qquad z(P) = \max\{c(x) : x \in S\} \tag{3.3}$$

Generico problema equivalente

$$(P_l) \begin{cases} -\min c(x) \\ x \in S \end{cases} \quad \text{dove } z(P) = z(P_l) \tag{3.4}$$

Il valore ottimo del problema sì può identificare con una delle tante soluzioni ammissibili, dove una soluzione ammissibile $x^* \in S$ tale che $c(x^*) = z(P)$ è detta soluzione ottima per (P). Un problema viene definito fornendo l'insieme S delle possibili risposte o soluzioni composti dalle soluzioni ammissibili è un insieme di valori dei parametri che compongono le possibili risposte o soluzioni ammesse dal problema e la parte di risposte o soluzioni non ammesso dal problema.

3.2.2 soluzioni problema di ottimizzazione

Dato un problema di ottimizzazione (P), sono possibili quattro casi:

- Il problema è vuoto: ossia $S = \emptyset$ in questo caso si assume per convenzione $z(P) = +\infty$($-\infty$ per un problema di massimo).
- Il problema è inferiormente illimitato (superiormente illimitato per un problema di massimo): $\forall M \in \mathbb{R} \ \exists x \in S$ tale che $c(x) \leq M$($\geq M$ per un problema di massimo) questo caso il valore ottimo è $z(P) = +\infty$($-\infty$ per un problema di massimo).
- Il problema ha valore ottimo finito $-\infty < z(P) < \infty$ ed ammette:
  - Soluzione non ottima: $\nexists x \in F$ tale che $c(x) = z(P)$.
  - Soluzione ottima: $\exists x \in S$ tale che $c(x) = z(P)$.

3.2.3 problema matematico

Le funzioni $g_j : \mathbb{R}^m \to \mathbb{R} \ \ j = 1, \cdots, m$ ed m scalari $b_j \in \mathbb{R}^m \ \ j = 1, \cdots, m$ si esprime S nella forma dell'insieme delle soluzioni si può scrivere:
per il problema di massimo (3.5) $S = \{x \in \mathbb{R}^n : g_j(x) \leq b_j \ j = 1, \cdots, m\}$, invece per il problema di minimo (3.6) $S = \{x \in \mathbb{R}^n : g_j(x) \geq b_j \ j = 1, \cdots, m\}$.

$$\begin{cases} \max \quad C(x_1, \cdots, x_n) \\ g_i(x_1, \cdots, x_n) \leq b_i \quad i = 1, \cdots, n \end{cases} \tag{3.5}$$

$$\begin{cases} min \quad C(x_1, \cdots, x_n) \\ g_i(x_1, \cdots, x_n) \ \geq b_i \ \ i = 1, \cdots, n \end{cases} \tag{3.6}$$

Un problema di questo tipo viene chiamato problema di Programmazione Matematica. I punti dell'insieme ammissibile di questo tipo di problemi sono quelli per i quali tutti i vincoli sono soddisfatti, cioè, tutti quei punti x tali che tutte le disuguaglianze $g_j(x) \leq b_j$ $(g_j(x) \ \geq b_j \ )$ $j = 1, \cdots, m$ sono verificate.

3.3 Algoritmo

L'utilizzo nella pratica dei problemi di ottimizzazione è collegato allo sviluppo d'algoritmo (o processi di calcolo) con la capacità di risolvere efficacemente le istanze. In generale, un algoritmo che risolva il problema (P) è una procedura che prende in input una qualsiasi istanza p di (P) e fornisce in output una soluzione ottima x^* per quell'istanza. Un algoritmo per (P) che determini una soluzione ottima per qualsiasi istanza viene detto algoritmo ottimo. Approssimazione viene usata con i problemi di tipo programmazione lineare intera (PLI) o programmazione lineare intera mista (PLIM).

3.3.1 Approssimazione

Gli algoritmi arrivando ad una complessità troppo elevata per essere utilizzati nella pratica, quindi necessità l'utilizzo di algoritmi euristici[11].
Gli algoritmi euristici sono algoritmi che determinano solamente una qualsiasi soluzione ammissibile, andata a valutare quanto è buona l'approssimazione nel caso superiore per un problema di minimo (inferiore per un problema di massimo).
Valore calcolato dall'algoritmi euristico della istanza può essere:

- Valore "*ottimo*" può determinare una qualsiasi soluzione ammissibile, mentre mantengo la stessa complessità del problema;
- Valore "*arbitrariamente cattiva*" se è $+\infty$ $(-\infty)$ con nessuna soluzione ammissibile anche per istanze in cui $S \neq \emptyset$, la quale è data da una maggiore complessità rispetto a quelle del problema;

3.3.2 Rilassamenti

il rilassamento è una strategia di modellazione, esso è un'approssimazione di un problema difficile da un problema vicino che è più facile da risolvere. Una soluzione del problema rilassato fornisce informazioni sul problema originale.
Rilassamento per un Problema di minimo
avendo un Problema di minimo $(P) \ \min\{c(x) : \ x \in S \subseteq \mathbb{R}^n\}$ Per un qualsiasi problema $\left(\overline{P}\right) \ \min\left\{\overline{c}(x) : \ x \in \overline{S} \subseteq \mathbb{R}^n \ \right\}$ tale che $S \subseteq \ \overline{S}; \overline{c}(x) \leq \ c(x) \ \forall x \in \mathbb{R}^n$.
Rilassamento per un problema di massimo
avendo un problema di massimo $(P) \ \max\{c(x) : \ x \in S \subseteq \mathbb{R}^n\}$ Per un qualsiasi problema $\left(\overline{P}\right) \ \max\left\{\overline{c}(x) : \ x \in \overline{S} \ \subseteq \mathbb{R}^n\right\}$ tale che $S \supseteq \ \overline{S}; \overline{c}(x) \geq \ c(x) \ \forall x \in \mathbb{R}^n$.
Proprietà:

[11] algoritmi euristici = algoritmi progettato per risolvere problemi in maniera ottimale, preciso e veloce nella esecuzione. Nel caso dove il metodo classico si ha complessità troppo elevata per trovare una soluzione esatta.

1. Una valutazione inferiore del valore ottimo di $\left(\overline{P}\right)$, avendo $x^* \in \mathbb{R}^n$ soluzione del problema originale $\exists x \in S$ tale che $c(x) \leq c(x^*)$ ($c(x) \geq c(x^*)$ problema di massimo);
2. Nel caso è possibile che sia soluzione del rilassamento sia soluzione ottima del problema, se la soluzione ottima x^* del problema (P) se $x^* \in S$ dove $\overline{c}(x^*) = c(x^*)$;

È spesso possibile definire i rilassamenti in modo che siano risolubili con algoritmi di complessità inferiore rispetto a quelli necessari per il problema originale.

3.4 Programmazione Lineare

La funzione obbiettivo è lineare se La funzione multivaribile $c : \mathbb{R}^n \to \mathbb{R}$ si dice lineare se rispetta le proprietà:

1. $\forall x, y \in \mathbb{R}^n \; c(x+y) = c(x) + c(y)$;
2. $\forall x \in \mathbb{R}^n \; , \; \alpha \in \mathbb{R} \;\; c(\alpha x) = \alpha c(x)$;

Un problema di Programmazione Lineare (PL) è un problema di ottimizzazione caratterizzato dalle seguenti proprietà:

- Numero finito di variabili n la quale $x \in \mathbb{R}^n$;
- Funzione obbiettivo lineare $z(x) = cx = \sum_{i=1}^{n} c_i x_i$, dove $c \in \mathbb{R}^n$ e il vettore dei costi (fissato);
- L'insieme ammissibile è definito da un insieme finito di m vincoli lineari del tipo $ax = b$, e/o $ax \leq b$, e/o $ax \geq b$, dove $a \in \mathbb{R}^n$, $b \in \mathbb{R}$;

Struttura di un modello di programmazione lineare

Funzione massimo	Funzione minimo
In forma max $\{cx \; : \; Ax \leq b \}$ dove $x \in \mathbb{R}^n$, $A \in \mathbb{R}^{n \times m}$, $b \in \mathbb{R}^m$, $c \in \mathbb{R}^n$	In forma min $\{cx \; : \; Ax \geq b \}$ dove $x \in \mathbb{R}^n$, $A \in \mathbb{R}^{n \times m}$, $b \in \mathbb{R}^m$, $c \in \mathbb{R}^n$
In forma lineare $\begin{cases} \max \;\; c^T x \\ Ax \;\geq\; b \end{cases}$	In forma lineare $\begin{cases} \max \;\; c^T x \\ Ax \;\geq\; b \end{cases}$

Il vettore delle variabili $x^T = \begin{bmatrix} x_1 & \cdots & x_n \end{bmatrix}$

La matrice $A = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix}$, i vettori $b = \begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix}$ ed $c = \begin{bmatrix} c_1 \\ \vdots \\ c_n \end{bmatrix}$

3.4.1 Tipi di variabili decisionali

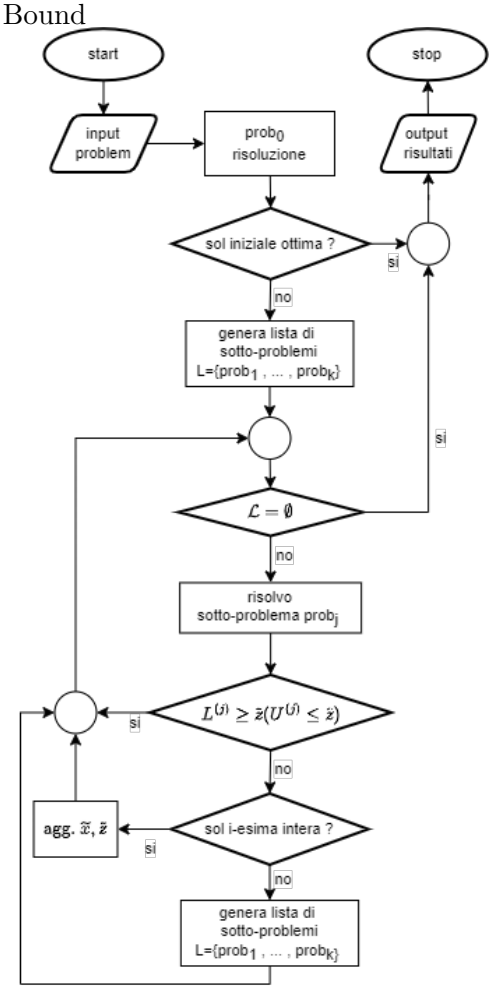
Programmazione lineare (PL): Dove Le variabili possono assumere tutti i valori reali $x \in \mathbb{R}^n$ andando a parlare di Problemi di *ottimizzazione Continua.* dove è Vincolata se $S \subset \mathbb{R}^n$,altrimenti è Non vincolata se $S = \mathbb{R}^n$;
Programmazione lineare intera (PLI) : Le variabili sono vincolate ad essere numeri interi ($x \in \mathbb{Z}^n$) vengono chiamati Problemi di *Ottimizzazione Discreta*, dove si possono distinguere all'interno due sub-classi: Programmazione a numeri interi se $S \subseteq \mathbb{Z}^n$, Ottimizzazione booleana se $S \subseteq \{0,1\}^n$;
Programmazione lineare intera mista (PLIM): È un misto di variabili vincolate ad essere numeri interi ($x \in \mathbb{Z}^n$), di tipo booleana $x \in \{0,1\}^n$ e quelle che possono assumere tutti i valori reali ($x \in \mathbb{R}^n$). In questo caso vengono chiamati *Problemi misti.*

3.5 Algoritmi enumerativi

Nei risolutori utilizzati sono basati su *metodi di ottimizzazione combinatoria*[12] per programmazione lineare intera (PLI)[13] o programmazione lineare intera mista (PLIM)[13]. I metodi in questione sono Branch and cut ed Branch and price si basa su Branch and Buond. I risolutori usati sono: CBS MIPS resolver basato su Branch and cut; SCIP resolver basato su Branch and price;

3.5.1 Branch and Buond

Figura 3.1: algoritomo Branch End Bound



Metodo Branch and Buond è un metodo combinatorio (enumerativo totale), ma generalmente viene usato per calcolare risoluzione nei problemi di tipo PLI. Nel caso dei problemi PLI si utilizza la metodologia d'approssimazione tramite i rilassamenti o, meglio, la risoluzione in sotto problemi più piccoli. L'algoritmo di enumerazione Branch and Buond è tra algoritmi euristici, esso viene utilizzato per problemi classe dei problemi NP-ardui (NP-completi).
L'algoritmo Branch and Buond è Metodo di enumerazione, in pratica "tentando" ogni possibile soluzione fino a trovare quella ottima ma scartando quelle che a propri non dimostrino non ottimali. La funzione obiettivo viene calcolata per un sottoinsieme di cardinalità abbastanza piccola delle soluzioni ammissibili con la proprietà di contenere almeno una soluzione ottima. Branch and Buond ha due fasi principali:

- Buond: Calcolo dei "lower bound" o "upper Bound" dei sotto problemi allo scopo di acquisire l'informazione necessaria per capire se scartare o meno un sotto problema.
- Branch: Generazione dei sotto problemi in inseriti come figli di un nodo di un albero.

Prob inziale formalismo con funzioni : max ,min

Funzione massimo	Funzione minimo
in forma lineare: $\max\{cx \ : \ x \in S\}$ (3.7)	in forma lineare: $\min\{cx \ : \ x \in S\}$ (3.8)
dove $S = \{ \ Ax \leq b \ , \ x \ \in \ \mathbb{R}^n \ x \geq 0\}$	dove $S = \{ \ Ax \geq b \ , \ x \ \in \ \mathbb{R}^n \ x \geq 0\}$

[12] ottimizzazione combinatoria (OC) studia i problemi di ottimizzazione in cui l'insieme ammissibile è definito in termini di strutture combinatori.

L'insieme ammissibile S che per l'Assunzione A ha cardinalità finita e quindi il problema non può essere illimitato inferiormente per problemi di minimo (illimitato superiormente per problemi di massimo.

Branch
Questo viene realizzato suddividendo il problema originale in q sotto problemi: Effettuando una partizione di S l'insieme delle soluzioni ammissibili PLI in una famiglia di sottoinsiemi $\left(S^0, \cdots, S^q\right)$ $r \geq 2$ tale che $S^i \cap S^j = \emptyset$ per ogni coppia $1 \leq i \leq j \leq q$ e $\bigcup_{i=1}^{q} S^i = S$.
Identificando il problema originario (denotato come $Prob^{(0)}$ con il problema P e il suo insieme ammissibile S con S_0, possiamo affermare che i nuovi problemi generati, $Prob^{(1)}, \cdots, Prob^{(q)}$ sono i "figli" del problema "padre" $Prob^{(0)}$. Questo processo vene iterato al un sotto-problema $Prob^{(i)}$ se risulta difficile da risolvere partizionando l'insieme S^i, fino a decomporre il problema originario in sotto-problemi elementari di facile soluzione. Questa operazione crea un albero di enumerazione.

Bound
Il "lower bound" ("upper buond") è una tecnica Utilizza per calcolare i sotto-problemi il "lower bound" L_i di $z^{(i)}$ ("upper buond" U_i di $z^{(i)}$) e cioè un valore $L_i \leq z^{(i)}$ ($U^{(j)} \geq z^{(i)}$). Se In un certo passo dell'algoritmo il miglior valore della funzione obiettivo trovato fino a quel momento che chiameremo valore ottimo corrente indicato $\widetilde{z}$ e $\widetilde{x}$ Se $\widetilde{z} \leq L_i \leq z^{(i)}$ ($\widetilde{z} \geq L_i \geq z^{(i)}$) allora $\widetilde{x} \notin S^i$ (se è intero nel caso dei PLI), non può migliorare il sotto problema e vene scartato.
Le possibile implementazioni per calcolare "lower bound" ("upper buond") è usando l rilassamenti o le euristiche.

Metodo del "Branch and Bound"
Indicheremo con $\mathcal{L}$ l'insieme dei sotto-problemi aperti che devono ancora essere analizzati, che sono generati nelle varie fasi di branching. Con $\widetilde{z}$ indicheremo il valore ottimo corrente e con $\widetilde{x}$ la soluzione ottima corrente. Ciascun problema $Prob^{(i)} \in \mathcal{L}$ con L_i il valore "lower bound" (U_i il valore "upper buond") e con $\bar{x}^{(i)}$ il vettore che lo raggiunge.
Inizializzazione:
Input problema = Si calcola la soluzione ammissibile iniziale inizializzando $\widetilde{x}$ e $\widetilde{z}$. Risoluzione $Prob^{(0)}$= Si applica bound per determinare un "lower bound" L_0 ("upper buond" U_0) del valore ottimo del problema $Prob^{(0)}$ su un vettore $\bar{x}^{(0)}$, per il quale si ha L_0 (U_0).

- la soluzione è "ottima" = se il suo "lower bound" $L_0 = \widetilde{z}$ ($U_0 = \widetilde{z}$) il valore ottimo corrente o la soluzione ottima corrente $\bar{x}^{(0)} \in S^{(0)}$ e termino algoritmo soluzione ottima;
- Altrimenti = Si applica branch per generare nuovi sotto-problemi che vengono inseriti nella lista dei sotto-problemi aperti $\mathcal{L}$;

Iterazione Generica:
Se $\mathcal{L} = \emptyset$: si imposta $\widetilde{x}$ e $\widetilde{z}$ e termino algoritmo soluzione ottima. altrimenti $\mathcal{L} \neq \emptyset$ = si estrae un sotto-problema aperto $Prob^{(j)}$ dalla lista $\mathcal{L}$ e si procede come segue:

- Risoluzione $Prob^{(j)}$ = Si applica bound per calcolare un "lower bound" L_j ("upper buond" U_0) del valore ottimo del sotto-problema $Prob^{(j)}$ su un vettore $\bar{x}^{(j)}$, chiudendo il sotto-problema se risulta inammissibile.
- Confronto soluzione corrente:
  - Se $L^{(j)} \geq \widetilde{z}$ ($U^{(j)} \leq \widetilde{z}$) = il sotto-problema $Prob^{(j)}$ viene chiuso in quanto nessuna $\bar{x}^{(j)} \notin S^j$ non migliorando la soluzione;
  - altrimenti $L^{(j)} < \widetilde{z}$ ($U^{(j)} > \widetilde{z}$) = il sotto-problema $Prob^{(j)}$ possibilità di migliorare l'ottimo corrente;
- verifica soluzione (intera nel caso dei PLI):
  - Se $\overline{x}^{(j)} \in S^j$ = soluzione ammissibile, Si aggiorna $\bar{z}$, e $\bar{x}$, in base ai nuovi valori;
  - altrimenti $\overline{x}^{(j)} \notin S^j$ = soluzione non ammissibile, si applica branch per generare nuovi sotto-problemi da inserire nella lista $\mathcal{L}$;

3.5.2 Branch and cut

Il Branch and cut è la combinazione del branch-and-bound con rilassamenti LP e generazione dei piani di taglio (disuguaglianze valide). I tagli permettono di migliorare sensibilmente i rilassamenti e quindi ridurre (a volte drammaticamente) il numero di operazioni di branching necessarie a risolvere il problema, inoltre spesso permettono di ottenere soluzioni continue "meno frazionarie" (quando non direttamente intere, risolvendo così i sotto-problemi)
l COIN-OR Branch and Cut solver (CBC[24]) è un open source programma misto intero (mixed-integer programming 'MIP') resolver si basa sul Branch and cut (che dovrebbe essere chiamato branch-and-cut-and-bound)

CBS MIPS resolver
Branch: viene scelta una "non-integer" ed creare due nodi dove un è "upper bound" e l'altro è un "lower bound" ed infine aggiunti nel albero di ricerca.
CBC MIPS resolver è un branch-and-cut algorithm andando a limitare in numero di variabili e il suo dominio vado a semplificare il problema LP.
Prima utilizza il Branch and cut prima di fare il loop risoluzione per ridurre il numero di variabili e semplificare i rilassamenti. Il processo si di questa parte descritto passo a passo:

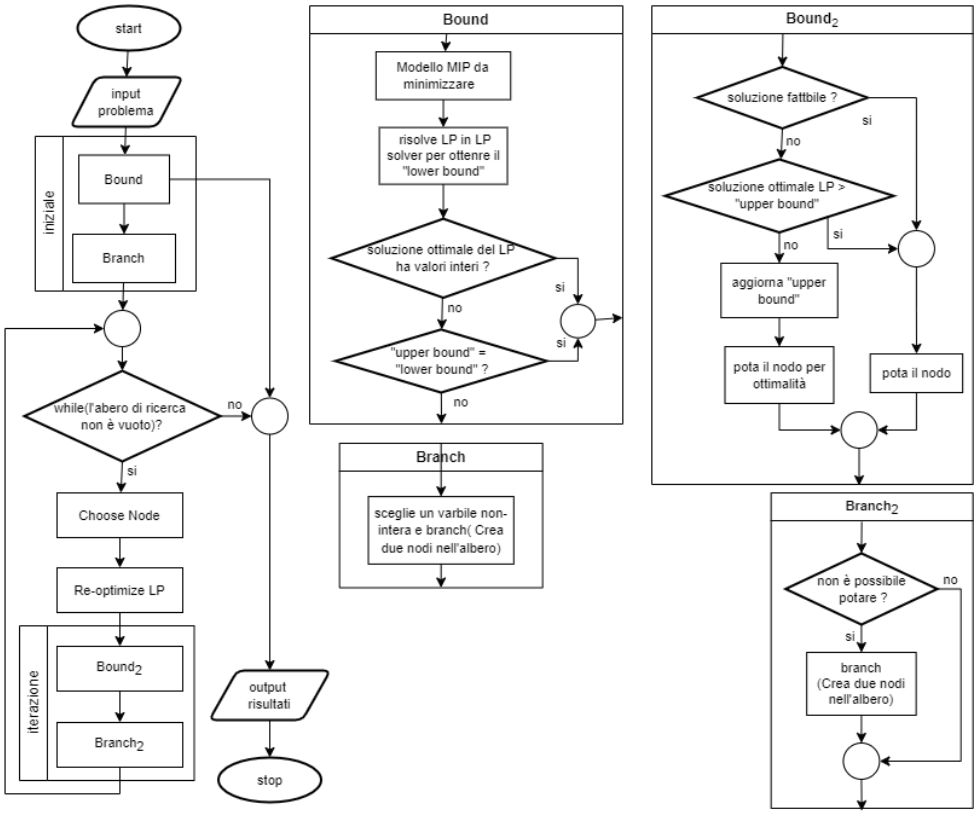
- Bound in questo passo: prima riduco variabili del modello MIP poi dopo risoluzione modello lineare nel risolutore LP per ottenere "lower bound" e ad accertare se è una soluzione ottima.
- Branch in questo passo: se esiste un variabile "integer" viene scelta quella, altrimenti viene fatto l'operazione di branch.

Nella parte della risoluzione utilizza il branch and bound con i rilassamenti semplificati. Prima seleziona il nodo dall'albero (Chose node), poi dopo (Re-optimize LP) crea il rilassamento del LP andandolo anche a risolvere. Il processo si di questa parte descritto passo a passo:

- Bound in questo passo: vado Interrogare la soluzione LP ottimale, e cercare di potare il nodo.

- Branch in questo passo: se non è possibile potare il nodo, altrimenti si fa l'operazione di Branch.

Figura 3.2: algoritomo CBS MIPS resolver



3.5.3 Branch and price and cut

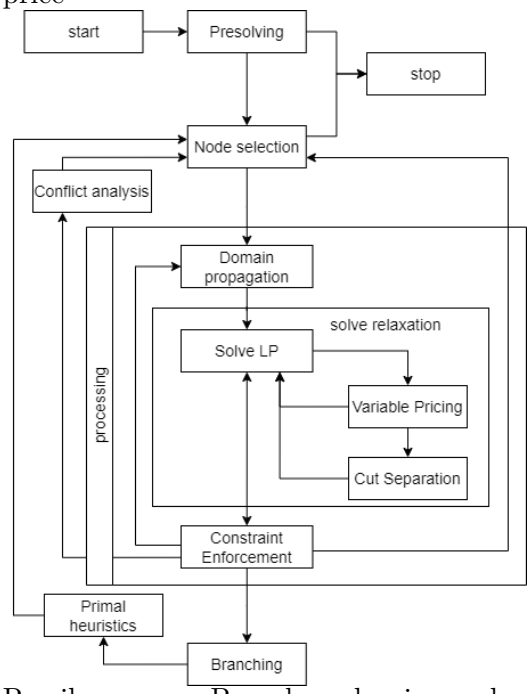
per prefazione parto introducendo Il branch-and-price è una variante del branch-and-bound. Durante l'ottimizzazione processo, a ciascun nodo, il rilassamento LP viene risolto con la metodologia generazione di colonna.
Infine,il branch-and-cut-and-price[25] è una combinazione di branch-and-price ed generazione dei piani di taglio , dove I componenti fondamentali componenti sono:

- Generatore di colonne: Le variabili sono date implicitamente e aggiunte al problema solo quando necessario.
- Piani di taglio: sono aggiunti al problema durante il processo di risoluzione per rafforzare il rilassamento LP.

SCIP[31] incorpora un risolutore di programmazione mista-intera (mixed-integer programming 'MIP'), nonché un solutore di programmazione non lineare (mixed-integer nonlinear programming 'MINLP') a interi misti basato su LP, ed è un framework per Il Branch and price and cut.

SCIP resolver

Figura 3.3: schema a blocchi Branch and price



SCIP (Solving Constraint Integer Programs) è un framework che fornice Branch and price and cut come metodo combinatori per trovare la soluzione. La maggior parte degli algoritmi che sono necessari per controllare la ricerca della soluzione, deve essere inclusi come plugin esterni.
I plugin sono oggetti di callback definiti dall'utente, che interagiscono con il framework attraverso un'interfaccia molto dettagliata fornito da SCIP. Questo è, definiscono un insieme di metodi che sono registrati in un framework e chiamato da SCIP durante il processo di risoluzione (illustrato anche nel branch-and-cut-and-price[25]). Inoltre, possono fornire ulteriori metodi specifici per i vincoli come presolving, separation, domain propagation e branching.

Per il processo Branch and price and cut è un metodo combinatorio basato su un branch-and-bound, ma nei sotto-problemi durante la risoluzione i tagli dei piani (taglio di separazione) per trovare soluzioni possibili quando è necessario e con un generatore di colonne (prezzo variabili) per aggiungere variabili quando è necessario.
I vari elementi che compongo il flow chart SCIP spigati in Intro-SCIP[30]:
"Pre-elaborazione": riduce grandezza del modello variabili irrilevanti e stringere i domini del problema generale "Selettore nodi": seleziona il prossimo nodo da elaborare, quando termina il processo di risoluzione del nodo o pre-elaborazione.
"Processo"= processo di risoluzione del nodo dell'albero:

- "Propagazione del " del nodo: viene eseguita in ogni nodo dell'albero branch-and-bound. Il suo obiettivo è quello di stringere i domini locali di variabili per il subproblem corrente. Inoltre controlla se la soluzione è impossibile.
- "Risolutore dei rilassamenti"= risolve i rilassamenti del sub-problema
  - "solve LP": nel risolutore dei rilassamenti LP che può calcolare dual bounds e soluzioni primarie, utilizzando vari risolutori.
  - "variable pricing": partendo da un insieme più piccolo di variabili, aggiungere variabili al problema durante il processo di risoluzione. Le variabili possono essere trattate implicitamente e aggiunte al problema solo quando necessario.
  - "cut separation": taglio separatori piano può produrre disuguaglianze valide che tagliano l'attuale LP soluzione o una data soluzione arbitraria, rinforzando il rilassamento. Questa operazione è fatta solo se è necessario.
- "constrain enforcement": in alcuni casi le soluzioni del LP può violare un vincolo

non contenuto nel rilassamento, è necessario correggere l'implementazione tramite il "gestore di vincoli". Il "gestore di vincoli" risolve l'infattibilità mediante la Riduzione il dominio di una variabile, Separare un piano di taglio (può utilizzare integralità), Aggiunta di un vincolo (locale), creare una ramificazione, Concludere che il sotto-problema è irrealizzabile e può essere eliminato, o Basta dire "soluzione irrealizzabile".

"Branching": regole di diramazione, se è possibile suddivide il sotto-problema corrente e crea nuovi nodi dell'albero branch-and-bound.
"Analizzatore dei confitti": equilibra albero del branch-and-bound rispetto al limite del vincolo e analizza se la soluzione e impossibile.
"Primal Heuristics": aiuta a cercare la soluzione e ad migliorare la soluzione primaria.

3.6 La modellazione del problema

Il modello del problema dell'algoritmo è il Problema dello zaino (ksnap) è un problema NP-difficile. Dipende dalle varie casistiche utilizzo delle sue varianti.
Il problema dello zaino (KP): è con la funzione obbiettivo di tipo max, nel mio caso è anche multiplo (MKP) quindi come se fossero più istanze del problema.in tal caso crea un vettore obbiettivo che prende tutti i parametri scelti che sono nei componenti in tutte le blockchain. Per cercare il numero massimo di parametri scelti di una singola blockchain per massimizzare l'obbiettivo. I parametri scelti dipendono quali componenti della blockchain si vanno ad analizzare.
Quella min-KP[23] il problema dello zaino con la funzione obbiettivo di tipo min, nel mio caso è anche multiplo. Ed in uno specifico test è multiplo e multidimensionale. Per cercare il numero minimo di parametri scelti di una singola blockchain per minimizzare l'obbiettivo. I parametri possono essere in unita secondi o con valori in virgola mobile come costi degli elementi (tipi di transazioni, blocchi, ...), velocità (tempi) di creazione degli elementi (tipi di transazioni, blocchi, ...) ed infine parametri per il controllo volume dei componenti della blockchain.

3.6.1 Problema dello zaino (ksnap)

Il problema dello zaino, nella versione di ottimizzazione, è di fondamentale importanza in quanto può essere risolto in maniera soddisfacente in molti casi di comune applicazione; infatti, per questo problema sono disponibili buone euristiche e buoni rilassamenti. Un algoritmo di enumerazione implicita, ad esempio Branch and bound, normalmente non impiega molto tempo per risolverlo. Varianti del problema dello zaino:
problema dello zaino (KP)
Cerca un subset di parametri $x \in \mathbb{R}^n$ dove per ogni componete contiene delle blockchain un peso $w \in \mathbb{R}^n$ per massimare valore p, in modo da non superare o eguagliare la capacità $C \in \mathbb{R}$ dello 'zaino' (nella formulazione generale si parte dal problema dello zaino 0-1)

$$max \left\{ \sum_{i=1}^{N} p_i x_i \ : \ \sum_{i=1}^{N} w_i x_i \leq \ W, \ x_i = \{0,1\} \ \forall i = 1, \ldots, n \right\} \tag{3.9}$$

Il problema dello zaino minimo (min-KP)
Cerca un subset di parametri $x \in \mathbb{R}^n$ dove per ogni componete contiene delle

blockchain un peso $w \in \mathbb{R}^n$ per minimizzare valore p, in modo da superare o eguagliare la $C \in \mathbb{R}$ capacità dello 'zaino'(nella formulazione generale si parte dal problema dello zaino 0-1)

$$min \left\{ \sum_{i=1}^{N} p_i x_i \ : \ \sum_{i=1}^{N} w_i x_i \geq \ C, \ x_i = \{0,1\} \ \forall i = 1, \ldots, n \right\} \tag{3.10}$$

Problema dello zaino varianti
Avendo $W \in \mathbb{R}^{n \times m}$ le varianti dipendono dal tipo di variabili utilizzate:

- Problema dello zaino 0-1 $X = \{x_i \, \{0,1\} \ \forall i = 1, \ldots, n \}$;
- Problema dello zaino con limiti (buond) B $X = \{x_i \leq b_i \ \forall i = 1, \ldots, n \}$;
- Problema dello zaino senza limiti (unbuond) UB $X = \{x_i \in N \ \forall i = 1, \ldots, n \}$;

Ma anche dalla dimensione del vettore di costi $P \in \mathbb{R}^{n \times 1}$ o multi-obiettivo $P \in \mathbb{R}^{n \times r}$

KP
Multiple costanti o multidimensionale $P \in \mathbb{R}^{n \times 1}$

$$\begin{cases} max \ \sum_{i=1}^{N} p_i^T \cdot x_i \\ \sum_{i=1}^{n} w_{ij} \cdot \ x_i \leq \ c_j \quad j = 1, \cdots m \\ x_i \in X \ , \ x \geq 0 \ \forall i = 1, \ldots, n \end{cases} \tag{3.11}$$

multi-obiettivo e multidimensionale $P \in \mathbb{R}^{n \times r}$

$$\begin{cases} max \ \sum_{k=1}^{r} \sum_{i=1}^{n} p_{ik}^T \cdot x_i \\ \sum_{i=1}^{n} w_{ij} \cdot \ x_i \leq \ c_j \quad j = 1, \cdots, m \\ x_i \in X \ , \ x \geq 0 \ \forall i = 1, \ldots, n \end{cases} \tag{3.12}$$

Min-KP
Multiple costanti o multidimensionale $P \in \mathbb{R}^{n \times 1}$

$$\begin{cases} min \ \sum_{i=1}^{N} p_i^T \cdot x_i \\ \sum_{i=1}^{n} w_{ij} \cdot \ x_i \geq \ c_j \quad j = 1, \cdots m \\ x_i \in X \ , \ x \geq 0 \ \forall i = 1, \ldots, n \end{cases} \tag{3.13}$$

multi-obiettivo e multidimensionale $P \in \mathbb{R}^{n \times r}$

$$\begin{cases} min \ \sum_{k=1}^{r} \sum_{i=1}^{n} p_{ik}^T \cdot x_i \\ \sum_{i=1}^{n} w_{ij} \cdot \ x_i \geq \ c_j \quad j = 1, \cdots, m \\ x_i \in X \ , \ x \geq 0 \ \forall i = 1, \ldots, n \end{cases} \tag{3.14}$$

3.6.2 Risoluzione

La risoluzione è data da un algoritmo che prende le matrici A e i vettori c ed b. Queste matrici e vettori vengono inserite nei risolutori nei test per verificare tempo di esecuzione e iterazioni.
La matrice A prende tutte le blockchain con i loro componenti che a suo interno hanno dei parametri. Invece la matrice b è data dai vari limiti dei parametri che compongono componenti, essi con varie blockchain. Infine, la funzione lineare dell'obbiettivo che varia per utilizzo del parametro da ottimizzare, in questo documento sono testati elementi, velocità e tempo. Esiste un caso trattato nel documento in cui c'è una matrice avendo più una funzione obbiettivo.

algoritmo risoluzione base
Viene scelto l'insieme delle BlockChain $BC = \{BC_1 \cdots BC_s\}$ e i parametri $x = \{x_1 \cdots x_n\}$. il problema se è di massimo o di minimo dipende dal tipo di problema.
$\forall BC_k$ BlockChain possiede t Componenti(entità) $BC_k = \bigcup_{v=1}^{t} E_v$ che $\forall E_v$ dove l'assieme E_v conteni i paramatri x_v con e elementi si può scrivere in maniera matriciale dove $p = \dim x_p$ si ha $E_v \in \mathbb{R}^{e \times p}$. quindi se hanno i parametri dell'insieme

x allora $x \cap E_v \neq \emptyset$ altrimenti $x \cap E_v = \emptyset$.
$\forall BC_k$ avendo $l_k = \sum_{v=1}^{t} \dim E_v$ (numero di tutti elementi dei componenti) e $\exists x_k \in \mathbb{R}^{l_k}$ si avrà dei vicoli lineari sono composti da: $A_k \in \mathbb{R}^{n \times l_k}$ e $b_k \in \mathbb{R}^n$, la sua Funzione obbiettivo $c_k \in \mathbb{R}^{l_k}$.
la risoluzione si può descrivere:
$\forall BC_k \in BC$, essendo $X_{BC} = \bigcup_{k=1}^{s} x_k$ avendo funzione obbiettivo $z : \mathbb{R}^n \to \mathbb{R}$ dove $x \in \mathbb{R}^n$, $C_k \in \mathbb{R}^{l_k}$ si ha $z_k(x_k, C_k) = C_k^T \times x_k$ si le matrici

$$x_k = \begin{bmatrix} x_{k1} \\ \vdots \\ x_{kl_k} \end{bmatrix}, A_k = \begin{bmatrix} a_{11} & \cdots & a_{1l_k} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nl_k} \end{bmatrix}, b_k = \begin{bmatrix} b_{k1} \\ \vdots \\ b_{kn} \end{bmatrix} \text{ ed } C_k = \begin{bmatrix} c_{11} & \cdots & c_{1l_k} \end{bmatrix}$$

per iniziare prima setto il problema (la funzione obbiettivo) se è di massimo (max) o di minimo (min), poi $\forall BC_k \in BC$ (BC : BlockChain). Il settaggio della funzione obbiettivo $\sum_{k=1}^{s} z_k(x_k, C_k)$, nel caso multi-obbiettivo $\sum_{r=1}^{q} \sum_{k=1}^{s} z_k(x_k, C_{kr})$.
Con la matrice A_k e il vettore b_k , si definisce le disequazioni $\forall BC_k \in BC$ dove $\forall E_v \in BC_k$ ed $\exists\, x$(esistono i paramatri) per cui utilizzando li funzione $g(x)$. dove la funzione $g(x)$ se hanno i parametri x l'insieme $x \cap E_v \neq \emptyset$ allora si crea $a_v \in \mathbb{R}^n$ con tutti paramatri in comune ed il resto si si aggiunge degli zeri, altrimenti $x \cap E_v = \emptyset$ si crea un vettore $0_{[n,1]}$.

$$\begin{cases} \sum_{v=1}^{t} g(x) \leq b_{kv} & \text{se la funzione obbiettivo max} \\ \sum_{v=1}^{t} g(x) \geq b_{kv} & \text{se la funzione obbiettivo min} \end{cases} \tag{3.15}$$

Le variabili del problema dell'insieme di blockchain si possono definire $\forall BC_k \in BC$ dove $\exists\, x_k \in X_{BC}$ per cui $x_k \geq 0$. Alla fine risolvo il problema tramite il risolutore
Rappresentazione del problema:
multidimensionale

$$max \left\{ \sum_{k=1}^{s} z_k(x_k, C_k) \ : \ \forall BC_k \in BC \ x_k \times A_k \leq b_k; \exists\, x_k \in X_{BC} \ x_k \geq 0 \right\} \tag{3.16}$$

multi-obiettivo multidimensionale

$$max \left\{ \sum_{r=1}^{q} \sum_{k=1}^{s} z_k(x_k, C_{kr}) \ : \ \forall BC_k \in BC \ x_k \times A_k \leq b_k; \exists\, x_k \in X_{BC} \ x_k \geq 0 \right\} \tag{3.17}$$

multidimensionale

$$min \left\{ \sum_{k=1}^{s} z_k(x_k, C_k) \ : \ \forall BC_k \in BC \ x_k \times A_k \geq b_k; \exists\, x_k \in X_{BC} \ x_k \geq 0 \right\} \tag{3.18}$$

multi-obiettivo multidimensionale

$$min \left\{ \sum_{r=1}^{q} \sum_{k=1}^{s} z_k(x_k, C_{kr}) \ : \ \forall BC_k \in BC \ x_k \times A_k \geq b_k; \exists\, x_k \in X_{BC} \ x_k \geq 0 \right\} \tag{3.19}$$

[13]Tra tecniche efficienti per la soluzione di problemi di PLI si fondano sull'individuazione e sullo sfruttamento di strutture combinatorie specifiche nel modello PLI, ossia sull'individuazione di (sotto)problemi di OC corrispondenti al modello di PLI o a sue parti.

Capitolo 4: data base

Il database è stato modellato con parametri contenuti nelle blockchain analizzate nella fase II. Dopo aver predisposto in parametri del data base ho verificato che sia compatibile con la maggioranza delle blockchain. Alcune blockchain sono fork da altre al contrario altre cambia il protocollo e struttura diversa. Tutto dipende dallo scopo e le funzionalità che implementate da esse, si avrà varie architetture strutturate in layer.

4.1 verifica dei dati

La verifica compatibilità con le varie blockchain dipende dalla struttura interna dei componenti e le funzionalità dei componenti.
Nelle varie blockchain dipende dalla funzionalità implementate si avrà una struttura dei layer differente e differente scopo. Si avrà lo stesso cambiamento se considero che la blockchian è fork quindi cambiano e regole o/e i protocolli o/e aggiungono funzionalità, se una blockchian si apponga ad un ecosistema di blockchain esistente, come altra possibilità è aver una blockchain con il protocollo e struttura diversa.
Le funzionalità delle blockchian parametrizzate si possono dividere in quelle della sua struttura interna e dell'implementazione dei blocchi.
Per la struttura interna della blochchain sono: algo type, fee, channel message, channel message, layer, node, account.
Per i blocchi della blochchain sono: Transazione blocco, Micro transazione, smart contract, Transaction other.

4.2 glossario

Tabella 4.1: glossario dei termini db

termine	descrizione	sinonimi
algo type	tipologia di algoritmi di protezioni utilizzati	Tipo algoritmo
BlockChain	blockchain da testare	Catena di blocchi
layer	layer della blockchain	livello
node	miner, server	nodo
account	wallet all'interno della blockchain	Portafogli, conto
fee	valore del lavoro del validatore per effettuare la transazione	tassa
Block	blocco di una blockchain	blocco

tabella 4.1 continua nella prossima pagina

termine	descrizione	sinonimi
smart contract	smart contract sono operazioni fatte dai validatori per confermare le transazioni	Contratto intelligente
block Transaction	transazioni base effettuate dentro un blocco	transazione blocco
micro Transaction	transazioni intermedie fatte prima della transazione base	Micro transazione
Transaction other	transazioni interne basate su smart contratti alla transazione base	altra transazione
channel message	messaggi (log) dei canali	Canale messaggi
channel transaction	message delle transazioni	Canale transazione

4.3 diagramma

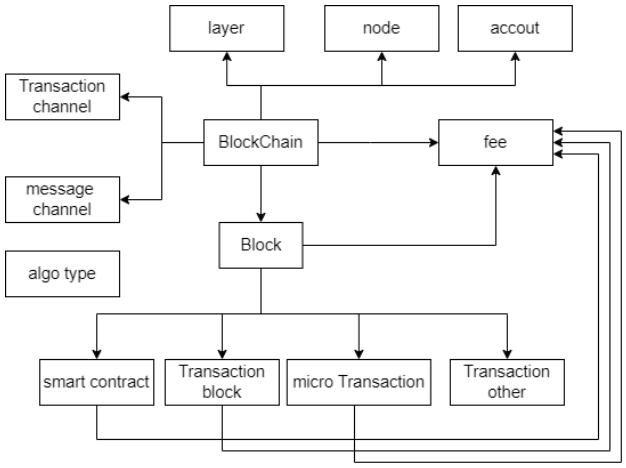


Figura 4.1: digramma delle entità

4.4 tabella entità

Tabella 4.2: tabella delle entità del db

entità	collegatemi	attributi	opzionale
algo type		type: str, bit: int, asimmetrico: bool, out max char: int, out max char inf: bool, note: text,	

tabella 4.2 continua nella prossima pagina

entità	collegatemi	attributi	opzionale
BlockChain	layer, node, account, fee, block, channel message, channel transaction	nome: text classifica posizione: int, nascita, note: text	
block	fee, smart contract, block Transaction, micro Transaction, Transaction other	n tot translation: int, time, size: float, difficulty: double, last nonce: double, reward	weight: float, n fail: int, n witness: int, n block: int,
block Transaction	fee	avg time: sec, version: int, n token:int	size, risk: double, min time: sec, max time: sec,
micro Transaction	fee	n micro- -transaction: int, avg time: sec,	n in count: int, n out count: int, min time: sec, max time: sec,
Transaction other	fee	avg time: sec, n int event: int,	min time: sec, max time: sec,
smart contract	fee	avg time: sec, n avg contract: int, avg fee	n verified contract: int, n min contract: int, n max contract: int, min time: sec, max time: sec,
fee			eth:[burn, saving,], eth: [gas use: int, gas limit: int, gas for fee base: int, gas for fee max: int, gas for fee max- -priority: int,] bitcoin: [avg fee kb, avg fee kwb]
layer		n layer: int n trans proc: int, n tot bloks: int, n tot transaction: int, size blockchain	
node		n node: int, hash_rate_tot: float	
wallet		n account: int n account active: int n avg tx account: int,	n contract: int, n max tx account: int, n min tx account: int,
channel message		n message: int	n tx message: int, in: bool, out: bool
channel transaction		n transaction: int	in: bool, out: bool

Capitolo 5: algoritmo

In questa parte del documento andremo a descrivere il funzionato dell'algoritmo. Sia l'integrazione delle varie librerie per la risoluzione, anche la formattazione dei dati che utilizzeranno. la verifica sei risultati e componenti sono stati utilizzati unitest del framework.

5.1 obiettivi algoritmo

algoritmo per identificare con parametri di default o inseriti per trovare la blockchain giusta con il costo massimizzando le transizioni ottimali utilizzando due librerie diverse per la risoluzione, come uscita vado a ridare la blockchain giusta per i miei parametri.

5.2 settaggio algoritmo

L'algoritmo utilizza dati in un database che contiene le "immagini della blockchain" con i vari componenti che la compongono ed ogni componente ha i suoi i vari parametri. Le "immagini della blockchain" caricate con una funzione, poi per convertirle utilizzo una classe che prende i dati e li divide per parametri e li converte in liste che possono essere prese con una funzione della classe in forma di vettore o matrice selezionando i parametri.
Le librerie che utilizzano l'algoritmo sono Pulp, la libreria di Google OR-Tools che ha delle classi per andare a settare i modelli ed avere la funzione di risoluzione del modello settato. Queste classi sono basate su una classe con delle funzioni comuni che convertono ogni singola "immagine della blockchain" e il controllo argomenti per la chiamata della classe come funzione.
Per testare i vari componenti e l'algoritmo ho utilizzato i unitest integrato del framework.

5.3 implementazione librerie nell'algoritmo

L'algoritmo di ottimizzazione per calcolare i problemi(PL)utilizza: la libreria Pulp[27] , la libreria di Google OR-Tools[29]. L'algoritmo ha una classe per ogni libreria dove il suo risolutore dell'algoritmo, che è richiamabile come una funzione.
Il primo risolutore che utilizzo è 'CBC'[24] della COIN-OR Foundation[28] dove una delle librerie in cui ha un'interfaccia in python è Pulp[27]. il risolutore 'CBC'[24] utilizza la metodologia del Branch-and-Cut solver.
Il secondo risolutore che utilizzo è 'SCIP'[31] dove la sua libreria in cui ha una delle librerie in python è la libreria di Google OR-Tools[29]. il risolutore 'SCIP'[31] utilizza la metodologia del branch price and cut.

5.4 algoritmo

Utilizzo algoritmo all'interno di django per trovare la blockchain ideale per lo scopo e i vicoli (limiti) impostato dati dal cliente.

Questo use case descrive il funzionamento dell'algoritmo con i dati del cliente già presi nell'interfaccia web, ed rida i risultati per la parte di visualizzazione dalla stessa interfaccia.

Figura 5.1: algoritmo generale

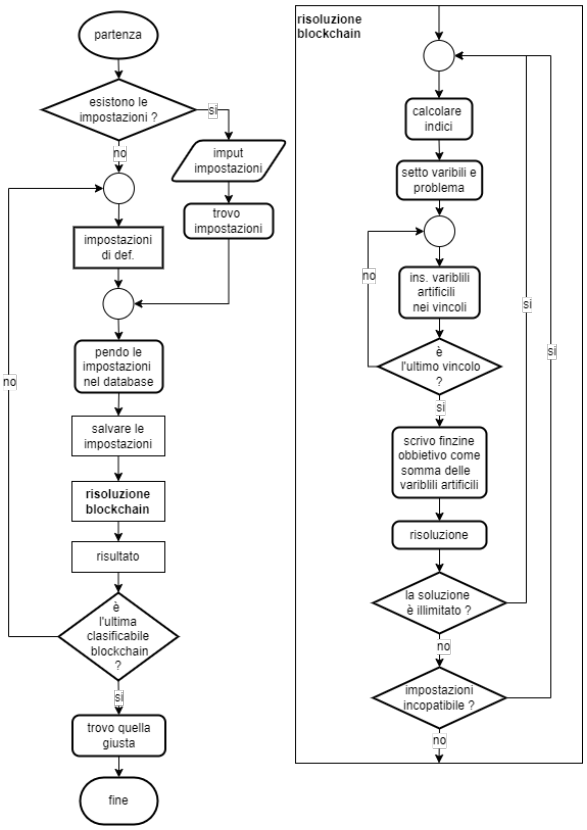
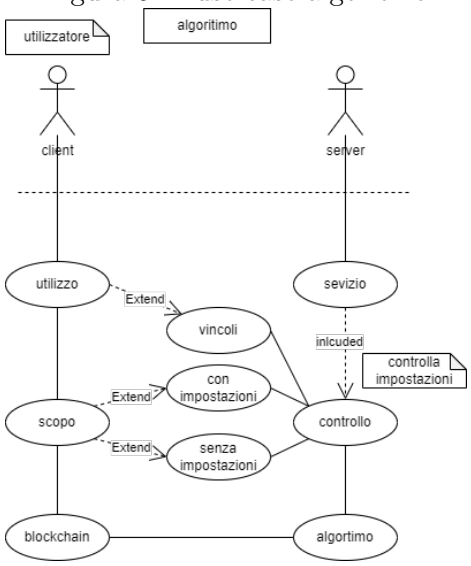


Figura 5.2: use case algoritmo



Attori

Cliente: utilizza algoritmo per l'analisi della blockchain ideale per lo scopo specifico.

Sever: macchia dove l'algoritmo è implementato.

UseCase

- Il **client** *utilizza* algoritmo per analisi blockchain è esteso (exstand) inserendo in *vincoli*.
- Specifica lo *scopo* dell'analisi è esteso (exstand) dalla scelta di fare l'analisi se è *senza impostazioni* o se è *con impostazioni*.
- Il **server** che ha il *sevizio* include (incuded) la funzione nel caso di controllo per verificare le impostazioni.
- Il *controllo* verifica i vincoli, se è *senza impostazioni* applicando quelle di default, se con *impostazioni*.
- Queste impostazioni vengono date all'*algoritmo*.
- L'*algoritmo* rida la *blockchain* ideale per scopo dell'analisi.

Eccezioni

Risultato analisi blockchain ha dato un errore in queste casistiche:
Configurazione errata senza vincoli, impostazioni caricate o impostazioni di default errate.

5.4.1 implementazione in django

Nel web-framework django l'algoritmo è implementato nelle varie classi descritte e spiegato il funzionamento.

Creazione liste dei queryset del data base
Nel file "convert_data.py" funzinameto:

- La funzione "objects_get" tramite un ingresso prende la classe o se esiste la funzione 'all' per prendere i suoi dati; questa funzione è usata in "queryset" per "algotype", e "set_objects_get" per prendere tutti i dati tramite la funzione 'all';
- La funzione "get_data_db" crea un dizionario con tutti i componenti che sono all'interno del database dove in ogni componente ha una lista di singole queryset;

Funzioni

Figura 5.3: presa dati da db: "get_data_db_in"

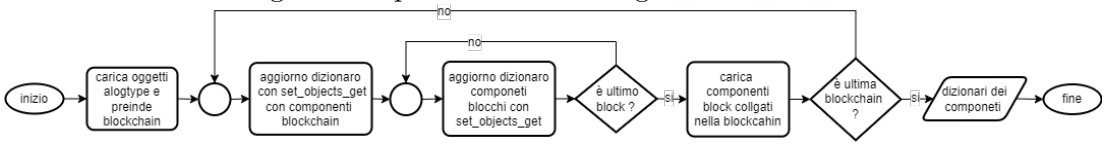
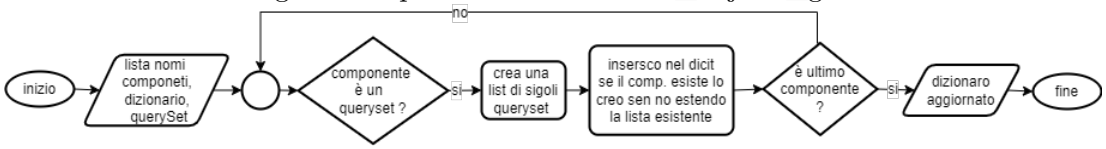


Figura 5.4: presa dati da db: "set_objects_get"



Creazione classe inserimento
La classe "ProcessData" nel file file "convert_data.py"
Creazione classe inserimento funzionamento:

- Nella funzione di "___init___" inserisce gli argomenti di baseche sono 'data' e 'columns' e quelli opzionali sono 'exclude' esclude attributi e 'comune' attributi in comune. Al suo interno per convertire i dati in liste si ha la funzione "_take_querryset";
- "_take_querryset" al suo modo ha la funzione "___get_columns" per convertire i parametri dei componenti prendendo anche gli argomenti opzionali come 'exclude' e 'comune';
- "___get_columns": Prima usufruendo di "___elem_map" per prendere gli attributi comuni ed assieme agli attributi dati vengono filtrati da "___field_get". Alla fine utilizza "___querrySet_get" per prendere le queryset degli attributi processati ridandole in un dict e la lista di chiavi del dict;

Funzioni della classe

Figura 5.5: classe inserimento: "__take__querryset"

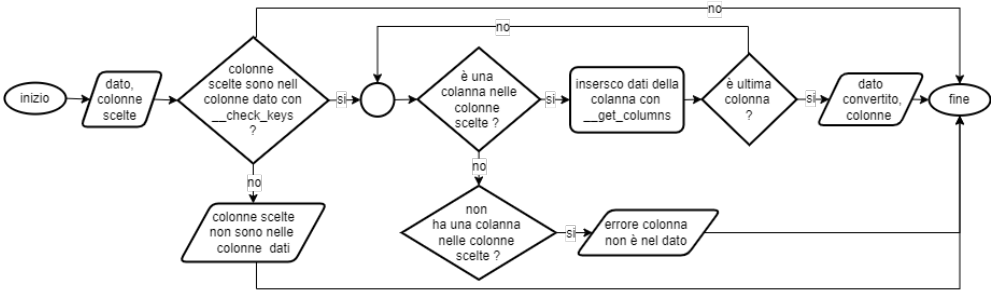


Figura 5.6: classe inserimento: "get__colunms"

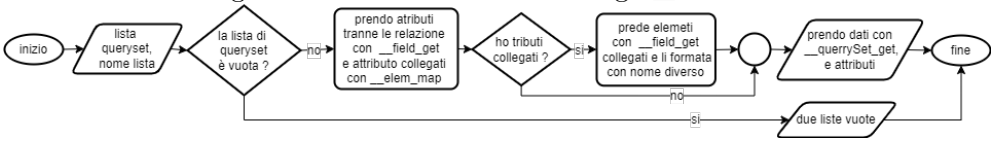


Figura 5.7: classe inserimento: “___elem__map”

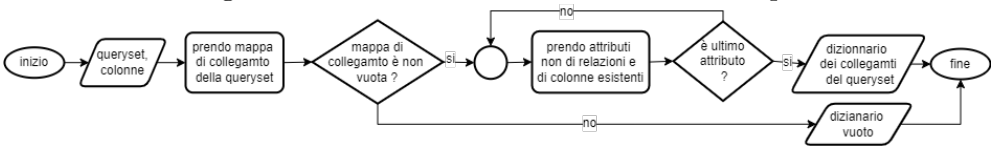


Figura 5.8: classe inserimento: “___field__get”

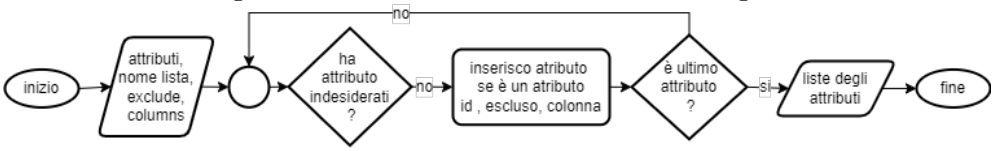
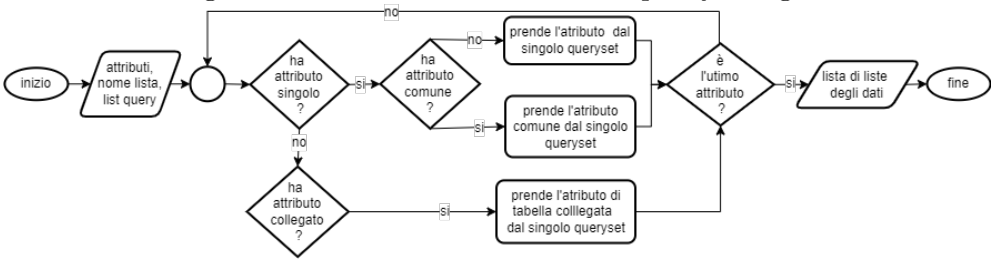


Figura 5.9: classe inserimento: “___querrySet__get



Creazione matrici e vettori

Funzionamento:

- Nella funzione "conv__list" prende gli parametri scelti nei vari componenti tramite ’param’ e scegliere se un vettore o una matrice con ’matrix’, i valori nulli di base sono 0 altrimenti si possono cambiare tramite ’vectror__def’;

- La funzione "conv_list" prima controlla che i parametri scelti siano dentro i parametri della classe poi nei componenti cerca i parametri selezionati, se alcuni dei componenti con quei parametri non ci sono vengono messi degli zeri. Alla fine per ridare il risultato se è una matrice viene ritornato come un dizionario e è un vettore viene ridato come una lista;

Tabella 5.1: funzioni creazione matrici e vettori

"___check_keys": Controlla che nel dato 's' contenga elementi delle colonne 'd' utilizzando "___check_vector"= • Se 'vector' è True guarda se sono tutti falsi; • Se 'vector' è False guarda se ne esiste uno falso;
"___check_vector": La funzione ha un ingresso 's' lista di bool che controlla= • Se sono tutti falsi se 'vector' è True; • Se ne esiste uno falso se 'vector' è False;
"control_param": Controlla che i parametri 'p' dati siano nei parametri dentro la classe utilizzando "___check_param"= • Se 'vector' è True guarda se sono tutti falsi; • Se 'vector' è False guarda se ne esiste uno falso;

Figura 5.10: classe inserimento: "___check_param"

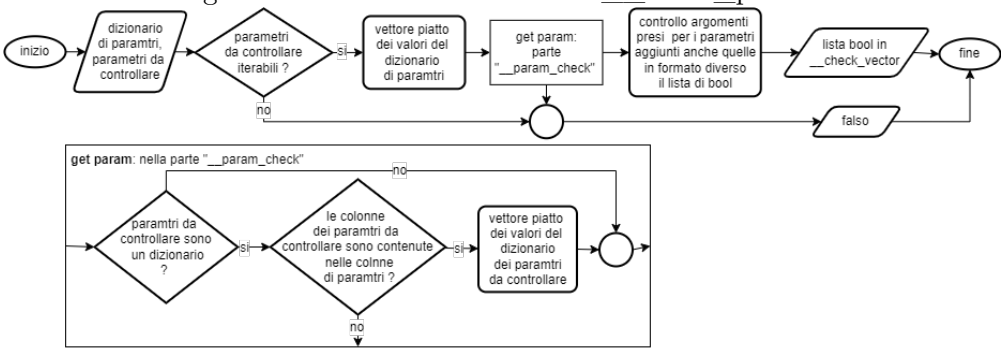
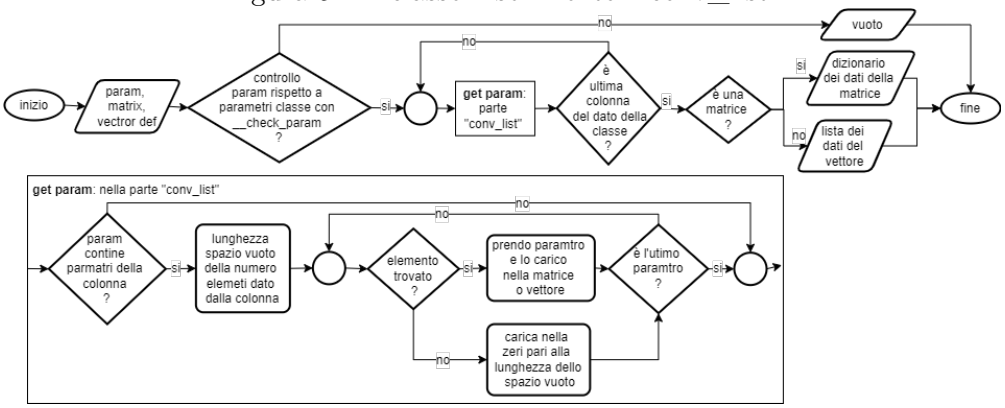


Figura 5.11: classe inserimento: "conv_list"



Modello utilizzando libreria Pulp

La classe "PuLPModel" funzionamento:

- Prima si inizializza il problema con la funzione "___init___" della classe con il nome del problema e il tipo di funzione se è minima 'LpMinimize', o massima 'LpMaximize';
- In seguito, si settano le variabili con la funzione "var_config" con una 'lista di nomi dei vincoli' e il 'kwargs' del settaggio delle variabili;
- Poi si settano i vincoli con la funzione "constraints_model" con una matrice 'A' ed il vettore 'b', con il tipo di operatore ('<=' = 'le'; '>=' = 'ge'; '==' = 'eq') e una lista dei nomi dei vincoli:
- Alla fine per settare l'obiettivo con 'c' ed 'variabili' e per risolvere il modello utilizza la funzione "___call___" composta dalle funzioni "_objecttive_model" e "___resolver_model". Essa ritorna il risultato del modello con lo stato della risoluzione e la soluzione dei vincoli e la soluzione delle variabili;

Funzioni della classe

Figura 5.12: classe : "___init___"

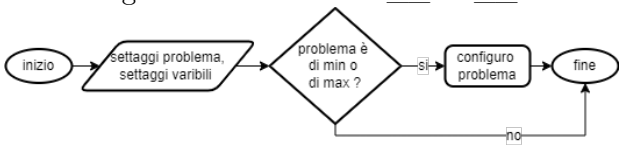


Figura 5.13: classe : "var_config"

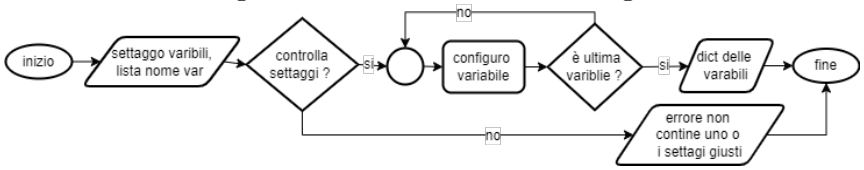


Figura 5.14: classe : "constraints_model"

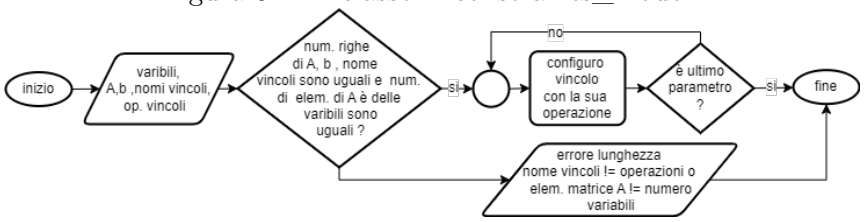
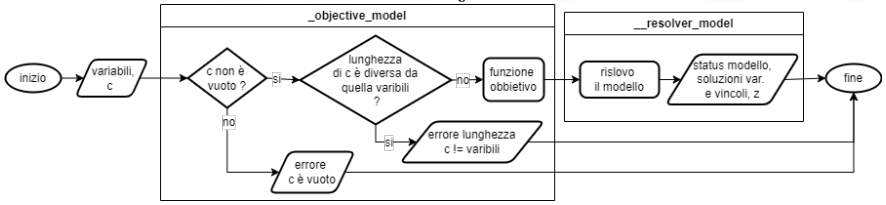


Figura 5.15: classe : "___call___" e "_objecttive_model" e "___resolver_model"



Modello utilizzando libreria OR-Tools

La classe "OrtoolsModel" funzionamento:

- Prima si inizializza il problema con la funzione "___init___" della classe con il nome del problema e il tipo di funzione se è minima 'min' o massima 'max';
- Poi si settano le variabili insieme ai vincoli con 'kwargs' del settaggio delle variabili, con una matrice 'A' ed il vettore 'b';
- Alla fine per settare l'obiettivo con 'c' ed 'variabili'. il risultato del modello viene dato con la funzione "___resolver_model" che calcola la soluzione delle variabili e vincoli, ritornando lo status modello e il valore obiettivo;

Funzioni della classe

Figura 5.16: classe : "___init___"

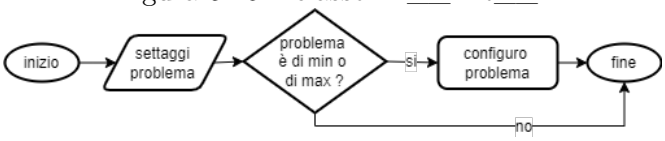


Figura 5.17: classe : "var_constraints_model"

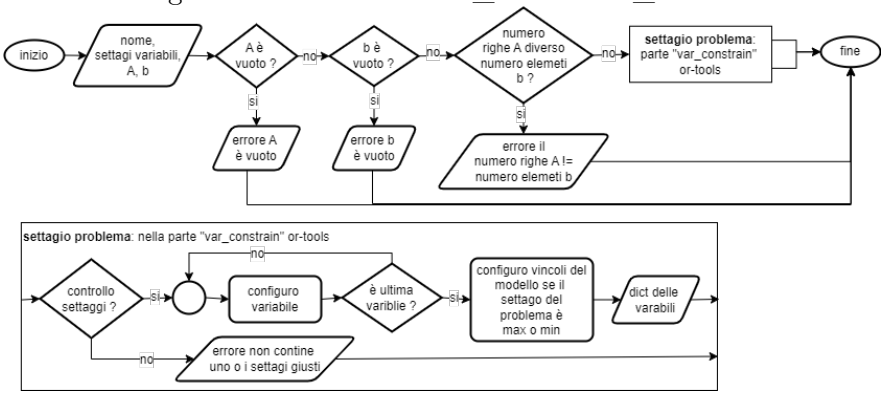
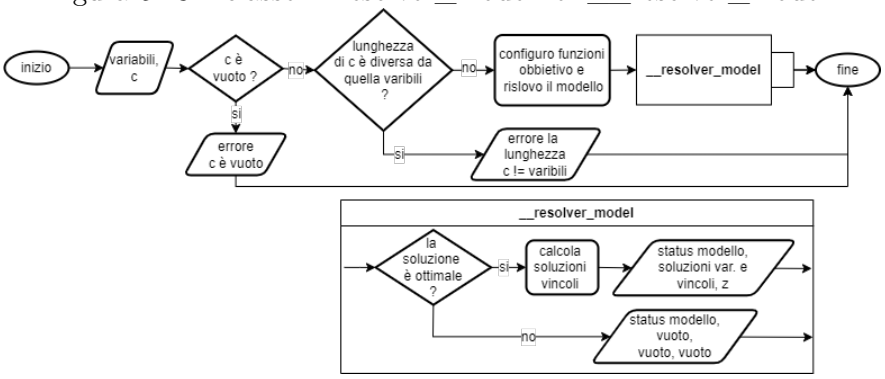


Figura 5.18: classe : "resolver_model" e "___resolver_model"



Base dell'algoritmo per funzioni comuni ed inizializzazione dei parametri
La classe "AlgoOttimoBase" funzionamento:

- La funzione "___init___" inizializzo la classe dell'algoritmo inserendo i parametri di default della matrice A ed i valori esterni di b ed il dizionario dei dati delle blockchain, in seguito i dati vengono processati dalla funzione "___get_data" poi i parametri di default della matrice A vengono controllati dalla funzione "_param_control";
- "_f_group_var_sol" è utilizzata per formattare la soluzione che ha n blockchain, facendo lista dei valori degli obiettivi delle singole blockchain e raggruppando le soluzioni delle variabili e dei vincoli ed il suo valore dell'obiettivo in un singolo dizionario;
- La funzione "___call___" ha integrato il controllo degli argomenti dato dalla funzione "_arg_call_controll" , che dal tipo di problema da risolvere ed i parametri inseriti. Codesta funzione viene utilizzata nelle classi "PuLPModel" e "ortoolsModel" per la chiamata alla risoluzione, dove l'algoritmo va a dare sempre un risultato;

Funzioni della classe

Figura 5.19: classe base dell'algoritmo: "___init___" e "___get_data"

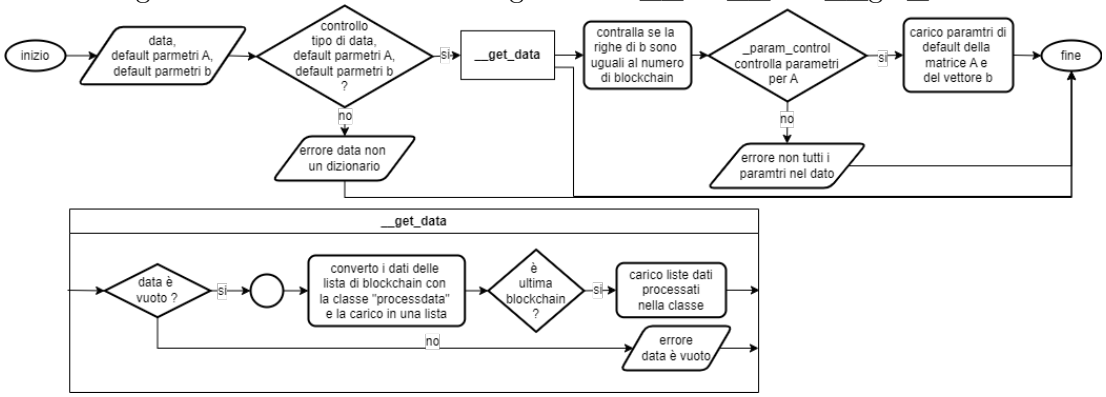
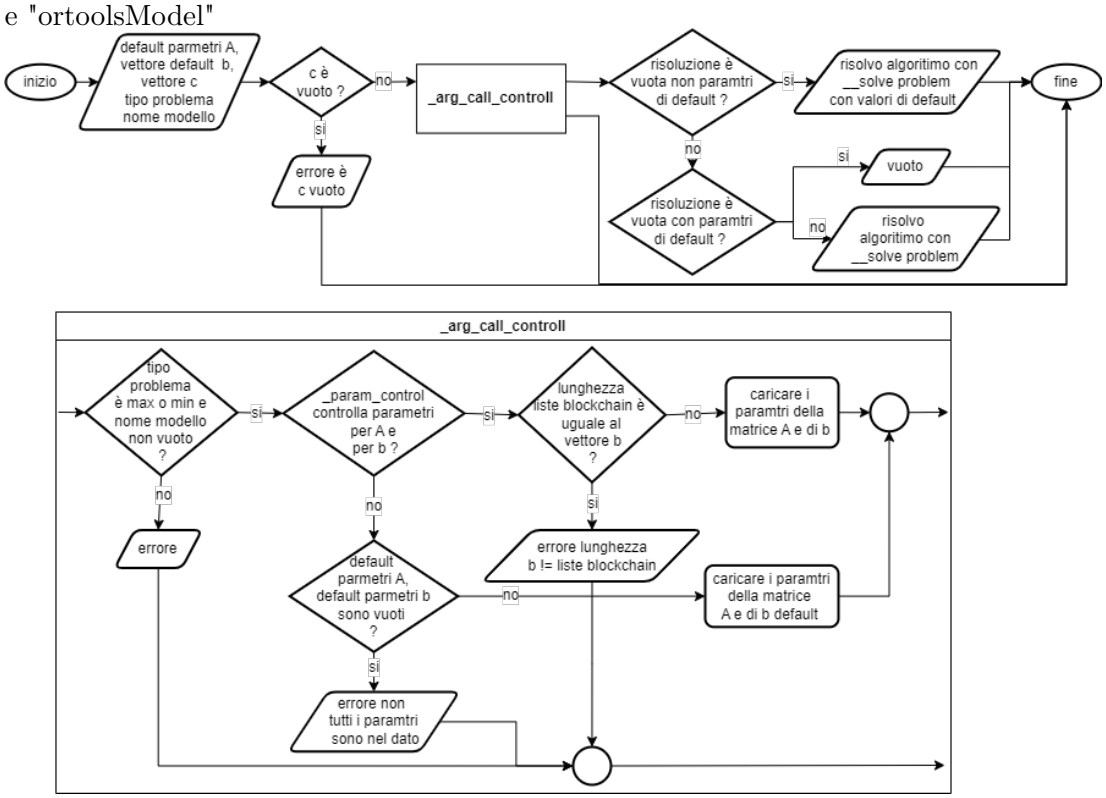


Tabella 5.3: funzioni base dell'algoritimo

"_param_control": Utilizza per controllare i parametri di per la matrice A utilizzando "___check_param_data"
"___check_param_data": Richiama su tutta la lista di blockchain convertite la funzione "control_param" della classe "ProcessData" per controllare Se quei parametri sono in tutte le blockchain
"_dict_format": Formatta dato della soluzione= soluzione variabili e vincoli utilizzando la funzione "_name_dict_find" per essere presi invece valore obiettivo si prende direttamente funzione
"_name_dict_find": Trova la substring in comune nei keys di un dict

Figura 5.20: classe base dell'algoritmo: "__f__group__var__sol"

Figura 5.21: classe base dell'algoritmo:"___call___" nelle classi derivate "PuLPModel" e "ortoolsModel"

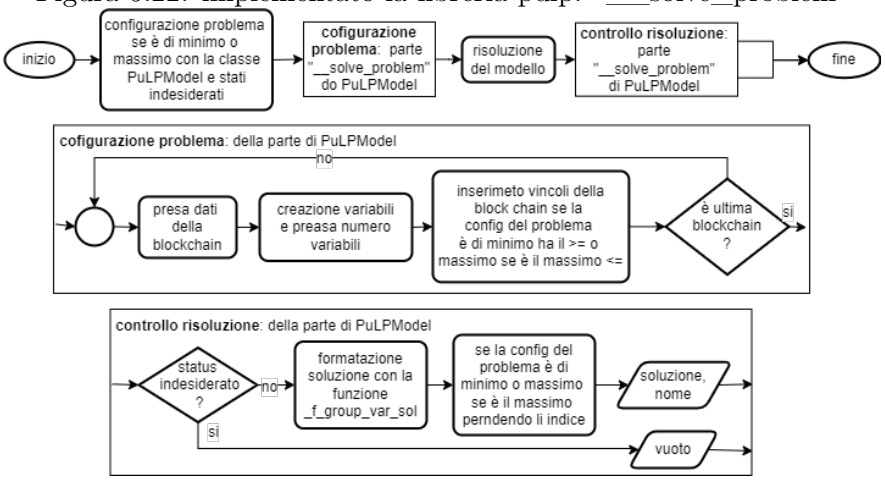


Algoritmo implementato la libreria pulp

La funzione di risoluzione della classe "PuLPModel" con la base "AlgoOttimoBase" funzionamento:

- La classe base "AlgoOttimoBase" inizializza i parametri di default utilizzando funzioni comuni;
- La funzione "___call___" usa il controlla parametri con una funzione in comune con la sua base, mentre per risolvere il modello utilizza la funzione "___solve__problem" che è nella classe specifica della singola libreria;
- La funzione "___solve__problem" è la funzione di risoluzione del modello che può essere di massimo o minimo ed ha una funzione obiettivo. Egli prende i dati su tutte le blockchain e dati esterni per i vincoli;

Figura 5.22: implementato la libreria pulp: "___solve_problem"

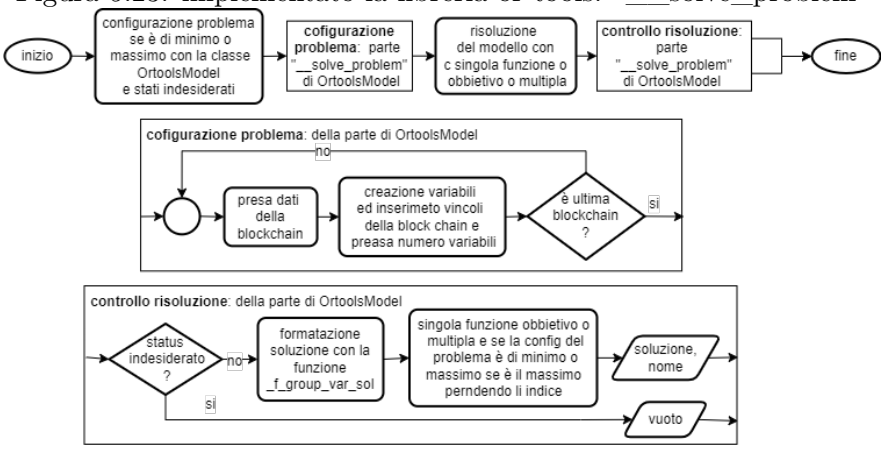


Algoritmo implementato la libreria or-tools

La classe "OrtoolsModel" con la base "AlgoOttimoBase" funzionamento:

- La classe base "AlgoOttimoBase" inizializza i parametri di default utilizzando funzioni comuni;
- La funzione "___call___" usa il controlla parametri con una funzione in comune con la sua base, mentre per risolvere il modello utilizza la funzione "___solve_problem" che è nella classe specifica della singola libreria;
- La funzione "___solve_problem" è la funzione di risoluzione del modello che può essere di massimo o minimo ed ha una funzione obiettivo. egli prende i dati su tutte le blockchain e dati esterni per i vincoli;

Figura 5.23: implementato la libreria or-tools: "___solve_problem"



Capitolo 6: test algoritmo

in questa parte del documento andiamo ad analizzare i risultati dei unitest del framework, pertanto andando a spiegare il motivo di quel settaggio (blochchain e parametri presi dal data base) caso per caso trattato. I risultati del test verranno confrontati rispetto ai diversi risolutori implementati nelle diverse librerie, con vari parametri che combino nei vari casi che dipende dall'obbiettivo del test trattato.

6.1 casi trattati test

i casi trattati si ho scelto delle blockchain e parametri delle loro immagini con dati 'dummy' (che simulino un caso di utilizzo reale). Visionato nel dettaglio la scelta di blockchain ed le motivazioni legate alla su scelta e 'explorer blockchain'(esploratore blockchain usato per veder i dati della blockchain) usati. blockchain utilizzate di casi trattati per il motivo:

- Bitcoin: blockchain principale dove operazioni a micro-transazioni nella transazione, dove nuove blockchain usate per diversi scopi si basano o si creano fork di Bitcoin. 'explorer blockchain' per prendere i dati Blockchair[18] , CryptoQuant[12], ViaWallet[21].
- Ethereum: blockchain principale dove operazioni a interne nelle transazioni con l'implementazione di token che rendono ampio il suo ecosistema. 'explorer blockchain' per prendere i dati Blockchair[19] ,etherscan[16], CryptoQuant[13].
- BNB: blockchain basata su Ethereum che ottima per la scalabilità, essa è usato come base per servizi di binance e di sevizi terzi basati sul suo ecosistema. 'explorer blockchain' per prendere i dati bscscan[11], BSCTrace[14].
- Solana: blockchain moto veloce a processare transazioni con bassi costi, con l'implementazione di token per creazione di ecosistemi. 'explorer blockchain' per prendere i dati Solscan[17], Solana-Beach[15].
- Litecoin: fork di bitcoin con più veloce a creare blocchi e verifica transazioni in modo più sicuro, dove è ancora possibile minare. 'explorer blockchain' per prendere i dati Blockchair[20].

In questa parte dei test viene utilizzato dei parametri dei componenti a valore medio e altri a che indica un valore puntuale. Nel considerare la casistica generale si possono avere come parametri delle immagini delle blockchain un valore massimo o valore minimo o valore medio o un valore puntuale (come un tempo o un numero di elementi o un valore).

6.2 test

Per testare l'utilizzo dell'algoritmo sviluppato ho utilizzato la libreria integrata per i test del web framework Django. Nella seguente parte andrò a descrivere la struttura

de test e la loro funzione, tutto riportato nelle seguenti tabelle.
Test componenti : inserimento e presa dal db

- Test test_data : presa dati dal database tramite la funzione "get_data_db" che la funzione prenda tutte le lista di queryset;
- Test_creazione_classe_inserimento : tese creazione della classe che le converte i dati del database in liste che converta tutte le queryset i componenti con gli attributi giusti e le dimensioni giuste;
- Test_matrice___vettore : test della funzione che crea matrici e vettori che è dentro alla classe che le converte il dato del database in liste tramite parametri che sono dentro componenti;

Funzioni : creazione funzioni obbiettivo

- get_elem : Crea la funzione obbiettivo nel caso del problema dello zaino degli elementi;
- get_c : Tramite i settaggi della funzione obbiettivo prende i dati dal db con la classe "ProcessData";

Test algoritmo
"test_numero_elementi_dati" : "ProcessData" avendo come argomenti settaggi di prova per verificare che sono effettivamente presi dati da db;
"test_algoritmo_elementi": Test dell'algoritmo nel caso degli elementi [13]:

- La funzione obbiettivo: Numero parametri presenti nei componenti delle immagine delle blockchain;
- Le variabili scelte sono: 'avgFee', 'avgTime', 'time', 'nAvgContract','nToken', 'nIntEvent', 'nInCount','nOutCount';

"test_algoritmo_costi" : Test dell'algoritmo ne caso dei costi (2):

- La funzione obbiettivo: 'avgFee';
- Le variabili scelte sono: 'avgTime', 'time', 'nAvgContract', 'nToken', 'nIntEvent', 'nInCount', 'nOutCount';

"test_algoritmo_velocita" : Test dell'algoritmo nel caso della velocità[14]:

- La funzione obbiettivo: 'avgTime';
- Le variabili scelte sono: 'avgFee', 'time', 'nAvgContract', 'nToken','nIntEvent', 'nInCount', 'nOutCount';

"test_algoritmo_velocita_costi" : Test dell'algoritmo nel caso della velocità e costi[13]:

- La funzione obbiettivo: 'avgTime','avgFee';
- Le variabili scelte sono:'time', 'nAvgContract', 'nToken', 'nIntEvent', 'nInCount', 'nOutCount';

6.3 risultati algoritmo

alla fine dell'algoritmo abbiamo fatto i test queste sono le casistiche nei casi *elementi*, *costi* , *velocità* , *velocità costi*.

[14]per creare la matrice c utilizza la funzione "get_elem" per generare il numero di elementi
[15]per prendere il settaggio della matrice c utilizza la finzione "get_c" per prendere i dati per la funzione obbiettivo.

Tabella 6.1: tabella risultati per tipo di blockchin

risultati	blockchains					tipo problema
	bitcoin	Ethereum	BNB	Solana	Litecoin	
elementi or-tools	52	29	12	32	21	max
elementi pulp	52	29	12	32	21	max
costi or-tools	0,15381232	0,163054945	0,268736842	0,2866578947	0,174	min
costi pulp	0,1538123275	0,163054944	0,268736843	0,275557887	$0,173\overline{9}$	min
velocità or-tools	71	225,9190476	118,3531646	96	86	min
velocità pulp	71	225,9190476	118,3531646	96	86	min
velocità e Costi or-tools	0,43531	0,31636	0,41136	$0,6\overline{4}$	0,16876	min
Velocità e costi or-tools	72,6	231,8	126,6	96,6	66,5	min

Tabella 6.2: tempi prove: tempi risolutori

prove	tot ms or-tools (wall time)	iterazioni in ms or-tools	numero iterazioni or-tools	tot ms pulp (wall time)	iterazioni in ms pulp	numero iterazioni pulp
elementi	15	1,667	9	25	1,0869565	23
costi	13	3,25	4	17	$1,41\overline{6}$	12
velocità	12	12	1	23,5	$1,95\overline{3}$	12
velocità e costi	21	10,5	2	0	0	0

Tabella 6.3: tabella risultati: blockchin scelte

prove	descrizione	blockchain migliore	
elementi	quanti elementi ci stanno all'interno del vincoloimposto; variabili: 0, $+\infty$	pulp: bitcoin 52,0	or-tools: bitcoin 52,0
	l risultati sono conformi con la linea di pensiero che mi ero fatta di blockchain di test; bitcoin, si perché quella migliore.		
costi	costi minori della blockchain per numero di elementi dove nella funzione obiettivo ho avgFee; variabili: 0, $+\infty$	pulp: bitcoin 0,15381	or-tools: bitcoin 0,15381
	l risultati sono conformi con la linea di pensiero che mi ero fatta di blockchain di test; bitcoin, si perché quella migliore.		
velocità	l'efficienza della blockchain per numero di elementi, dove nella funzione obiettivo ho AvgTime; variabili: 0, $+\infty$	pulp: bitcoin 71,0	or-tools: bitcoin 71,0
	l risultati sono conformi con la linea di pensiero che mi ero fatta di blockchain di test; bitcoin, si perché quella migliore		
velocità e costi	costi minori e l'efficienza della blockchain per numero di elementi, dove nella funzione obiettivo ho AvgTime, e avgFee; variabili: 0, $+\infty$	or-tools: Litecoin costi 0,16876 efficienza 66,5	
	l risultati sono conformi con la linea di pensiero che mi ero fatta di blockchain di test, Litecoin, sì perché è quella media		

Capitolo 7: conclusioni

A conclusione di questo elaborato successivamente di aver introdotto la blockchain messa in un database modellando i dati creando "l'immagine della blockchain". Questi dati scegliendo dei parametri per la ricerca della blockchain ottimale, esso è possibile usufruendo della metodologia che ho introdotto della ricerca operativa. Tutto questo implementato nell'algoritmo che utilizza librerie per risolvere il problema impostato in precedenza.

7.1 conclusioni finali

Prima di partire col progetto si trova la documentazione per introduzione al progetto e i suoi concetti come blockchain per avere una base sulla trattazione della problematica del "prouf of conpcet" del trattato.
Poi si passa alla Modellazione data-base secondo la documentazione trovata per costruire la base dell'introduzione e la ricerca di blockchain compatibili per la verifica del data-base. Creazione settaggio problema di ricerca operativa andando a trovare possibili modelli di problemi similare all'utilizzo pratico con i parametri presi dal data-base.
I modelli di problemi utilizzati per l'applicazione sono il problema della aziono e il problema dello zaino minimo ambe del tipo multidimensionale. Anche andando a considerare le varianti con che dipendo dai tipi di variabile e il la dimensione del vettore costi.
La parte iniziale del progetto comprende la creazione della repository github con la predisposizione del web framework Django dove è inserito li data-base contente le "immagine della blockchain".
Successivamente ho Sviluppo algoritmo sono partito con la creazione di una struttura delle classi per prendere i dati e settare le matrici all'interno delle librarie utilizzate per risoluzione. Le librerie che utilizzo implementano la metodologia basata sulla risoluzione di problemi di ricerca operativa. Ogni libreria ha diversi tipi di risolutori nel caso di utilizzo trattato in questo documento, nella libreria PULP utilizzo il risolutore COIN-OR e invece nella libreria OR-Tools ho utilizzato il risolutore SCIP. I risultati algoritmo vengono confrontati nei vari casi elementi, costi, velocità e infine costi e velocità in questo caso tale risolutore COIN-OR nella libreria PULP non supporta più di una funzione obbiettivo.
Le analisi dei risultati tra le "immagine della blockchain" viste nella parte dei test confronta nei singoli risultati di esse e la migliore nei singoli casi descritti nei test fatti. inoltre, si mette a confronto i vari risolutori selezionati con questi parametri totale ed iterazioni in ms ed anche numero di iterazioni.
Alla fine per tempistiche del tirocinio non si è potuto a sviluppare il passo successivo, quello che prendeva il settaggio del problema utilizzato come metrologia di risoluzione

l'intelligenza artificiale.

7.2 esperienza

In questa esperienza fatta nel tirocinio con IntegrityKEY ho provato lo sviluppo di una applicazione web di backed trovando difficoltà di sviluppo già passate da altri sviluppatori, anche quelle relative ad alcune materie non inserite nella carriera università. Queste difficoltà risolte problem solving applicato alla soluzione, ma anche grazie indirizzamento e l'aiuto del tutor in IntegrityKEY.
Durante il tirocinio svolto la mia conoscenza non idonea del web framework Django, mi ha fatto perdere molto tempo per lo sviluppo dell'algoritmo. In questa fase l'algoritmo creato però, non è ottimale per alcuni suoi processi per le librerie utilizzate nella ricerca della blockchain ottimale per la non sufficiente conoscenza del web framework. Nello sviluppo dell'algoritmo nel web framework ho sviluppato una buona struttura nelle classi in un "prouf of conpcet", con un ampio spettro per modifiche future.
Per le librerie utilizzate per la risoluzione utilizzano il metodo di ricerca operativa, quello più ottimale è utilizzare metrologia basate su l'intelligenza artificiale, non è stato sviluppato per motivi di tempistiche. Le librerie sono PULP con il risolutore CBC e la libreria OR-Tools che utilizza il risolutore SCIP, per lo scopo di confrontare con un'altra metodologia di calcolo della risoluzione sempre nello stesso campo. L'utilizzo di tali librerie con questa metodologia è veloce per processare i dati, non quanto se utilizzassi i metodi basati l'intelligenza artificiale. Ma se nel caso si si volesse cambiare il metodo di calcolo, si potrebbero riutilizzare classi o funzioni già esistenti per prendere i dati da database.

7.3 Futuro

Uno sviluppo futuro si può utilizzare l'intelligenza artificiale per la ricerca della blockchain ottimale, utilizzando libreria apposite per esempio tesorflow o pythorch. Un'altra possibile aggiunta può essere il carico delle "immagine della blockchain" nel data-base con un'interfaccia ed infine settare il metodo di analisi delle "immagine della blockchain" andando a sceglie i parametri.

bibliografia

documenetazione

[1] mauro bellini. *Blockchain: cos'è, come funziona e applicazioni oggi — blockchain4innovation.it*. https://www.blockchain4innovation.it/. 28 Set. 2022. URL: https://www.blockchain4innovation.it/esperti/blockchain-perche-e-cosi-importante/.

[2] *Blocchi | ethereum.org*. thereum.org. URL: https://ethereum.org/it/developers/docs/blocks/.

[3] *Block Chain — Bitcoin*. developer.bitcoin.org. URL: https://developer.bitcoin.org/reference/block_chain.html.

[4] Techskill Brew. *Layer 2 Blockchain scaling solutions: Channels, Sidechains, Rollups and Plasma (P art 16)*. https://medium.com/. 3 Mar. 2022. URL: https://medium.com/techskill-brew/layer-2-blockchain-scaling-solutions-channels-sidechains-rollups-and-plasma-part-16-79819e058ef6.

[5] Techskill Brew. *Merkle Tree in Blockchain (Part 5- Blockchain Series)*. https://medium.com/. 10 Gen. 2022. URL: https://medium.com/techskill-brew/merkle-tree-in-blockchain-part-5-blockchain-basics-4e25b61179a2.

[6] *FAQ: Generale: Django documentation*. Django Project. URL: https://docs.djangoproject.com/it/5.0/faq/general/#faq-mtv.

[7] Sepideh Mollajafari e Kamal Bechkoum. «Blockchain Technology and Related Security Risks: Towards a Seven-Layer Perspective and Taxonomy». In: *Sustainability* 15.18 (2023). ISSN: 2071-1050. DOI: 10.3390/su151813401. URL: https://www.mdpi.com/2071-1050/15/18/13401.

[8] FTA Online. *NFT (Non-Fungible Token): cosa sono e come funzionano - Borsa Italiana*. borsaitaliana. 28 Apr. 2022. URL: https://www.borsaitaliana.it/notizie/sotto-la-lente/nft-cosa-sono.htm.

[9] *Transactions — Bitcoin*. developer.bitcoin.org. URL: https://developer.bitcoin.org/reference/transactions.html.

[10] *Transazioni | ethereum.org*. ethereum.org. URL: https://developer.bitcoin.org/reference/transactions.html.

block chain

[11] *Binance (BNB) Blockchain Explorer*. bscscan. URL: https://bscscan.com/.

[12] *Blockchain data analysis Bitcoin*. CryptoQuant. URL: https://cryptoquant.com/asset/btc/summary.

[13] *Blockchain data analysis Ethereum*. CryptoQuant. URL: https://cryptoquant.com/asset/eth/summary.

[14] *BSCTrace - BNB Smart Chain (BSC) Blockchain Explore*. BSCTrace. URL: https://bsctrace.com/.

[15] *Dashboard | Solana Beach*. Solana Beach. URL: https://solanabeach.io/.

[16] *Ethereum (ETH) Blockchain Explorer*. etherscan. URL: https://etherscan.io/.

[17] *solana Blockchain Explorer*. Solscan. URL: https://solscan.io/.

[18] *Universal blockchain explorer and search engine - bitcoin*. Blockchair. URL: https://blockchair.com/bitcoin.

[19] *Universal blockchain explorer and search engine - ethereum*. Blockchair. URL: https://blockchair.com/ethereum.

[20] *Universal blockchain explorer and search engine - litecoin*. Blockchair. URL: https://blockchair.com/litecoin.

[21] *ViaWallet Blockchain Explorer - bitcoin*. ViaWallet. URL: https://explorer.viawallet.com/btc.

ricerca operativa

[22] Giancarlo Bigi e Antonio Frangioni e Giorgio Gallo e Stefano Pallottino e Maria Grazia Scutellà. *Appunti di Ricerca Operativa*. Dipartimento di Informatica Universit'a di Pisa. 10 Apr. 2024. URL: https://commalab.di.unipi.it/wp-content/uploads/2024/05/v1.0.0-240424.pdf.

[23] V. Boyer, M. Elkihel e D. El Baz. «Heuristics for the 0–1 multidimensional knapsack problem». In: *European Journal of Operational Research* 199.3 (2009), pp. 658–664. ISSN: 0377-2217. DOI: https://doi.org/10.1016/j.ejor.2007.06.068. URL: https://www.sciencedirect.com/science/article/pii/S0377221708003639.

[24] *Cbc Introduction*. coin-or. URL: https://coin-or.github.io/Cbc/intro.html.

[25] Pd Dott. M. Lübbecke Gerald Gamrath. «Generic Branch-Cut-and-Price». thesis. Dipartimento di Matematica Università Tecnica di Berlino Berlino, mar. 2010. URL: https://www.zib.de/userpage//gamrath/publications/gamrath2010_genericBCP.pdf.

[26] A. Ruberti e Massimo Roma. *Appunti dalle lezioni di Ricerca Operativa*. SAPIENZA Universit'a di Roma,Facolta di Ingegneria dell'Informazione ed Informatica e Statistica. 2022. URL: https://www.diag.uniroma1.it/roma/didattica/RO21-22/main.pdf.

[27] *Optimization with PuLP*. coin-or. URL: https://coin-or.github.io/pulp/.

[28] *Optimization with PuLP*. coin-or. URL: https://www.coin-or.org/projects/#ffs-tabbed-111.

[29] *OR-Tools | Google Developers*. google. URL: https://developers.google.com/optimization?hl=it.

[30] SCIP. *Introduction to SCIP*. SCIP. URL: https://www.scipopt.org/workshop2018/SCIP-Intro.pdf.

[31] *SCIP Doxygen Documentation*. SCIP. URL: https://scipopt.org/doc/html/index.php.