UNIVERSITÀ DEGLI STUDI DI MILANO
FACOLTÀ DI SCIENZE E TECNOLOGIE

DIPARTIMENTO DI INFORMATICA
GIOVANNI DEGLI ANTONI

Corso di Laurea triennale in Informatica

PATTERN DI ASSEGNAMENTO OTTIMIZZATI PER L'EFFICIENZA ENERGETICA IN EDGE COMPUTING

Relatore: Prof. Alberto Ceselli
Correlatore: Dr. Marco Premoli

Tesi di Laurea di:
Manuel Parati
Matr. 958584

ANNO ACCADEMICO 2021-2022

Ringraziamenti

Desidero ringraziare il Professor Alberto Ceselli e il Dottor Marco Premoli, rispettivamente relatore e correlatore di questa tesi, per avermi guidato nella sua realizzazione e per l'immensa disponibilità mostrata.
Vorrei esprimere inoltre un ringraziamento speciale alla mia famiglia e ai miei amici, che mi hanno sostenuto in questo percorso alleggerendomi i momenti più pesanti e spronandomi a dare sempre il massimo.

Indice

Capitolo 1

Introduzione

Negli ultimi anni le reti di comunicazione hanno subito una rapida evoluzione, abbracciando la virtualizzazione dei sistemi come metodologia per ottimizzare l'efficienza delle risorse fisiche e le spese necessarie per gestire la rete, oltre che aumentare la qualità dell'esperienza per l'utente finale. Un'area di ricerca particolarmente promettente è quella riguardante il Mobile Edge Computing (MEC) ([1]), che propone una versione distribuita del classico cloud computing, in cui i server applicativi sono istallati nell'edge della rete tramite un sistema di virtualizzazione, in modo da diminuire la distanza di rete tra gli utenti e i server, ed offrire quindi maggiore affidabilità e migliori prestazioni alle connessioni di rete, oltre che a ridurre la quantità di energia necessaria a gestire l'intera infrastruttura. Questo tipo di architettura migliora il servizio offerto all'utente riducendo drasticamente la latenza di connessione e virtualizzando le operazioni di calcolo sul nodo più vicino, permettendo di avere un basso impatto energetico sui dispositivi mobili e quindi poter eseguire applicazioni computazionalmente pesanti utilizzando device a basso consumo ([2]). Diversi scenari applicativi traggono vantaggio dall'utilizzo dell'infrastruttura MEC, come la realtà aumentata, i veicoli a guida autonoma, l'Internet tattile ([3]), o più in generale tutti quelli in cui sia necessaria elevata potenza di computazione e bassa latenza.

1.1 Obbiettivi

Questo lavoro propone un modello di orchestrazione ottima che permette di gestire in modo esplicito la gestione dell'energia da parte dei server edge, considerando lo scenario in cui ogni sito ospita impianti fotovoltaici di produzione e accumulo dell'energia. Questi impianti permettono di organizzare la computazione considerando, oltre al livello di servizio rivolto agli utenti, il costo energetico pagato

dovuto al calcolo, che sarà molto inferiore se si riesce ad utilizzare in maniera efficiente l'energia prodotta ed accumulata negli impianti fotovoltaici. Il modello è stato formulato secondo il formalisto della programmazione lineare, ottenendo una variante del *time-dependent generalized assignment problem*, successivamente implementato attraverso due algoritmi risolutivi euristici, che verranno approfondidi, analizzati e confrontati nel corso dei capitoli.

1.2 Struttura dell'elaborato

Il documento è diviso in sei capitoli:

1. **Introduzione**: introduzione al lavoro svolto;
2. **Modellazione del sistema**: fornisce una panoramica riguardo l'infrastruttura Mobile Edge Computing e illustra il funzionamento del modello di orchestrazione proposto;
3. **Modelli di ottimizzazione**: definisce i modelli di ottimizzazione utilizzati dagli algoritmi euristici, mostrando la loro rappresentazione matematica e descrivendo nel dettaglio le variabili, i vincoli e l'obbiettivo da raggiungere;
4. **Algoritmi euristici**: illustra i due algoritmi euristici proposti, mostrando il loro pseudocodice e descrivendo le operazioni che svolgono;
5. **Analisi sperimentale**: inizialmente presenta il setup e i dati utilizzati, successivamente descrive i test e mostra i risultati ottenuti;
6. **Conclusioni**: descrive dettagliatamente le conclusioni più rilevanti ottenute nella parte precedente.

Dopo l'ultimo capitolo, è presente l'appendice **Modelli di ottimizzazione aggiuntivi**, in cui sono presentati due modelli di ottimizzazione che propongono due diverse metodologie per frammentare un array in blocchi.

Capitolo 2

Modellazione del sistema

Questo capitolo fornisce una panoramica riguardo all'infrastruttura Mobile Edge Computing ed illustra il modello di orchestrazione proposto in questo lavoro.

2.1 L'infrastruttura Mobile Edge Computing

All'interno di un'infrastruttura Mobile Edge Computing (MEC) si trovano i cluster di virtualizzazione, spesso noti come 'MEC facility' o più semplicemente 'facility', e gli access point (AP). Le facility sono il luogo in cui avviene la virtualizzazione e sono composte dalle macchine virtuali (VM) su cui si eseguono le applicazioni degli utenti finali, mentre gli AP sono i dispositivi (per esempio le antenne wireless) a cui gli end point si collegano per poter ricevere il servizio. Ogni AP è associato ad una facility, a cui inoltra tutto il traffico che riceve: questo significa che ciascuna avrà in esecuzione nelle proprie VM tutte le applicazioni utilizzate dagli utenti connessi agli AP a lui associati. Ogni facility, per poter gestire il traffico che le viene inoltrato, deve utilizzare una quantità di energia direttamente proporzionale a tale domanda. Per questo motivo, possiedono dei pannelli fotovoltaici che generano una quantità di energia variabile nel tempo, dipendente dal numero di pannelli installati e dall'irraggiamento a cui sono sottoposti. L'energia prodotta può essere direttamente utilizzata oppure immagazzinata all'interno di alcune batterie, dotate di capacità limitata. Nel caso in cui l'energia disponibile, data dalla somma tra quella accumulata e quella generata, non basti a soddisfare la domanda, è possibile acquistarne altra ad un prezzo dipendente dalla facility e dall'istante temporale.

Data la natura mobile degli end point, che sono per esempio smartphone o laptop, il traffico a cui sono sottoposti gli AP varia nel tempo e di conseguenza cambia la domanda rivolta alle facility. Per questo motivo l'assegnamento viene effettuato dinamicamente, e lo strumento incaricato di svolgere tale compito è l'orchestratore, che implementa la logica definita dal modello di orchestrazione. Le azioni

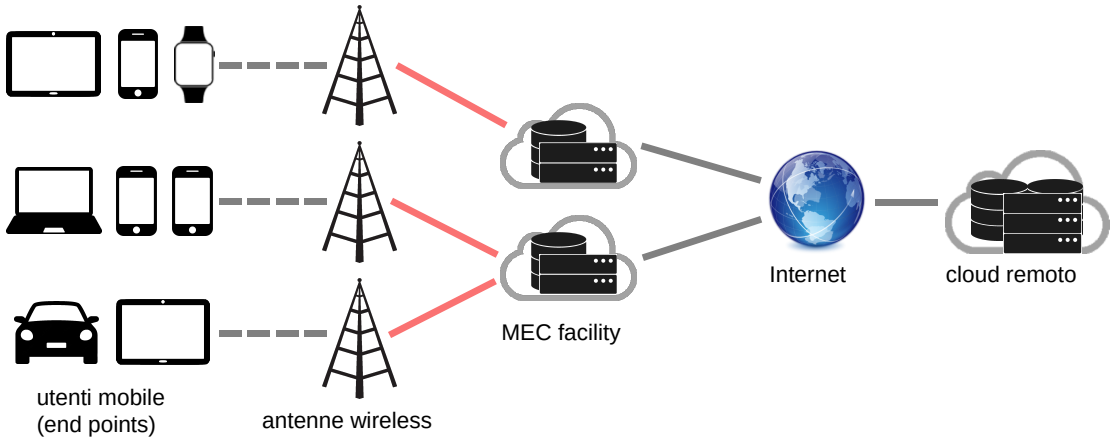

Figura 1: Esempio di un'infrastruttura Mobile Edge Computing (MEC).

che svolge vengono chiamate 'orchestrazioni' o 'switch' e consistono nell'assegnare un AP ad una facility diversa da quella attuale, provocando il ridimensionamento della potenza delle VM in termini del numero di processori e memoria disponibile, e la migrazione del loro stato. Il ridimensionamento è dovuto alla variazione di domanda da gestire, mentre la migrazione dello stato è necessaria per avere in esecuzione in ogni facility le applicazioni utilizzate dagli utenti connessi agli AP che le sono assegnati. Il costo prodotto dalla migrazione prende il nome di 'costo di migrazione'.

Nella figura 1 è presente un'infrastruttura MEC composta da due facility e tre AP, a cui sono collegati diversi dispositivi mobili di vario genere. Si noti come ogni dispositivo invii il proprio traffico all'AP a cui è collegato, e come tutto il traffico ricevuto da ciascuno sia inoltrato alla stessa facility. Si può osservare, per esempio, come all'AP nel mezzo siano connessi un laptop e due smartphone, mentre a quello in basso un'automobile smart ed un tablet. Nella figura, i due AP sono assegnati alla stessa facility (le linee rosse indicano gli assegnamenti) che riceverà e dovrà quindi gestire il traffico di tutti e cinque i dispositivi. Si noti infine come sia presente un cloud centralizzato che gestisce e sincronizza le varie facility della rete.

2.2 Modello di orchestrazione proposto

Il modello di orchestrazione proposto rappresenta una variante di quello presentato negli articoli [4] e [5], in cui nell'effettuare le scelte di orchestrazione viene esplicitamente considerato l'utilizzo e la gestione dell'energia da parte dei siti edge.

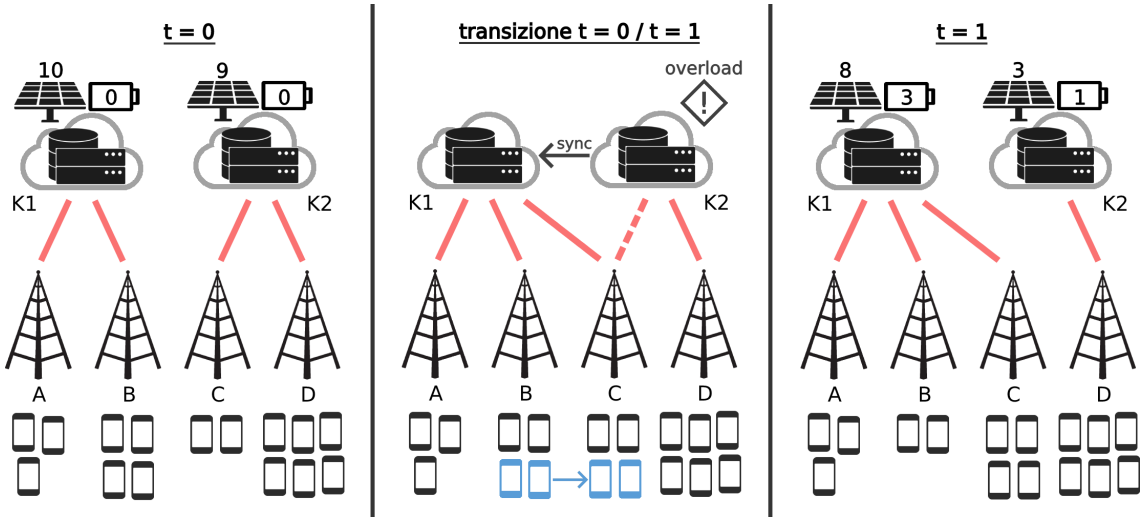

Figura 2: Esempio funzionamento del modello di orchestrazione.

Come illustrato in [5], all'interno del modello viene introdotta una discretizzazione temporale che permette di effettuare migrazioni solo in determinati istanti, per esempio ogni 15 minuti. Questa scelta è ragionevole, dato che in caso contrario si pagherebbe un costo troppo elevato in termini di migrazione e spese generali dovute alla gestione del traffico. L'orizzonte temporale viene quindi rappresentato da un insieme di fascie temporali, chiamate anche 'time-slot', della stessa durata.

L'obbiettivo del modello è quello di stabilire un piano degli assegnamenti che permetta di raggiungere un ragionevole compromesso tra: ottenere una buona qualità del servizio (cioè bassa latenza per l'utente finale), costi di migrazione contenuti e adeguata gestione energetica nelle facility. Nell'effettuare le scelte, bisogna tenere in considerazione che la domanda degli AP e l'energia prodotta varia nel corso del tempo, e inoltre che ogni facility possiede un limite massimo di domanda che è in grado di soddisfare simultaneamente. Si supponga inoltre che le batterie abbiano capacità limitata e che per gestire una unità di computazione sia necessaria una unità di energia.

2.2.1 Esempio del funzionamento

Una semplice applicazione di esempio è presentata nella figura 2. Come si può osservare, è presente una rete MEC con due facility (`K1` e `K2`) e quattro AP (`A`, `B`, `C`, `D`) considerata in due time-slot consecutivi (`t`=0, 1). Gli end point connessi agli AP sono rappresentati con dei piccoli rettangoli, e si supponga che inizialmente le batterie siano vuote in entrambe le facility. Nel primo time-slot (`t`=1) gli AP

A e B sono associati alla facility K1 mentre C e D a K2, quindi tutti gli utenti che sono connessi ad A e B avranno le proprie applicazioni in esecuzione su una VM presente in K1, mentre quelli connessi a C e D le avranno in K2. Questi assegnamenti permettono di avere una buona latenza, dato che ogni AP è connesso alla facility più vicina, ma anche una buona gestione energetica, in quanto nella facility K1 vengono prodotte 10 unità di energia ed utilizzate solo 7, mentre in K2 se ne producono 9 e utilizzano 8. In entrambi i casi avanzano delle unità (3 in K1 e 1 in K2) che vengono immagazzinate nelle batterie per poter essere spese nel prossimo time-slot. Successivamente (transizione t=0/t=1) due utenti connessi all'AP B si spostano e si agganciano a C: a questo punto K2 riceve una richiesta di domanda eccessiva che non riesce a gestire, e di conseguenza è necessario effettuare un'orchestrazione per ribilanciare il traffico. Questo comporta l'assegnare C a K1, ridimensionare le VM (aumentarne la potenza in K1 e diminuirla in K2) e sincronizzare le facility, trasferendo lo stato delle VM riguardanti tutti gli utenti connessi a C da K2 a K1. In un contesto reale, il modello deve prevenire una situazione di questo tipo, effettuando orchestrazioni preventive che tengano in considerazione la futura domanda dei vari AP, dato che questa circostanza va ad inficiare la qualità complessiva del servizio. Nel secondo time-slot (t=1) si avranno quindi gli AP A, B e C assegnati alla facility K1, mentre D a K2. Dal punto di vista energetico, la facility K1 riesce a soddisfare la domanda di traffico (9 unità) utilizzando l'energia presente nella batteria e quella prodotta in questo time-slot (8 + 3 unità complessive). Per quanto riguarda invece K2, l'energia disponibile (3 + 1 unità) non è sufficiente, e quindi è necessario acquistare 2 ulterione unità. Da notare come l'energia potrebbe avere costi diversi nel corso dei time-slot, e di conseguenza potrebbe risultare conveniente acquistarla quando il costo è basso per immagazzinarla ed utilizzarla successivamente.

2.2.2 Implementazione

Il comportamento del modello di orchestrazione fin qui descritto viene formulato secondo il formalismo della programmazione lineare misto-intera (MILP), ottenendo una variante del *time-dependent generalized assignment problem* descritta nella sezione 3.1. La prima opzione considerata è stata rivolvere direttamente il modello di programmazione matematica usando solver general purpose, in modo da determinare la soluzione ottima del problema, ma data la sua complettità, al crescere della dimensione dell'istanza, i tempi di calcolo aumentano fino a diventare proibitivi. Per questo motivo vengono successivamente proposti due algoritmi risolutivi eutistici, in modo da ottenere una risoluzione computazionalmente più efficiente. Nella sezione 4.1 viene presentata la prima idea euristica, che mira ad abbassare la complessità semplificando il modello utilizzato, mentre nella sezione 4.2 la seconda, che aggrega i time-slot dell'istanza per ottenerne una più facilmente risolvibile.

Capitolo 3

Modelli di ottimizzazione

In questo capitolo vengono introdotti i modelli di ottimizzazione utilizzati nel corso dell'elaborato. Nel primo sottocapitolo si illustra il modello di assegnamento dinamico, che descrive il problema presentato, mentre nei due successivi quelli utilizzati dagli algoritmi euristici: il modello di assegnamento statico ed il modello che effettua l'assegnamento utilizzando istanze composte da un solo time-slot.

3.1 Modello di assegnamento dinamico

Il 'modello di assegnamento dinamico' è un modello di programmazione lineare misto-intera (MILP) che, considerata un'infrastruttura MEC, è in grado di definire dinamicamente gli assegnamenti AP-facility all'interno di un arco temporale in cui sono noti i livelli di traffico rivolti agli AP e l'energia prodotta dalle facility. Con 'dinamicamente' si intende la possibilità di assegnare lo stesso AP a facility differenti nel corso dei time-slot, permettento di bilanciare nel tempo i flussi di traffico da gestire. Questo migliora la latenza complessiva e la gestione energetica attuata, ma provoca la necessità di dover gestire i vari switch e pagare i costi di migrazione. L'output del modello è un piano degli assegnamenti: viene indicato in ogni time-slot a quale facility della rete deve indirizzare il traffico ogni AP. Di conseguenza è possibile estrarre anche il piano delle orchestrazioni, che corrisponde ad un valore booleano per ogni time-slot, per ogni AP e per ogni coppia di facility, che assume come valore *vero* se in quell'instante temporale l'AP effettua uno switch tra le facility considerate, altrimenti *falso*.

Dati. Data un'infrastruttura MEC, siano A l'insieme degli AP e K l'insieme delle facility che la compongono. Si supponga inoltre di avere un orizzonte temporale discretizzato in un insieme T di time-slot. Vengono indicati, per ogni AP $i \in A$, con d_i^t la quantità di traffico a cui i è sottoposto al time-slot $t \in T$, e con $m_{i,k}$ la distanza

fisica tra i e la facility $k \in K$. La distanza di rete tra due nodi $j, k \in K \times K$ è invece denotata con l_{jk}: tale valore è direttamente proporzionale alla loro distanza fisica ed alla latenza di rete, includendo anche il tempo necessario a processare i pacchetti all'interno dei nodi intermedi. Data una facility $k \in K$, siano: C_k la quantità di domanda che riesce a gestire contemporaneamente, G_k la capienza complessiva delle sue batterie, p_k l'energia inizialmente presente all'interno delle batterie, e_k^t l'energia che produce nel time-slot $t \in T$, e c_k^t il prezzo a cui può acquistare l'energia nel time-slot $t \in T$.

Variabili. Siano x_{ik}^t variabili binarie che assumono valore 1 nel caso in cui al time-slot $t \in T$ l'AP $i \in A$ è assegnato alla facility $k \in K$, altrimenti 0. Le variabili y_{ijk}^t sono anch'esse binarie, e valgono 1 se l'AP $i \in A$ viene associato alla facility $j \in K$ al tempo $t-1$ ed alla facility $k \in K$ al tempo t, altrimenti 0. Da notare come queste variabili indicato le orchestrazioni da effettuare. Per ogni facility $k \in K$ e time-slot $t \in T$, siano: g_k^t l'energia residua della facility k al tempo t (quindi quella che viene immagazzinata all'interno delle batterie e disponibile nel time-slot $t+1$), v_k^t l'energia utilizzata dalla facility k al tempo t, e z_k^t la quantità di energia acquistata dalla facility k al tempo t.

Modello. Di seguito la rappresentazione matematica del modello.

$$\min \quad \alpha \sum_{t\in T}\sum_{i\in A}\sum_{\substack{(j,k)\in \\ K\times K}} d_i^t l_{jk} y_{ijk}^t + \beta \sum_{t\in T}\sum_{i\in A}\sum_{k\in K} d_i^t m_{ik} x_{ik}^t + \gamma \sum_{t\in T}\sum_{k\in K} c_k^t z_k^t \tag{3.1.1}$$

$$\text{s.t.} \quad v_k^t = \sum_{i\in A} d_i^t x_{ik}^t \qquad \forall k \in K, \forall t \in T \tag{3.1.2}$$

$$\sum_{k\in K} x_{ik}^t = 1 \qquad \forall i \in A, \forall t \in T \tag{3.1.3}$$

$$x_{ik}^t = \sum_{l\in K} y_{ilk}^t \qquad \forall i \in A, \forall k \in K, \forall t \in T \setminus \{1\} \tag{3.1.4}$$

$$x_{ik}^t = \sum_{l\in K} y_{ikl}^{t+1} \qquad \forall i \in A, \forall k \in K, \forall t \in T \setminus \{T\} \tag{3.1.5}$$

$$z_k^1 + e_k^1 + p_k \geq v_k^1 + g_k^1 \qquad \forall k \in K \tag{3.1.6}$$

$$z_k^t + e_k^t + g_k^{t-1} \geq v_k^t + g_k^t \qquad \forall k \in K, \forall t \in T \setminus \{1\} \tag{3.1.7}$$

$$g_k^t \leq G_k \qquad \forall k \in K, \forall t \in T \tag{3.1.8}$$

$$v_k^t \leq C_k \qquad \forall k \in K, \forall t \in T \tag{3.1.9}$$

$$x_{ik}^t \in \{0,1\} \qquad \forall i \in A, \forall k \in K, \forall t \in T \tag{3.1.10}$$

$$y_{ik'k''}^t \in \{0,1\} \qquad \forall i \in A, \forall k', k'' \in K, \forall t \in T \tag{3.1.11}$$

$$x, y, g, v, z \geq 0 \tag{3.1.12}$$

Obbiettivo. La funzione obbiettivo (3.1.1) ha lo scopo di ottenere la soluzione che possiede il miglior bilanciamento tra ottenere buona latenza per l'utente finale e costi di migrazione e di acquisto dell'energia contenuti. In particolare, si vuole minimizzare la somma tra (nell'ordine) i costi complessivi di migrazione, la latenza complessiva e il costo globale di acquisto dell'energia, dove i parametri α, β e γ indicano quanto peso dare a ciascun componente.

Vincoli. I vincoli 3.1.2 determinano il valore delle variabili v_k^t: la domanda a cui è sottoposta la facility $k \in K$ al tempo $t \in T$ è data dalla somma del traffico proveniente da tutti gli AP che le sono assegnati in quel time-slot. I vincoli 3.1.3 impongono ogni AP ad essere assegnato esattamente ad una facility per time-slot. I vincoli 3.1.4 e 3.1.5 collegano le variabili x alle y, e servono a gestire il flusso degli switch: quando x_{ik}^t vale 0 significa che l'AP i non è associato alla facility k e di conseguenza non permettono di effettuare nessuna orchestrazione, mentre nel caso contrario impongono di effettuarne esattamente una oppure mantenere i associato a k. I vincoli 3.1.6 e 3.1.7 gestiscono il flusso dell'energia: impongono che per ogni time-slot e per ogni facility, la quantità di energia disponibile (data dalla somma tra quella prodotta, quella presente nelle batterie e quella acquistata) sia non minore della somma tra la quantità utilizzata e quella che viene immagazzinata nelle batterie. I vincoli 3.1.8 e 3.1.9 limitano superiormente i valori che possono assumere rispettivamente le variabili g e v, facendo in modo che non venga sforata la capienza delle batterie e la capacità delle facility.

3.2 Modello di assegnamento statico

Il 'modello di assegnamento statico' è un modello di tipo MILP che ha lo stesso scopo del modello di assegnamento dinamico (3.1), ma con la differenza che non permette di effettuare switch. Ogni AP sarà quindi assegnato alla stessa facility per tutto l'arco temporale considerato, e non attuando migrazioni, il costo di migrazione complessivo sarà 0. In generale, utilizzare un approccio di assegnamento dinamico fornisce minore latenza e migliore efficienza energetica complessive rispetto ad uno statico, in quanto è possibile adattare gli assegnamenti alla variazione del traffico richiesto dai vari AP ed alla quantità di energia prodotta dalle facility. La formulazione del problema statico richiede però un numero inferiore di variabili e vincoli, con conseguente diminuzione della complessità combinatoria. L'output del modello in questo caso sarà una lista di assegnamenti validi per tutti i time-slot.

Dati. I dati su cui opera sono gli stessi del modello dinamico, si veda la sezione 3.1 per una panoramica completa.

Variabili. Siano x_{ik} variabili binarie che assumono valore 1 se l'AP $i \in A$ viene assegnato alla facility $k \in K$, 0 altrimenti. Per ogni facility $k \in K$ e per ogni time-slot $t \in T$, siano inoltre g_k^t l'energia residua della facility k al tempo t, v_k^t l'energia utilizzata dalla facility k al tempo t e z_k^t la quantità di energia acquistata dalla facility k al tempo t. Come si può notare, il set di variabili viene notevolmente ridotto rispetto al modello dinamico, in quanto imponendo assegnamenti statici non è più necessario gestire il flusso degli switch e memorizzare gli assegnamenti per ogni time-slot. Le variabili y diventano quindi superflue, e per determinare gli assegnamenti è sufficiente una variabile x per ogni coppia di AP e facility.

Modello. Di seguito la rappresentazione matematica del modello.

$$\min \quad \beta \sum_{t \in T} \sum_{i \in A} \sum_{k \in K} d_i^t m_{ik} x_{ik} + \gamma \sum_{t \in T} \sum_{k \in K} c_k^t z_k^t \tag{3.2.1}$$

$$\text{s.t.} \quad v_k^t = \sum_{i \in A} d_i^t x_{ik} \qquad \forall k \in K, \forall t \in T \tag{3.2.2}$$

$$\sum_{k \in K} x_{ik} = 1 \qquad \forall i \in A \tag{3.2.3}$$

$$z_k^1 + e_k^1 + p_k \geq v_k^1 + g_k^1 \qquad \forall k \in K \tag{3.2.4}$$

$$z_k^t + e_k^t + g_k^{t-1} \geq v_k^t + g_k^t \qquad \forall k \in K, \forall t \in T \setminus \{1\} \tag{3.2.5}$$

$$g_k^t \leq G_k \qquad \forall k \in K, \forall t \in T \tag{3.2.6}$$

$$v_k^t \leq C_k \qquad \forall k \in K, \forall t \in T \tag{3.2.7}$$

$$x_{ik} \in \{0, 1\} \qquad \forall i \in A, \forall k \in K \tag{3.2.8}$$

$$x, g, v, z \geq 0 \tag{3.2.9}$$

Obbiettivo. La funzione obbiettivo (3.2.1) ha lo scopo di determinare la soluzione che rappresenta il miglior compromesso tra ottenere una bassa latenza complessiva e un prezzo totale di acquisto dell'energia contenuto. I parametri β e γ corrispondono ai pesi da assegnare a ciascuna componente. Da notare come l'unica differenza con la funzione del modello normale (3.1.1) sia l'assenza della prima parte, riguardante i costi di migrazione.

Vincoli. I vincoli 3.2.2 determinano il valore delle variabili v_k^t: l'energia utilizzata dalla facility k al tempo t è data dalla somma del traffico proveniente da ogni

AP i che le è assegnato; mentre i vincoli 3.2.3 impongono ad ogni AP di essere associato esattamente ad una facility. I vincoli 3.2.4 e 3.2.5 riguardano la gestione ed il flusso dell'energia e sono equivalenti ai 3.1.6 e 3.1.7 del modello dinamico, mentre quelli rappresentati dalle disequazioni 3.2.6 e 3.2.7 impongono di non eccedere rispettivamente la capacità delle batterie e la capienza delle facility, e sono l'equivalente dei 3.1.8 e 3.1.9.

3.2.1 Variante con quadro energetico completo

Il modello presentato non garantisce di tracciare il reale flusso di energia che avviene all'interno delle facility, dato che per come sono posti i vincoli 3.2.4 e 3.2.5 ci potrebbero essere delle quantità di energia residua che non vengono accumulate, in quanto non utilizzate nei time-slot successivi. Nel determinare la soluzione ottima questo non causa problemi, ma quando si necessita di un quadro realistico della gestione energetica è necessario apportare alcune modifiche. In particolare, si deve immagazzinare ogni volta tutta l'energia possibile, e quindi i vincoli 3.2.4 e 3.2.5 vanno rispettivamente riformulati come:

$$z_k^1 + e_k^1 + p_k = v_k^1 + g_k^1 + s_k^1 \qquad \forall k \in K \tag{3.2.10}$$

$$z_k^t + e_k^t + g_k^{t-1} = v_k^t + g_k^t + s_k^t \qquad \forall k \in K, \forall t \in T \setminus \{1\} \tag{3.2.11}$$

dove s_k^t sono variabili non negative che indicano la quantità di energia non accumulabile nella facility $k \in K$ al tempo $t \in T$ a causa della capienza limitata delle batterie. La quantità di energia disponibile (data dalla somma tra quella acquistata, quella prodotta e quella presente nelle batterie) è posta quindi uguale alla somma tra la quantità utilizzata, la quantità accumulata e la quantità non accumulabile. A questo punto, nella funzione obbiettivo (3.2.1) è necessario minimizzare anche la somma complessiva dell'energia non accumulabile ($\sum_{t\in T}\sum_{k\in K} s_k^t$), trasformandola in:

$$\min \quad \beta \sum_{t\in T}\sum_{i\in A}\sum_{k\in K} d_i^t m_{ik} x_{ik} + \gamma \sum_{t\in T}\sum_{k\in K} c_k^t z_k^t + \sum_{t\in T}\sum_{k\in K} s_k^t \tag{3.2.12}$$

in modo da ottenere nelle variabili g il valore più alto possibile, e quindi immagazzinare ogni volta tutta la quantità possibile. Di seguito è mostrata la rappresentazione matematica completa:

$$\min \quad \beta \sum_{t\in T}\sum_{i\in A}\sum_{k\in K} d_i^t m_{ik} x_{ik} + \gamma \sum_{t\in T}\sum_{k\in K} c_k^t z_k^t + \sum_{t\in T}\sum_{k\in K} s_k^t \tag{3.2.12}$$

$$\text{s.t.} \quad v_k^t = \sum_{i\in A} d_i^t x_{ik} \qquad \forall k \in K, \forall t \in T \tag{3.2.2}$$

$$\sum_{k \in K} x_{ik} = 1 \qquad \forall i \in A \qquad (3.2.3)$$

$$z_k^1 + e_k^1 + p_k = v_k^1 + g_k^1 + s_k^1 \qquad \forall k \in K \qquad (3.2.10)$$

$$z_k^t + e_k^t + g_k^{t-1} = v_k^t + g_k^t + s_k^t \qquad \forall k \in K, \forall t \in T \setminus \{1\} \qquad (3.2.11)$$

$$g_k^t \leq G_k \qquad \forall k \in K, \forall t \in T \qquad (3.2.6)$$

$$v_k^t \leq C_k \qquad \forall k \in K, \forall t \in T \qquad (3.2.7)$$

$$x_{ik} \in \{0, 1\} \qquad \forall i \in A, \forall k \in K \qquad (3.2.8)$$

$$x, g, v, z, s \geq 0 \qquad (3.2.13)$$

Da notare come l'aggiunta delle variabili s renda questo modello leggermente più complesso rispetto a quello di riferimento, e che nel risolvere le stesse istanze si otterranno risultati della stessa qualità.

3.3 Modello di assegnamento singolo time-slot

Il 'modello di assegnamento singolo time-slot' è un modello di tipo MILP ottimizzato per definire gli assegnamenti all'interno di istanze composte da un singolo time-slot, e fornisce in output la facility a cui deve indirizzare il traffico ogni AP nel time-slot considerato.

Dati. Data un'infrastruttura MEC, siano A l'insieme degli AP e K l'insieme delle facility che la compongono. Per ogni AP $i \in A$, vengono indicati con d_i la quantità di traffico a cui i è sottoposto e con m_{ik} la distanza fisica tra i e la facility $k \in K$. La distanza di rete tra due facility $j, k \in K \times K$ è invece denotata con l_{jk}: tale valore è direttamente proporzionale alla loro distanza fisica ed alla latenza di rete, incluso il tempo di processamento dei pacchetti nei nodi intermedi. Data una facility $k \in K$, siano: C_k la quantità di domanda che riesce a gestire contemporaneamente, p_k l'energia inizialmente presente all'interno delle sue batterie, e_k l'energia che produce, e c_k il prezzo a cui può acquistare l'energia.

Variabili. Gli assegnamenti sono rappresentati da variabili binarie x_{ik}, che assumono come valore 1 se l'AP i viene assegnagno alla facility k, altrimenti 0. Inoltre, data una facility k, siano: z_k l'energia che acquista, v_k l'energia che utilizza e s_k l'energia residua al termine del time-slot.

Modello.

$$\min \quad \beta \sum_{i \in A} \sum_{k \in K} d_i m_{ik} x_{ik} + \gamma \sum_{k \in K} c_k z_k \qquad (3.3.1)$$

$$\text{s.t.} \quad v_k = \sum_{i \in A} d_i x_{ik} \qquad \forall k \in K \tag{3.3.2}$$

$$\sum_{k \in K} x_{ik} = 1 \qquad \forall i \in A \tag{3.3.3}$$

$$z_k + e_k + p_k = v_k + s_k \qquad \forall k \in K \tag{3.3.4}$$

$$v_k \leq C_k \qquad \forall k \in K \tag{3.3.5}$$

$$x_{ik} \in \{0, 1\} \qquad \forall i \in A, \forall k \in K \tag{3.3.6}$$

$$x, v, z, s \geq 0 \tag{3.3.7}$$

Obbiettivo. La funzione obbiettivo (3.3.1) ha lo scopo di definire gli assegnamenti che rispecchiano il miglior trade-off tra l'ottenere una bassa latenza complessiva ed avere una spesa energetica contenuta. Da notare come prendendo in cosiderazione un singolo time-slot, non sia possibile effettuare switch e di conseguenza l'obbiettivo rispecchia quello presente nel modello di assegnamento statico (3.2.1).

Vincoli. I vincoli 3.3.2 definiscono il valore delle variabili v_k, mentre i 3.3.3 impongono di assegnare ogni AP ad esattamente una facility. I vincoli 3.3.4 gestiscono l'utilizzo dell'energia, mentre i 3.3.5 definiscono un limite superiore alla quantità di traffico che ogni facility può gestire. Da notare come in questo caso, trattandosi di un singolo time-slot, non viene tenuta in considerazione la possibilità di immagazzinare l'energia nella batterie, e quindi tutta la quantità in eccesso viene immagazzinata dalle variabili s.

Capitolo 4

Algoritmi euristici

Nel corso del capitolo vengono presentati i due algoritmi risolutivi euristici proposti in questo lavoro.

4.1 Euristica 1: Semplificazione del modello

Il primo algoritmo euristico tenta di abbassare la complessità del problema andando a semplificare il modello di ottimizzazione utilizzato. In particolare, l'idea è quella di effettuare le orchestrazioni solo in determinati time-slot prestabiliti, così da poter suddividere l'istanza considerata in sottoistanze, all'interno delle quali ogni AP rimane associato alla stessa facility per tutto l'arco temporale, e poter risolvere ciascuna indipendentemente in modo statico usando solutori generici. I risultati ottenuti verranno poi ri-aggregati per formare quello dell'istanza iniziale. Nel risultato ottenuto, le orchestrazioni vengono quindi effettuate solo in fascie consecutive appartenenti a sottoistanze diverse, ed essendo una conseguenza dell'aggregazione dei risultati parziali, non considerano i costi di migrazione. Ciò significa che verrà premiata la latenza e la gestione energetica, ma andando a pagare un costo di gestione della rete maggiore a quello previsto dalla soluzione ottima. Idealmente, per avere un buon trade-off di ottimizzazione, l'istanza inziale non deve essere suddivisa in troppe parti, e di conseguenza è ragionevole voler pagare poche volte alti costi di migrazione, pur di avere nel lungo periodo una migliore latenza e gestione dell'energia.

4.1.1 Descrizione dell'algoritmo

L'algoritmo 1 mostra le operazioni necessarie ad implementare l'idea euristica esposta. Come si può osservare, richiede in input un'istanza del problema e ritorna in output il piano degli assegnamenti, rappresentato delle variabili x_{ik}^t, che assumono

Algoritmo 1 Pseudocodice euristica 1

Input: l'istanza da risolvere, composta dagli insiemi T, A, K e i dati C, G, d, l, m, e, c

Output: il piano degli assegnamenti risultante, quindi il valore delle variabili x

1: Siano x, g, v, z, s le variabili del risultato globale
2: $t \leftarrow -1$
3: costo_migrazione $\leftarrow 0$
4: $S \leftarrow \text{split}(T, A, K, C, G, d, l, m, e, c)$
5: **for all** $j \in \{0, \ldots, |S| - 1\}$ **do**
6: {Siano j considerati in ordine crescente}
7: $(\bar{T},\ \bar{A},\ \bar{K}, \bar{C}, \bar{G}, \bar{d}, \bar{l}, \bar{m}, \bar{e}, \bar{c}) \leftarrow S[j]$
8: **if** $j = 0$ **then**
9: $p_k \leftarrow 0\ \forall k \in K$
10: **else**
11: $p_k \leftarrow g_k^t\ \forall k \in K$
12: **end if**
13: $(\hat{x}, \hat{g}, \hat{v}, \hat{z}, \hat{s}) \leftarrow \text{calcola\_assegnamenti}(\bar{T},\ \bar{A},\ \bar{K}, \bar{C}, \bar{G}, \bar{d}, \bar{l}, \bar{m}, \bar{e}, \bar{c})$
14: **if** $j > 0$ **then**
15: **for all** $i \in A$ **do**
16: $k_{\text{prec}} \leftarrow \arg\max_{k \in K} x_{ik}^t$
17: $k_{\text{succ}} \leftarrow \arg\max_{k \in K} \hat{x}_{ik}^0$
18: **if** $k_{\text{prec}} \neq k_{\text{succ}}$ **then**
19: costo_migrazione $\leftarrow$ costo_migrazione $+\ d_i^t \cdot l_{k_{\text{prev}} k_{\text{succ}}}$
20: **end if**
21: **end for**
22: **end if**
23: $x \leftarrow x \cup \hat{x}$
24: $g \leftarrow g \cup \hat{g}$
25: $v \leftarrow v \cup \hat{v}$
26: $z \leftarrow z \cup \hat{z}$
27: $s \leftarrow s \cup \hat{s}$
28: $t \leftarrow t + |T|$
29: **end for**

valore 1 se al tempo $t \in T$ l'AP $i \in A$ è assegnato alla facility $k \in K$, altrimenti 0; dove T è l'insieme dei time-slot che compongono l'istanza, mentre A e K gli insiemi degli AP e delle facility dell'infrastruttura MEC.

Per fornire una migliore leggibilità, i nomi dei dati e delle variabili rispecchino quelli utilizzati nei modelli di assegnamento descritti nel capitolo precedente. Da notare come i dati delle sottoistanze siano rappresentati con una barra (es. $\bar{d}$) e quelli dell'istanza globale senza (es. d), e allo stesso modo, le variabili delle sottoistanze con un cappello (es. $\hat{x}$) e quelle del risultato globale senza (es. x).

Successivamente vengono illustrate le operazioni effettuate dall'algoritmo raggruppandole in tre parti: frammentazione dell'istanza, ottimizzazione delle sottoistanze e composizione del risultato globale.

Frammentazione dell'istanza

La prima operazione effettuata dall'algoritmo consiste nel frammentare l'istanza in input utilizzando la funzione *split* (riga 4), che restituisce una lista composta dalle sottoistanze ottenute. La frammentazione si effettua andando a identificare alcuni time-slot chiamati 'split point' e generando un'istanza per ogni sequenza compresa tra ciascuna coppia, mentenendo il loro ordine originale. Le sottoistanze generate sono tra loro indipendenti e operano sulla stessa infrastruttura MEC di partenza.

Ottimizzazione delle sottoistanze

Successivamente si vanno a risolvere una per una tutte le sottoistanze utilizzando la funzione *calcola_assegnamenti* (riga 13), che implementa la variante con quadro energetico completo del modello di assegnamento statico, descritto nella sottosezione 3.2.1. Per poter riprodurre il corretto flusso di energia tra istanti temporali consecutivi ma appartenenti a sottoistanze diverse, è necessario effettuare l'esecuzione mantenendo il corretto ordine cronologico, così da poter inserire nelle batterie delle facility al primo time-slot di ogni sottoistanza, la quantità di energia che era stata immagazzinata nell'ultimo istante di quella precedente. Questo comportamento è descritto nelle righe 8:12, dove con p_k viene indicata tale quantità, e all'inizio della prima sottoistanza le batterie vengono considerate vuote. Da notare come utilizzando il modello di assegnamento statico (invece che la sua variante) non sarebbe stato possibile determinare la reale quantità di energia immagazzinabile nell'ultimo istante temporale di ogni sottoistanza.

Composizione del risultato globale

Avendo risolto le sottoistanze seguendo l'ordine temporale, le variabili della soluzione globale si ottengono andando ad accumulare di volta in volta quelle parziali,

come è mostrato nella righe 23:27.

Per determinare la validità delle soluzioni ottenute, devono poter essere confrontabili con quelle ottime, e quindi è necessario calcolare il valore della funzione obbiettivo del modello di assegnamento dinamico (equazione 3.1.1) utilizzando le variabili della soluzione globale. Le variabili y, a differenze delle altre necessarie, non sono però disponibili, quindi i costi di migrazione complessivi vengono calcolati iterativamente dall'algoritmo e memorizzati nella variabile *costi_migrazione*. Il valore cercato sarà quindi dato da:

$$\alpha \cdot \text{costi\_migrazione} \; + \; \beta \sum_{t \in T} \sum_{i \in A} \sum_{k \in K} d_i^t m_{ik} x_{ik}^t \; + \; \gamma \sum_{t \in T} \sum_{k \in K} c_k^t z_k^t. \tag{4.1.1}$$

Effettuando gli switch solo tra due sottoistanze adiacenti, i costi di migrazione vengono calcolati controllando per ogni AP, se la facility a cui è associato nell'ultimo time-slot della sottoistanza precedente (k_{prec}) sia diversa da quella del primo time-slot della sottoistanza successiva (k_{succ}): in questo caso il costo da pagare è dato dalla quantità di traffico da migrare per la distanza tra le due facility coinvolte, ossia k_{prec} e k_{succ} (righe 15:21).

4.2 Euristica 2: Aggregazione dei time-slot

L'euristica proposta in questa sezione mira ad abbassare la complessità del problema semplificando l'istanza da risolvere. In particolare, l'idea è quella di frammentare l'arco temporale considerato e rappresentare ciascun frammento attraverso un singolo time-slot chiamato 'rappresentante', dato dall'aggregazione di tutti gli istanti appartenenti alla stessa parte. In questo modo si ottiene un'istanza composta solo dai rappresentanti, ed avendo quindi un numero inferiore di istanti temporali, diventa più veloce da risolvere. In ogni fascia dell'istanza originale, vengono quindi eseguiti gli stessi assegnamenti del suo rappresentante, e di conseguenza le migrazioni verranno effettuate solo tra frammenti consecutivi. A differenza dell'euristica precedente, nell'ottimizzazione vengono considerati i costi di migrazione, anche se non saranno utilizzati i valori dei dati reali ma quelli dei rappresentanti. Da notare come in alcune situazioni venga generato un risultato globale non valido, in quanto gli assegnamenti da attuare in ogni frammento sono calcolati utilizzando esclusivamente il traffico dell'istante rappresentante, che in genere è diverso negli altri time-slot, e di conseguenza si potrebbe assegnare ad una facility più domanda di quanta è in grado di gestirne.

Algoritmo 2 Pseudocodice euristica 2

Input: l'istanza da risolvere, composta dagli insiemi T, A, K e i dati C, G, d, l, m, e, c

Output: il piano degli assegnamenti risultante, quindi il valore delle variabili x

Siano x, g, v, z le variabili del risultato globale
$t \leftarrow 0$
costi_migrazione $\leftarrow 0$
$(L, \bar{T},\ \bar{A},\ \bar{K}, \bar{C}, \bar{G}, \bar{d}, \bar{l}, \bar{m}, \bar{e}, \bar{c}) \leftarrow$ aggrega$(T, A, K, C, G, d, l, m, e, c)$
$\hat{x} \leftarrow$ calcola_assegnamenti$(\bar{T},\ \bar{A},\ \bar{K}, \bar{C}, \bar{G}, \bar{d}, \bar{l}, \bar{m}, \bar{e}, \bar{c})$
for all $j \in \{0, \ldots, |L| - 1\}$ **do**
    for all $u \in \{0, \ldots, L[j] - 1\}$ **do**
        $x_{ik}^t \leftarrow \hat{x}_{ik}^j\ \forall i \in A, \forall k \in K$
        if $t = 0$ **then**
            $p_k \leftarrow 0\ \forall k \in K$
        else
            $p_k \leftarrow g_k^{t-1}\ \forall k \in K$
        end if
        $v_k^t = \sum_{i \in A} d_i^t x_{ik}^t\ \forall k \in K$
        $g_k^t \leftarrow \max\{0,\ \min\{p_k + e_k^t - v_k^t,\ G_k\}\}\ \forall k \in K$
        $z_k^t = \max\{0,\ v_k^t - (p_k + e_k^t)\}\ \forall k \in K$
        $t \leftarrow t + 1$
    end for
    if $j > 0$ **then**
        for all $i \in A$ **do**
            $k_{\text{prec}} \leftarrow \arg\max_{k \in K} x_{ik}^{t-L[j]-1}$
            $k_{\text{succ}} \leftarrow \arg\max_{k \in K} \hat{x}_{ik}^0$
            if $k_{\text{prec}} \neq k_{\text{succ}}$ **then**
                costo_migrazione $\leftarrow$ costo_migrazione $+\ d_i^{t-1-L[j]} \cdot l_{k_{\text{prev}} k_{\text{succ}}}$
            end if
        end for
    end if
end for

4.2.1 Descrizione dell'algoritmo

L'implementazione di questa euristica viene mostrata nell'algoritmo 2. Tale algoritmo, richiede in input un'istanza del problema e ritorna il piano degli assegnamenti, dato dal valore delle variabili binarie x_{ik}^t, che assumono valore 1 solo se al tempo $t \in T$ l'AP $i \in A$ è associato alla facility $k \in K$, dove T è l'insieme del time-slot mentre A e K quelli degli AP e delle facility. I dati e le variabili vengono indicati seguendo la convenzione adotatta nell'algoritmo precedente (consultare la sottosezione 4.1.1 per una panoramica completa). Successivamente viene illustrato il funzionamento dell'algoritmo.

Semplificazione e ottimizzazione dell'istanza

La prima operazione eseguita dall'algoritmo consiste nel semplificare l'istanza in input attraverso la funzione *aggrega* (riga 4), che oltre al risultato atteso fornisce un array L: l'elemento $L[j]$ indica il numero di time-slot rappresentati dal j-esimo istante dell'istanza semplificata. Successivamente si determina il piano degli assegnamenti su tale istanza utilizzando il modello di ottimizzazione dinamico (sezione 3.1), implementato dalla funzione *calcola_assegnamenti*. In questo caso il valore risultante delle variabili $\hat{g}$, $\hat{v}$ e $\hat{z}$ è superfluo.

Composizione del risultato globale

Il piano degli assegnamenti globale viene generato effettuando l'operazione inversa all'aggregazione sulle variabili x: tutti i time-slot di ogni frammento effettuano gli stessi assegnamenti determinati dal proprio rappresentante (riga 8, t rappresenta lo slot-time del risultato globale considerato attualmente).

Anche in questo caso, si vuole poter confrontare le soluzioni ottenute con quelle ottime, e di conseguenza è necessario calcolare il valore della funzione obbiettivo (equazione 3.1.1) utilizzando le variabili del risultato complessivo. A questo scopo, bisogna conoscere la quantità di energia acquistata da ogni facility in ogni istante, e per ottenere tali valori è necessario simulare l'intero scenario energetico (righe 14:16) considerando gli assegnamenti dettati dal piano globale. Dato un time-slot $t \in T$ e una facility $k \in K$, come prima cosa va calcolata la quantità di energia necessaria a gestire la domanda a cui è sottoposta k al tempo t, indicata con v_k^t e data dalla quantità totale di traffico rivolto agli AP che le sono assegnati:

$$v_k^t = \sum_{i \in A} d_i^t x_{ik}^t. \tag{4.2.1}$$

Successivamente bisgona determinare quanta energia viene accumulata, indicata con g_k^t e data dalla differenza tra quella disponibile e qualla utilizzata:

$$g_k^t = p_k + e_k^t - v_k^t \tag{4.2.2}$$

da notare come questo valore non possa essere negativo e non debba poter superare la capienza della batteria (G_k). Con p_k viene indicata la quantità di energia disponibile all'interno delle batterie all'inizio dell'istante temporale. Infine, si arriva a calcolare l'energia acquistata, data dalla differenza tra la quantità utilizzata e la quantità disponibile:

$$z_k^t = v_k^t - \left(p_k + e_k^t\right) \tag{4.2.3}$$

anche questo valore deve essere posto a non negativo, in modo da poter gestire correttamente i casi in cui la disponibilità sia maggiore dell'utilizzo. Da notare come nello scenario generato, l'energia acquistata debba essere direttamente utilizzata nell'istante attuale, senza poterla accumulare, e di conseguenza il risultato ottenuto potrebbe non rappresentare quello reale nei casi in cui il costo dell'energia nelle facility non sia costante nel tempo. A questo punto è possible calcolare il valore della funzione obbiettivo:

$$\alpha \cdot \text{costi\_migrazione} \; + \; \beta \sum_{t \in T} \sum_{i \in A} \sum_{k \in K} d_i^t m_{ik} x_{ik}^t \; + \; \gamma \sum_{t \in T} \sum_{k \in K} c_k^t z_k^t \tag{4.2.4}$$

dove *costi_migrazione* sono i costi di migrazione complessivi, ottenuti nello stesso modo dell'euristica precedente.

Capitolo 5

Analisi sperimentale

Nel corso di questo capitolo viene illustrata la fase di analisi sperimentale effettuata in questo lavoro, che mira a valutare i due algoritmi euristici proposti. Nella prima sezione viene illustrato il setup utilzzato per i test, successivamente viene fatta una panoramica sulle istanze utilizzate, ed infine si descrivono gli esperimenti e i risultati ottenuti.

5.1 Setup

L'intera fase di analisi sperimentale, che comprende il caricamento dei dati, la loro manipolazione, l'elaborazione dei risultati e le analisi svolte, è stata effettuata utilizzando il linguaggio di programmazione Python [1] (versione `3.8.13`), integrato con alcune sue librerie:

- Python-MIP [2]: utilizzato per implementare ed eseguire i modelli di programmazione lineare illustrati precedentemente;
- NumPy [3]: utilizzata per rappresentare e manipolare i dati da gestire;
- Pandas [4]; utilizzata per effettuare analisi ed elaborazione dei dati:
- Matplotlib [5]; utilizzata per generare i grafici.

Tutti i test sono stati eseguiti su una macchina con processore `Intel i7 8-core 4.00GHz` e `32GB` di memoria RAM, utilizzando il solutore Gurobi [6] (versione `8.0.1`

[1] `https://www.python.org/`.
[2] `https://www.python-mip.com/`.
[3] `https://numpy.org/`.
[4] `https://pandas.pydata.org/`.
[5] `https://matplotlib.org`.
[6] `https://www.gurobi.com/`.

`build v8.0.1rc0`) su sistema operativo Linux Ubuntu [7], considerando un gap di ottimalità dello 0.1% per determinare la soluzione ottima.

5.2 Panoramica sui dati utilizzati

Per condurre gli esperimenti, viene utilizzato il set di dati presentati nell'articolo [4], estratti da [6] e raffiguranti le richieste di traffico mobile nel mondo reale in un arco di due mesi su un'area geografica che si estende per oltre 2500 km^2. Come descritto, partendo dalla domanda vengono identificati 1419 AP (insieme A) e utilizzando le loro distanze euclidee si posizionano le 10 facility (insieme K), che compongono l'infrastruttura MEC. Le distanze di rete m_{ik} e l_{jk} vengono di conseguenza calcolate come distanze euclidee e arrotondate all'intero più vicino. Partendo da questa infrastruttura, sono state generate delle istanze in cui agli AP viene fornita una quantità di traffico casuale in modi diversi. In particolare, sono stati creati due dataset:

- **Dataset A**: è sintetico e mette alla prova gli algoritmi dal punto di vista puramente computazionale;
- **Dataset B**: è realistico e riproduce le caratteristiche principali dei dati di partenza;

Tutte le istanze hanno un arco temporale di una giornata, e si considerano fascie composte da 12, 24, 48 e 96 time-slot. Per ogni dataset sono state create cinque diverse istanze per ogni numero di time-slot.

5.2.1 Generazione dei dati energetici

In ogni istanza precedentemente descritta, è necessario aggiungere i dati energetici e_k^t, che indicano per ogni time-slot, la quantità di energia prodotta da ciascuna facility. Ogni facility $k \in K$ possiede una certa potenza di produzione energetica indicata con P_k, che indica il suo tasso di conversione dell'irraggiamento solare in energia; in un contesto reale può essere considerato come una misura della quantità di pannelli fotovoltaici installati. Tale valore dipende dalla posizione geografica in cui si trova la facility, ed in particolare cresce all'aumentare della distanza dal centro dell'area considerata, con l'idea che andando verso l'esterno, lo spazio disponibile per piazzare i pannelli sia sempre maggiore. Il set di dati considerato riguarda l'area metropolitana della città di Milano, e di conseguenza

[7] `https://www.ubuntu-it.org/`.

il centro è definito dalla posizione geografica del Duomo di Milano [8]. La costante P_k viene quindi definita come segue:

$$P_k = \mu f(k) \tag{5.2.1}$$

dove μ è il coefficiente che relaziona la distanza alla potenza produttiva, necessario per poter definire la quantità di energia complessivamente prodotta, mentre $f(k)$ è una misura della distanza di $k \in K$ dal punto centrale rispetto alle altre facility, e può essere definita in vari modi:

- **costante**:
$$f_C(k) = 1 \tag{5.2.2}$$
- **lineare**:
$$f_L(k) = \frac{d(k)}{\max\limits_{k \in K} d(k)} \tag{5.2.3}$$
- **quadratica**:
$$f_Q(k) = \frac{d(k)^2}{\max\limits_{k \in K} d(k)^2} \tag{5.2.4}$$
- **esponenziale**:
$$f_E(k) = \frac{e^{d(k)}}{e^{\max\limits_{k \in K} d(k)}} \tag{5.2.5}$$

con $k \in K$ e dove $d(k)$ indica la distanza euclidea tra la facility k e il centro dell'area. La quantità e_k^t di energia prodotta dalla facility $k \in K$ al tempo $t \in T$ è quindi data dalla radiazione solare a cui sono sottoposti i pannelli di k nell'istante t, indicato con r_k^r, e dalla potenza di conversione della facility:

$$e_k^t = P_k r_k^t. \tag{5.2.6}$$

I dati riguardanti l'irraggiamento solare sono stati ottenuti dal Photovoltaic Geographical Information System [9], ossia un portale gestito dall'Unione Europea dal quale è possibile estrarre informazioni riguardo alla radiazione solare e le prestazioni del sistema fotovoltaico. In particolare, considerando le coordinate di ogni sito MEC, sono stati estratti i dati storici dell'anno 2020 con cadenza oraria: all'interno di ogni mese viene eseguita la media rispetto ai giorni, così da ottenere l'irraggiamento medio di ogni giorno di ogni mese.

[8] Coordinate UTM (WGS84) del Duomo di Milano: `E: 514962`, `N: 5034533`.

[9] `https://re.jrc.ec.europa.eu/pvg_tools/en/`.

I dati c_k^t, rappresentanti il prezzo che la facility $k \in K$ deve pagare per acquistare una unità di energia al tempo $t \in T$, vengono considerati fissi nel tempo:

$$c_k^t = 1. \qquad \forall t \in T,\ \forall k \in K \tag{5.2.7}$$

Pur producendo una diversa quantità di energia, tutte le facility hanno batterie con la stessa capienza, che viene fissata al valore medio di domanda da gestire in ogni facility e in ogni time-slot:

$$G_k = \frac{\sum_{t \in T} \sum_{i \in A} d_i^t}{|T||K|}. \qquad \forall k \in K \tag{5.2.8}$$

Infine, resta da definire il valore del coefficiente μ:

$$\mu = \frac{\sum_{t \in T} \sum_{i \in A} d_i^t}{\sum_{t \in T} \sum_{k \in K} f(k) r_k^t}. \tag{5.2.9}$$

5.3 Test effettuati

La fase di test ha lo scopo di valutare le due euristiche proposte comparandole con il modello di assegnamento dinamico: viene quindi definito un set di istanze da calcolare, in modo da poter paragonare i risultati ottenuti e il tempo di calcolo necessario. Per poter avere una panoramica completa su tale confronto, è necessario utilizzare un insieme di istanze eterogeneo e considerare gli algoritmi in diverse configurazioni: i risultati ottenuti sono infatti fortemente influenzati dalle modalità utilizzate per dividere (euristica 1) e aggregare (euristica 2) i time-slot dell'istanza iniziale. In tutti i test eseguiti, i valori di α, β e γ sono posti a $0.\bar{3}$. Successivamente viene fornita una panoramica sul set delle istanze e le configurazioni degli algoritmi.

5.3.1 Set di istanze

Per testare gli algoritmi vengono utilizzate le istanze appartenenti al dataset B, così da poter valutare i risultati ottenuti su casi realistici ed osservare come si adattano alle variazioni di traffico nell'arco della giornata. Tra i vari formati, ci si concentra sulle istanze composte da 96 time-slot, che permettono di impegnarli maggiormente dal punto di vista computazionale e quindi valutarne al meglio le prestazioni. In ognuna di queste istanze vengono aggiunti i dati energetici, calcolati per ogni definizione della funzione f (equazioni 5.2.2, 5.2.3, 5.2.4, 5.2.5) e considerando l'irraggiamento nei mesi di dicembre, settembre e luglio, che rappresentano rispettivamente in media, il minore, medio e maggiore. In conclusione, il set sarà quindi composto da: 5 (istanze da 96 time-slot) $\times$ 4 (definizioni di f) $\times$ 3 (mesi irraggiamento) = 60 istanze.

5.3.2 Configurazione degli algoritmi

Entrambi gli algoritmi euristici considerati presentano un certo grado di libertà, dato dal modo in cui viene rispettivamente frammentata e semplificata l'istanza da computare. Tale scelta condiziona i tempi di calcolo e il risultato ottenuto, e di conseguenza per avere una panoramica il più completa possibile sul loro funzionamento, si testano diverse strategie. Per ottenere una migliore chiarezza espositiva, ogni configurazione viene identificata da un nome del tipo `E1_CONFIG_ALTRO`, dove la prima parte indica l'euristica, la seconda la configurazione e il resto varie specifiche.

Euristica 1

Le strategie considerate per suddividere l'istanza iniziale in sottoistanze sono le seguenti:

- `E1_UGUALE_T`: Effettuare la suddivisione facendo in modo che ogni sottoistanza abbia la stessa quantità di domanda complessiva da gestire, ottenendo sottoistanze ristrette quando si considerano fascie orarie con ampio uso della rete (generalmente durante il giorno) ed istanze più grandi quando l'uso è limitato (di solito durante la notte). In genere, non è possibile effettuare la suddivisione ottenendo esattamente in ciascuna la stessa domanda, e per questo si tenta di minimizzarne la loro differenza tramite il modello di ottimizzazione descritto nella sezione A.1, dove il vettore in input rappresenta l'arco temporale, ed ogni elemento contiene la quantità di domanda complessiva da gestire in quel time-slot. I blocchi restituiti indicano come deve essere eseguita la frammentazione, infatti per ciascuno viene creata una sottoistanza.

- `E1_LAD-LSD_T`: Suddividere l'istanza in modo da minimizzare la differenza di domanda all'interno di ogni sottoistanza, ottenendo quindi in ciascuna time-slot il più omogenei possibile tra loro. L'implementazione viene effettuata attraverso il modello di ottimizzazione presente nella sezione A.2, dove il vettore in input, come nel caso precedente, è formato dalla quantità di traffico complessiva presente in ogni time-slot, e i blocchi in output descrivono in che modo segmentare l'arco temporale. Per definire la differenza d_{ij} tra due elementi i e j, vengono considerati due modelli:
  - Least Absolute Deviation (LAD): $d_{ij} = |q_i - q_j|$;
  - Least Squared Difference (LSD): $d_{ij} = (q_i - q_j)^2$.

In entrambi i casi si esamina la suddivisione in 12, 24 e 48 sottoistanze, e la `T` nel nome va sostituita con tale valore.

Vengono inoltre considerati i due casi estremi:

- `E1_MIGRAZIONI_0`: La suddivisione viene effettuata in tante parti quanti sono i time-slot che compongono l'istanza, ottenendo sottoistanze formate da un singolo istante temporale. In questo modo le istanze vengono risolte senza considerare i costi di migrazione da pagare, che si suppongono essere 0, e di conseguenza gli assegnamenti avvantaggeranno la latenza e la gestione energetica ottenute. Avendo grandezza unitaria, le sottoistanze vengono risolte utilizzando il modello di assegnamento descritto nella sezione 3.3, creato ad hoc per questa tipologia.

- `E1_MIGRAZIONI_INF`: Utilizzare l'istanza completa senza effettuare suddivisioni. In questo caso, nell'eseguire l'ottimizzazione si considera di pagare un costo infinito a seguito di ogni switch: le orchestrazioni diventeranno quindi inefficienti e ogni AP rimarrà associato alla stessa facility per tutto l'arco temporale, a discapito della latenza e dell'efficienza energetica. In questo modo il problema perde la caratteristica della dinamicità. Da notare come, trattandosi di una singola istanza, non sia necessario utilizzare l'algoritmo 1, ma sia sufficiente il modello di assegnamento statico (sezione 3.2).

Euristica 2

In questo caso viene considerata una sola strategia per semplificare l'istanza.

- `E2_LAD-LSD_T`: L'istanza iniziale viene frammentata minimizzando la differenza di domanda all'interno di ogni frammento allo stesso modo delle configurazioni `E1_LAD_T` e `E1_LSD_T`, e per ciascuno viene scelto come rappresentante il time-slot mediano utilizzato nell'effettuare l'ottimizzazione. Anche in questo caso vengono considerate suddivisioni da 12, 24 e 48 time-slot, e la `T` va sostituita con tale valore.

5.4 Risultati ottenuti

In questa sezione vengono mostrati ed analizzati i risultati ottenuti dalle computazioni.

5.4.1 Analisi generale

Nella tabella 1 vengono presentati i valori e i tempi di esecuzione medi dei risultati ottenuti per ogni configurazione, dove con `MODELLO_DINAMICO` viene indicato il modello di assegnamento dinamico (sezione 3.1). La prima colonna indica la configurazione considerata, nella seconda è presente la media dei valori della funzione

configurazione	valore obbiettivo	tempo di esecuzione
MODELLO_DINAMICO	1.212566e+11	00:36:35.896
E1_LAD_12	1.253934e+11	00:00:04.780
E1_LAD_24	1.245708e+11	00:00:05.213
E1_LAD_48	1.252199e+11	00:00:06.583
E1_LSD_12	1.251684e+11	00:00:04.658
E1_LSD_24	1.246440e+11	00:00:05.304
E1_LSD_48	1.258861e+11	00:00:06.509
E1_MIGRAZIONI_0	1.261016e+11	00:00:09.125
E1_MIGRAZIONI_INF	1.422782e+11	00:00:21.114
E1_UGUALE_12	1.252572e+11	00:00:05.349
E1_UGUALE_24	1.257407e+11	00:00:05.510
E1_UGUALE_48	1.261911e+11	00:00:07.255
E2_LAD_12	1.225128e+11	00:00:47.196
E2_LAD_24	1.258975e+11	00:02:13.861
E2_LAD_48	1.362601e+11	00:06:47.127
E2_LSD_12	1.251709e+11	00:00:48.317
E2_LSD_24	1.254649e+11	00:02:39.567
E2_LSD_48	1.367933e+11	00:07:56.186

Tabella 1: Valori medi dei risultati ottenuti.

obbietto del modello di assegnamento dinamico (equazione 3.1.1) calcolata sui risultati ottenuti, ed infine è indicata la media dei tempi di esecuzione, nel formato: ore, minuti, secondi e millisecondi. Osservando la tabella, si nota come l'uso degli algoritmi euristici apporta una grossa riduzione del tempo di calcolo necessario a risolvere le istanze del set considerato, infatti partendo dal MODELLO_DINAMICO che impiega in media più di mezzora, le configurazioni della seconda euristica necessitano di pochi minuti, mentre quelle della prima alcuni secondi.

Nella figura 3 viene mostrato il peggioramento medio percentuale dei risultati ottenuti da ciascuna configurazione rispetto all'ottimo. Si nota immadiatamente come il range di peggioramento sia tra l'1% ed il 17.5% circa, e come la configurazione E2_LAD_12 generi i risultati migliori, a differenza della E1_MIGRAZIONI_INF che ottiene in media i peggiori. Le restanti hanno tutte un peggioramento che varia tra il 2.5% ed il 4%, a differenza delle configurazioni da 48 time-slot dell'euristica 2, che peggiorano l'ottimo di circa il 12.5%.

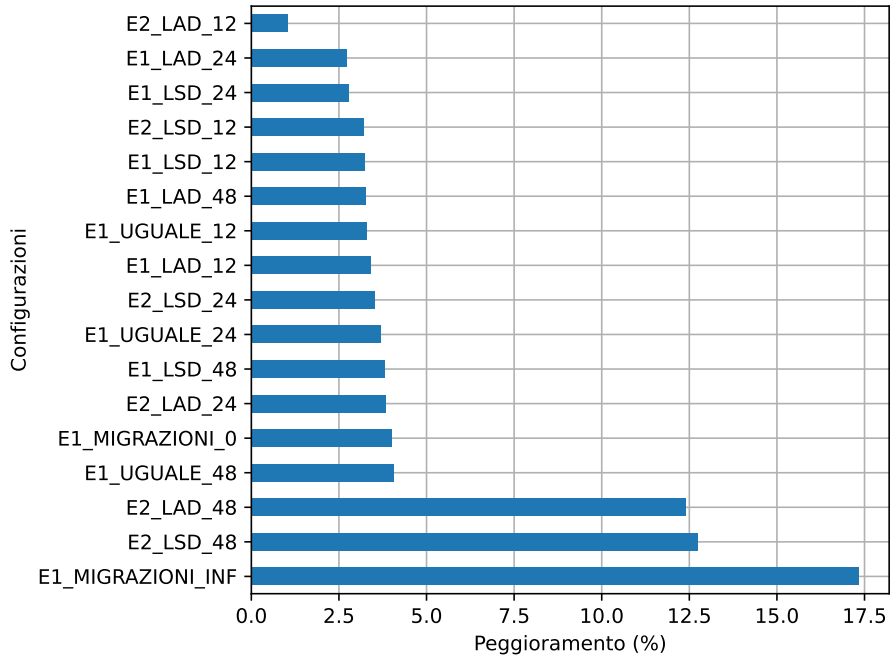

Figura 3: Peggioramento percentuale delle configurazioni rispetto ai valori ottimi.

5.4.2 Risultati euristica 1

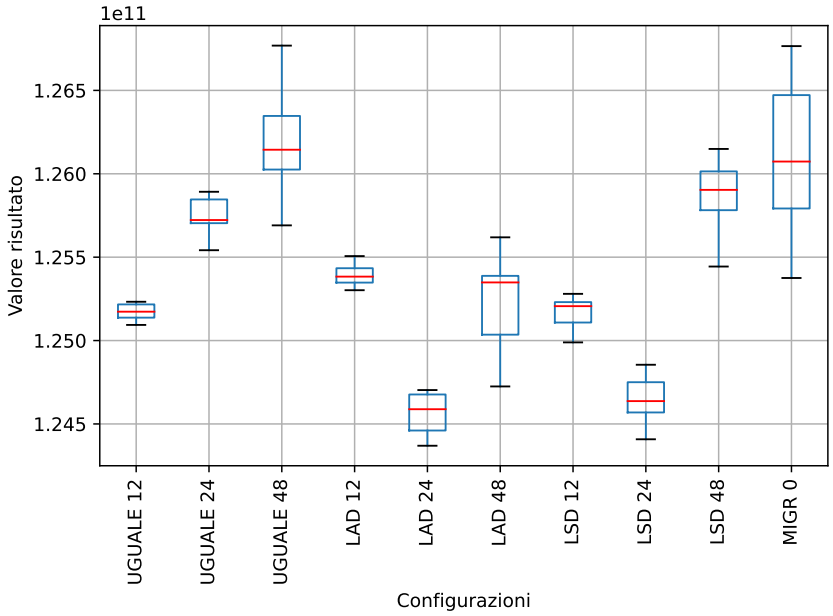

Figura 4: Distribuzione dei risultati ottenuti usando l'euristica 1.

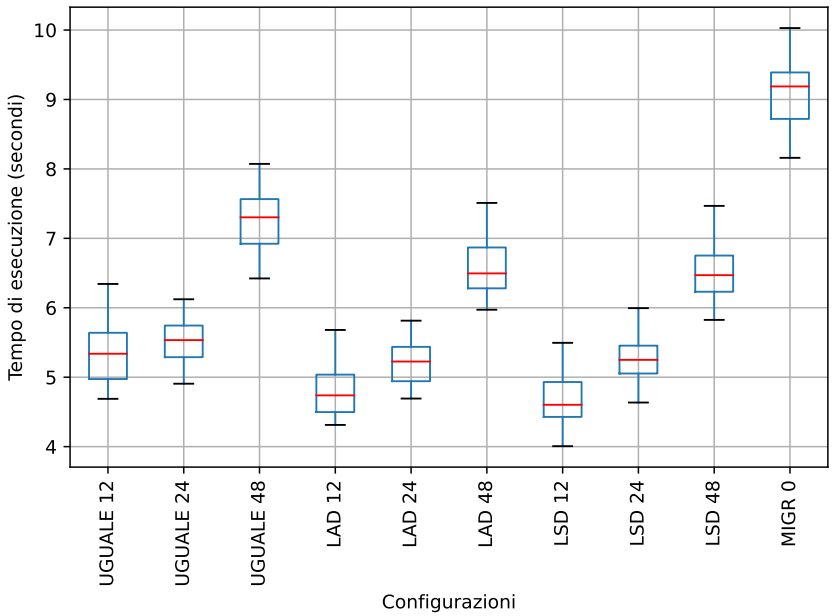

Figura 5: Distribuzione dei tempi di calcolo ottenuti usando l'euristica 1.

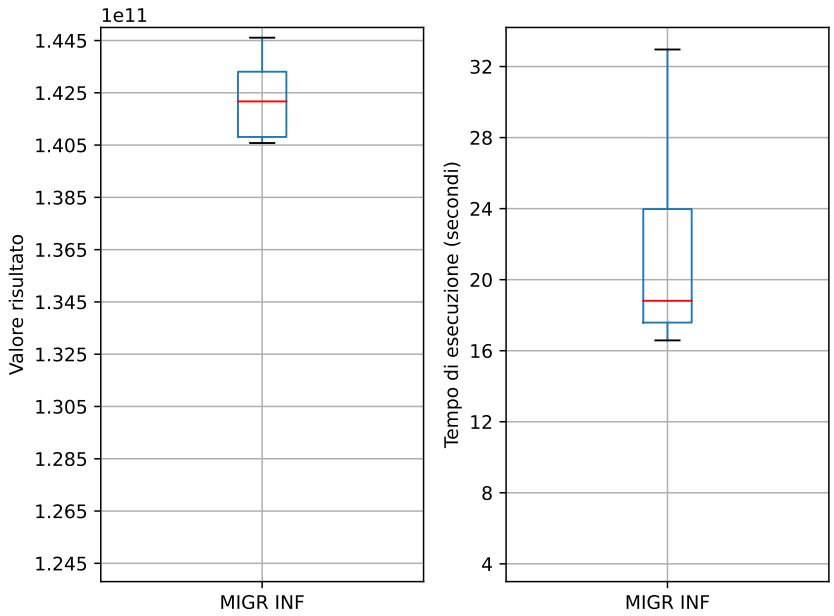

Figura 6: Distribuzione dei risultati e dei tempi di calcolo ottenuti usando la configurazione `MIGRAZIONI_INF`.

Una panoramica completa sui risultati ottenuti utilizzando le configurazioni della prima euristica, è mostrata nelle figure 4 e 5, che illustrano la distribuzione dei valori obbiettivo risultanti e dei tempi di calcolo impiegati. In tali figure non sono presenti i dati riguardanti la configurazione con migrazioni a costo infinito (`E1_MIGRAZIONI_INF`), dato che avendo una scala diversa rispetto agli altri valori, si sarebbero ottenuti grafici più compressi, e di conseguenza le sue distribuzioni sono mostrate nella figura 6.

Osservando le distribuzioni dei valori risultanti, sembra chiaro come nei casi `LAD` e `LSD` suddividere l'istanza in 24 parti porti a risultati migliori rispetto alla suddivisione in 12 o 24, mentre nella configurazione `UGUALE` conviene dividere in 12. In generale, osservando il confronto tra i due casi estremi (`MIGRAZIONI_0` e `MIGRAZIONI_INF`), si nota come l'algoritmo tragga più beneficio nell'effettuare molte suddivisioni rispetto a non farne nessuna: questo evidenzia come la latenza e la gestione dell'energia abbiano un impatto molto maggiore rispetto al costo da pagare per le migrazioni.

5.4.3 Risultati euristica 2

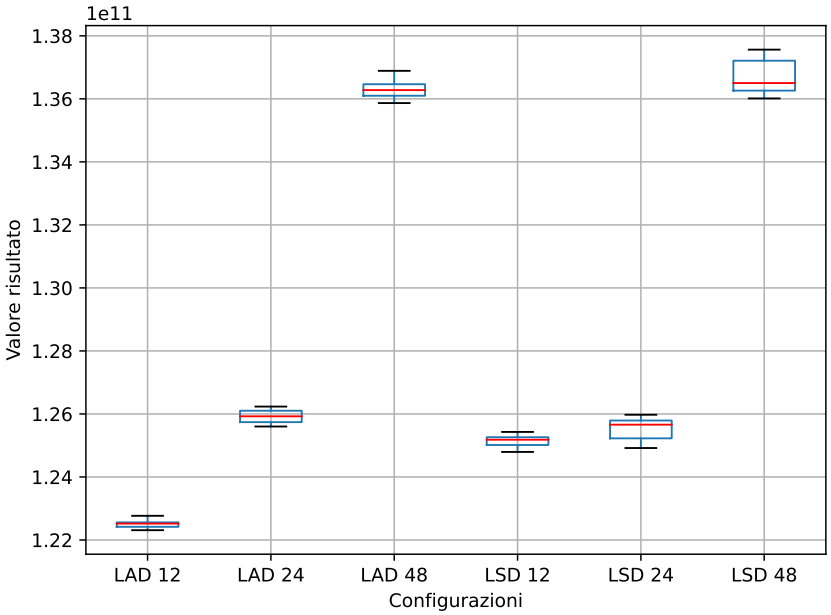

Figura 7: Distribuzione dei risultati ottenuti usando l'euristica 2.

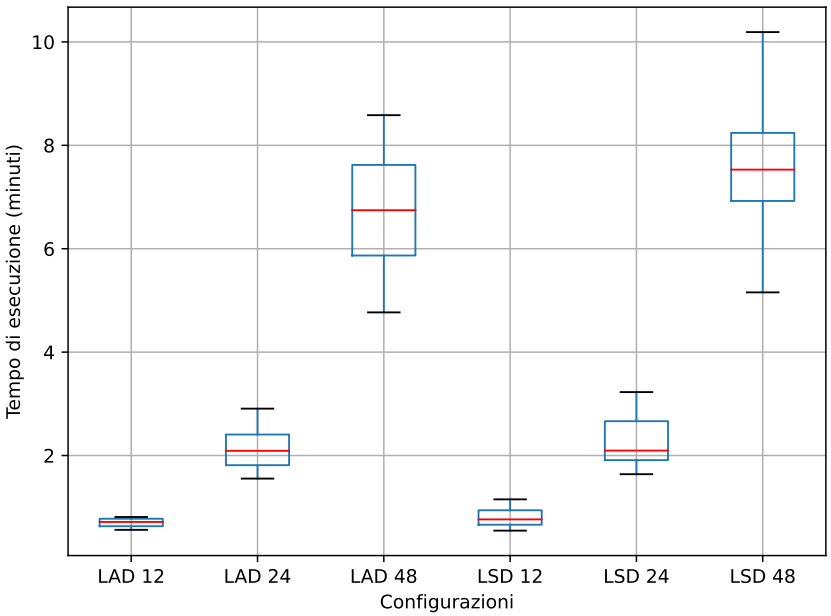

Figura 8: Distribuzione dei tempi di calcolo ottenuti usando l'euristica 2.

Come nell'analisi precedente, vengono rappresentati i grafici rappresentanti le distribuzioni dei valori ottenuti (figura 7) e dei tempi di esecuzione risultanti (figura 8).

Nel grafico riguardante i valori dei risultati, viene chiaramente mostrato come effettuare la semplificazione in istanze piccole sia più vantaggioso rispetto a quelle più grandi. I tempi di esecuzione sottolineano invece la complessità combinatoria del modello di assegnamento dinamico, utilizzato per risolvere le sottoistanze: al crescere nel numero di fascie temporali, il tempo di esecuzione aumenta esponenzialmente. Confrontando le due metriche, è chiaro come il metodo utilizzato per frammentare l'istanza sia molto meno determinante rispetto alla dimensione dell'istanza semplificata.

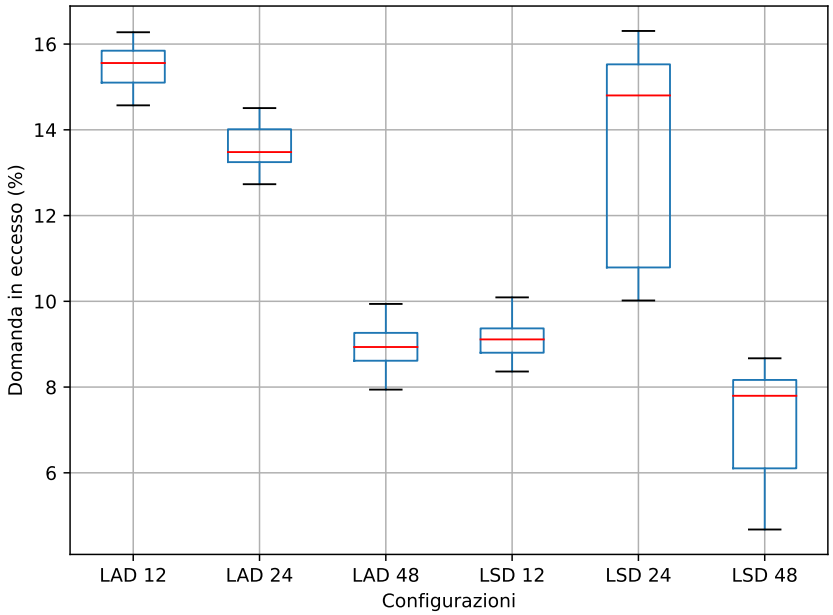

Figura 9: Distribuzione della quantità percentuale di domanda in eccesso assegnata alle facility in ogni configurazione.

Per ottenere un'analisi completa di questa euristica, è necessario calcolare e valutare la quantità di domanda che eccede dalla capienza delle facility. Tale valore è dato dalla somma della domanda in eccesso in ogni facility e in ogni istante temporale:

$$\sum_{t \in T} \sum_{k \in K} \max\{0, v_k^t - C_k\}. \tag{5.4.1}$$

Nella figura 9 viene mostrata la distribuzione percentuale di tale quantità.

Capitolo 6

Conclusioni

In conclusione, questo lavoro di tesi ha permesso di mostrare come sia possibile estendere in modo compatto i modelli di ottimizzazione presenti in letteratura, al fine di gestire esplicitamente la quantità di energia utilizzata nei siti MEC e considerando uno scenario di produzione e accumulo energetico. La fase sperimentale ha inoltre mostrato come le due euristiche proposte rappresentino valide implementazioni per il modello di orchestrazione considerato, in quanto permettono di abbassare notevolmente il tempo di calcolo e peggiorare le soluzioni solo di pochi punti percentuali. Negli esperimenti condotti, è stato anche sottolineato come la loro configurazione, vale a dire al modo in cui viene suddivisa l'istanza nella prima euristica e in che modo vengono aggregate le fascie temporali nella seconda, impatti notevolmente sui risultati ed i tempi di calcolo ottenuti, e mostra come nessuno dei due algoritmi abbia un impatto predominante sull'altro.

Appendici

Appendice A

Modelli di ottimizzazione aggiuntivi

A.1 Frammentazione di array in blocchi della stessa somma

Dato un array, l'obbiettivo di questo modello è quello di frammentarlo in un numero predefinito di blocchi aventi approssimativamente la stessa somma.

Dati. Sia I l'insieme dei blocchi e J quello degli indici dell'array. Il numero di blocchi che si intendono ottenere è indicato con n, e il valore del j-esimo elemento viene rappresentato con v_j ($j \in J$). Con m si denota invece la somma che avrebbe ciascun blocco nel caso in cui sia possibile attuare una frammentazione perfetta, vale a dire $m = \frac{1}{|I|} \sum_{j \in J} v_j$.

Variabili. Siano x_j^i variabili binarie che assumono valore 1 se il j-esimo elemento dell'array fa parte dell'i-esimo blocco, 0 altrimenti. Le variabili s_j^i sono anch'esse binarie e assumono valore 1 se il primo elemento dl'i-esimo blocco è il j-esimo elemento dell'array, altrimenti 0. Le variabili d^i definiscono la distanza tra la somma del blocco i e il valore ottimo m. Considerando $i \in I$ e $j \in J$.

Modello. Di seguito la rappresentazione matematica.

$$\min \quad \sum_{i \in I} d^i \tag{A.1.1}$$

$$\text{s.t.} \quad \sum_{i \in I} x_i^j = 1 \qquad \forall j \in J \tag{A.1.2}$$

$$\sum_{j \in J} x_i^j \geq 1 \qquad \forall i \in I \setminus \{0\} \tag{A.1.3}$$

$$x_0^0 = 1 \tag{A.1.4}$$

$$s_0^0 = 1 \tag{A.1.5}$$

$$s_0^i = 0 \qquad \forall i \in I \setminus \{0\} \tag{A.1.6}$$

$$s_j^i \geq x_j^i - x_{j-1}^i \qquad \forall i \in I, \forall j \in J \setminus \{0\} \tag{A.1.7}$$

$$\sum_{j \in J} s_j^i = 1 \qquad \forall i \in I \tag{A.1.8}$$

$$d^i \geq \sum_{j \in J} x_j^i v_j - m \qquad \forall i \in I \tag{A.1.9}$$

$$d^i \geq -\left(\sum_{j \in J} x_j^i v_j - m \right) \qquad \forall i \in I \tag{A.1.10}$$

$$x_j^i \in \{0, 1\} \qquad \forall i \in I \tag{A.1.11}$$

$$s_j^i \in \{0, 1\} \qquad \forall i \in I \tag{A.1.12}$$

Obbiettivo. L'obbiettivo (equazione A.1.1) è quello di ottenere somme dei blocchi il più possibile vicine tra loro, andando a minimizzare la somma delle distanze che li separa dal valore ottimo m.

Vincoli. I vincoli A.1.2 impongono ad ogni elemento dell'array di far parte di un singolo blocco, mentre i A.1.3 fanno in modo che ogni blocco contenga almeno un elemento. I vincoli A.1.4, A.1.5, A.1.6, A.1.7, A.1.8 impongono ad ogni blocco di avere un singolo indice di partenza e fanno in modo che i suoi elementi siano consecutivi. Da notare come la presenza dei vincoli A.1.3 sia dovuta al fatto che i A.1.7 non garantiscono che ogni blocco abbia un singolo indice iniziale, ma che se esiste allora è unico. I vincoli A.1.9 e A.1.10 sono l'implenentazione in programmazione lineare del concetto di valore assoluto, e definiscono il valore delle variabili d:

$$d^i = \left| \sum_{j \in J} x_j^i v_j - m \right| \qquad \forall i \in I \tag{A.1.13}$$

A.1.1 Algoritmo euristico

Quando la dimensione dell'array o quelle del numero di blocchi da ricavare aumenta, il tempo necessario a determinare la soluzione ottima cresce velocemente, e di conseguenza per semplificare la complessità del modello viene introdotto un

algoritmo euristico. Il problema viene semplificato scomponendo l'array in m parti composte da n/m elementi: ciascuna viene risolta indipendentemente, imponendo di generare un numero di blocchi che corrisponde all'incisione percentuale della somma dei suoi elementi rispetto alla somma dell'intero array. Nelle ottimizzazioni effettuate, la funzione obbiettivo deve minimizzare la distanza della somma dei blocchi rispetto allo stesso valore, quindi in ciascuna, m deve essere calcolato consideranto l'array completo e non la sottoparte da risolvere. Questo algoritmo non determina la soluzione ottima, ma fornisce una buona approssimazione che peggiora con l'aumentare delle scomposizioni attuate.

A.2 Frammentazione di array in blocchi omogenei

Dato un array, lo scopo di questo modello di ottimizzazione è quello di frammentarlo in un numero predefinito di blocchi in modo da minimizzare la differenza dei suoi elementi all'interno di ciascun blocco. Il problema descritto è una versione del p-median problem on a line, risolvibile in tempo polinomiale tramite programmazione dinamica [7].

Dati. Sia I l'insieme degli elementi dell'array, d_{ij} una misura della differenza tra gli elementi $i \in I$ e $j \in I$, e p un parametro che definisce quanti blocchi si intende generare.

Variabili. Le variabili y_j valgono 1 se l'elemento $j \in I$ è scelto come rappresentante, 0 altrimenti. Le variabili x_{ij} valgono 1 se l'elemento $i \in I$ è inserito in un blocco il cui rappresentante è $j \in I$, 0 altrimenti.

Modello. Di seguito la rappresentazione matematica del modello.

$$\min \quad \sum_{i \in I} \sum_{j \in I} d_{ij} x_{ij} \tag{A.2.1}$$

$$\text{s.t.} \quad \sum_{j \in I} x_{ij} = 1 \qquad \forall i \in I \tag{A.2.2}$$

$$x_{ij} \leq y_j \qquad \forall i \in I, \forall j \in I \tag{A.2.3}$$

$$\sum_{j \in I} y_j = p \tag{A.2.4}$$

$$x_{ij} \leq x_{kj} \qquad \forall i \in I, \forall j \in I, \forall k \in I : j > i \wedge k > i \wedge k < j \tag{A.2.5}$$

$$x_{ij} \leq x_{kj} \qquad \forall i \in I, \forall j \in I, \forall k \in I : j < i \wedge k < i \wedge k > j \qquad \text{(A.2.6)}$$
$$x_{ij} \in \{0,1\} \qquad \forall i \in I, \forall j \in I \qquad \text{(A.2.7)}$$
$$y_j \in \{0,1\} \qquad \forall j \in I \qquad \text{(A.2.8)}$$

Obbiettivo. La funzione obiettivo (A.2.1) minimizza la somma delle differenze tra un elemento e il rappresentante del blocco in cui è inserito.

Vincoli. I vincoli (A.2.2) assicurano che ogni elemento sia inserito in un blocco. I vincoli (A.2.3) assicurano che un elemento possa essere inserito in un blocco il cui rappresentante è $j \in I$ solo se j è identificato come un rappresentante. I vincoli (A.2.5) assicurano che se ci sono tre elementi $i, k, j \in I \times I \times I$ che compaiono uno dopo l'altro in ordine, e i è assegnato a j, allora anche k è assegnato a j. Simmetricamente, i vincoli (A.2.6) assicurano che se ci sono tre elementi $j, k, i \in I \times I \times I$ che compaiono uno dopo l'altro in ordine, e i è assegnato a j, allora anche k è assegnato a j.

Bibliografia

[1] Hédé, P., Joubert, J., Thornton, C., Naughton, B., Roldan Ramos, J., Chan, C., Young, V., Jin Tan, S., Lynch, D., Sprecher, N., Musiol, T., Manzanares, C., Rauschenbach, U., Abeta, S., Chen, L., Shimizu, K., Neal, A., Cosimini, P., Pollard, A., Klas, G., Patel, M., Hu, Y. Mobile-edge computing introductory technical white paper. technical report. mobile-edge computing (mec) industry initiative., 2014. `https://www.etsi.org/technologies/multi-access-edge-computing/mec-poc`.

[2] Tuyen X. Tran, Abolfazl Hajisami, Parul Pandey, and Dario Pompili. Collaborative mobile edge computing in 5g networks: New paradigms, scenarios, and challenges. *IEEE Communications Magazine*, 55(4):54–61, 2017.

[3] Adnan Aijaz, Mischa Dohler, A. Aghvami, Vasilis Friderikos, and Magnus Frodigh. Realizing the tactile internet: Haptic communications over next generation 5g cellular networks. *IEEE Wireless Communications*, PP, 12 2015.

[4] Alberto Ceselli, Marco Fiore, Marco Premoli, and Stefano Secci. Optimized assignment patterns in mobile edge cloud networks. *Computers & Operations Research*, 106:246–259, 2019.

[5] Alberto Ceselli, Marco Fiore, Angelo Furno, Marco Premoli, Stefano Secci, and Razvan Stanica. Prescriptive analytics for mec orchestration. In *2018 IFIP Networking Conference (IFIP Networking) and Workshops*, pages 1–9, 2018.

[6] Barlacchi. A multi-source dataset of urban life in the city of milan and the province of trentino. *Scientific Data*, 2(1):150055, Oct 2015.

[7] R. Hassin and A. Tamir. Improved complexity bounds for location problems on the real line. *Operations Research Letters*, 10(7):395–402, 1991.