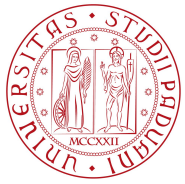


UNIVERSITÀ DEGLI STUDI DI PADOVA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

TESI DI LAUREA MAGISTRALE IN

INGEGNERIA DELLE TELECOMUNICAZIONI

MONITORAGGIO E CONTROLLO LOCALE DEI CONSUMI ELETTRICI IN UNA SMART MICROGRID

RELATORE: *Prof. Michele Rossi*

LAUREANDO: *Alberto Rosponi*

Padova, 11 luglio 2011

Indice

Sommario

La rete elettrica tradizionale è destinata a subire, nell'arco di pochi anni, radicali modifiche, che la trasformeranno in un sistema distribuito "intelligente" di produzione e gestione delle risorse energetiche. Tale sistema è noto con il nome di "*Smart Grid*". L'efficienza della nuova rete dovrà essere garantita da un'adeguata struttura gerarchica di sensori e algoritmi di controllo, al fine di ottimizzare l'utilizzo delle risorse e ridurre conseguentemente i costi e gli sprechi energetici. Nella Tesi viene proposto un algoritmo di controllo a livello locale, il cui obiettivo è quello di pianificare le attività più onerose dal punto di vista energetico, nell'intervallo temporale stabilito, in modo da minimizzare la spesa complessiva del consumatore e massimizzare al tempo stesso lo sfruttamento delle risorse rinnovabili quando disponibili. Viene presentato inoltre un prototipo di "*Radio Smart Meter*" realizzato in laboratorio, in grado di compiere misurazioni di tensione e corrente in tempo reale e di trasmetterle via radio al controllore locale. Esse permettono il calcolo della potenza attiva assorbita o generata (mediante pannelli solari) dall'edificio e quindi l'esecuzione dell'algoritmo di *scheduling* delle attività.

Capitolo 1

Introduzione

1.1 Concetto di *Smart Grid*

Negli ultimi anni si è diffuso un interesse sempre maggiore nel campo delle modalità di produzione e consumo di energia elettrica. I grandi impianti di produzione e distribuzione hanno dovuto adeguarsi alla costante crescita della domanda energetica, imposta dall'esponenziale sviluppo tecnologico e dall'aumento della popolazione. Al tempo stesso, le previsioni sull'esaurimento delle risorse non rinnovabili, che rappresentano oggi la fonte di oltre l'80% dell'energia prodotta a livello mondiale, ci spingono sempre più nella direzione delle risorse rinnovabili. In un futuro non troppo lontano, l'utilizzo di queste ultime diventerà determinante, e di conseguenza anche le tecnologie e le strutture adibite alla loro gestione e distribuzione dovranno garantire la maggior efficienza possibile.

Lo stesso concetto di rete elettrica è destinato a cambiare in pochi anni, trasformando gradualmente l'attuale sistema centralizzato (dalla centrale elettrica al consumatore) in un sistema distribuito: l'utente sarà in grado di produrre non solo l'energia necessaria al proprio fabbisogno, ma anche un *surplus* energetico, che potrà essere venduto alle società distributrici (come avviene attualmente nel caso di consumatori che aderiscono all'installazione degli appositi impianti) o addirittura sfruttato dagli utenti geograficamente vicini.

La rete elettrica tradizionale è nota in letteratura anche con il nome di "*Grid*", ovvero una Griglia, i cui nodi rappresentano edifici che utilizzano energia o centrali che la producono. Nel nuovo scenario descritto sopra, più edifici adiacenti possono essere raggruppati a loro volta in reti più piccole a Media e Bassa Tensione ("*Microgrid*"), in grado di scambiare energia elettrica

tra loro o, eventualmente, con i grandi impianti di produzione. Si tratta di una struttura a livelli gerarchici paragonabile a Internet, in cui viaggiano flussi energetici in modo bi-direzionale tra le Microgrid e la "*Backbone*" della rete principale ad Alta Tensione [1]. Rendere efficiente questo sistema significa sostanzialmente monitorarne i flussi ai vari livelli e regolarli, per ridurre quanto più possibile gli sprechi e contemporaneamente i costi sostenuti dai vari consumatori, soddisfandone le esigenze; in altre parole si tratta di rendere "intelligente" la rete, realizzando così una "*Smart Grid*" [2], [3].

Nella pratica tale obiettivo può essere raggiunto mediante un sistema di sensori e controllori ai vari livelli gerarchici. A livello locale, si dovrà fare in modo che ogni nodo, appartenente a una certa sotto-rete, concentri la maggior parte delle attività che richiedono energia (soprattutto quelle più onerose) nei periodi della giornata in cui è più elevata la quantità di energia rinnovabile prodotta all'interno della sotto-rete. Inoltre, è ragionevole che, nel caso si debba ricorrere all'energia fornita dalle centrali, si cerchi di utilizzarla nelle fasce orarie in cui essa costi meno.

1.2 Stato dell'arte

In [4], [5] si definisce uno schema gerarchico per il controllo di una generica Smart Microgrid: poiché numerosi edifici si trovano a dover condividere le medesime risorse e cercano ognuno di sfruttarle per minimizzare i propri costi, è necessaria la presenza di un controllore globale che, in base alle varie richieste, risolva un problema di ottimizzazione e quindi assegni le risorse a ciascun nodo. Il nodo, a sua volta, dovrà individuare una politica di gestione ottimale (mediante un controllore locale) delle risorse disponibili per cercare di soddisfare la richiesta del relativo utente. Il problema principale è dovuto al fatto che l'energia rinnovabile e l'energia richiesta non sono quantità definibili a priori. Il modello proposto in [4] non considera la possibilità di usufruire delle fonti rinnovabili, ma assume che i nodi della Smart Grid siano dotati di "*micro CHP*", ovvero dispositivi in grado di produrre calore ed elettricità a partire da combustibili di vario tipo (nel caso specifico, da gas). Si assume quindi che il controllore globale sia in grado di agire su tali dispositivi, attivandoli o disattivandoli nei vari nodi, per gestire i flussi energetici dell'intera Microgrid.

Il controllore locale ha invece il compito di distribuire ai vari apparecchi dell'edificio la potenza erogata dal micro CHP nel periodo in cui esso viene attivato. Per eseguire questa operazione, il controllore deve conoscere il profilo energetico quotidiano dell'utenza, ovvero

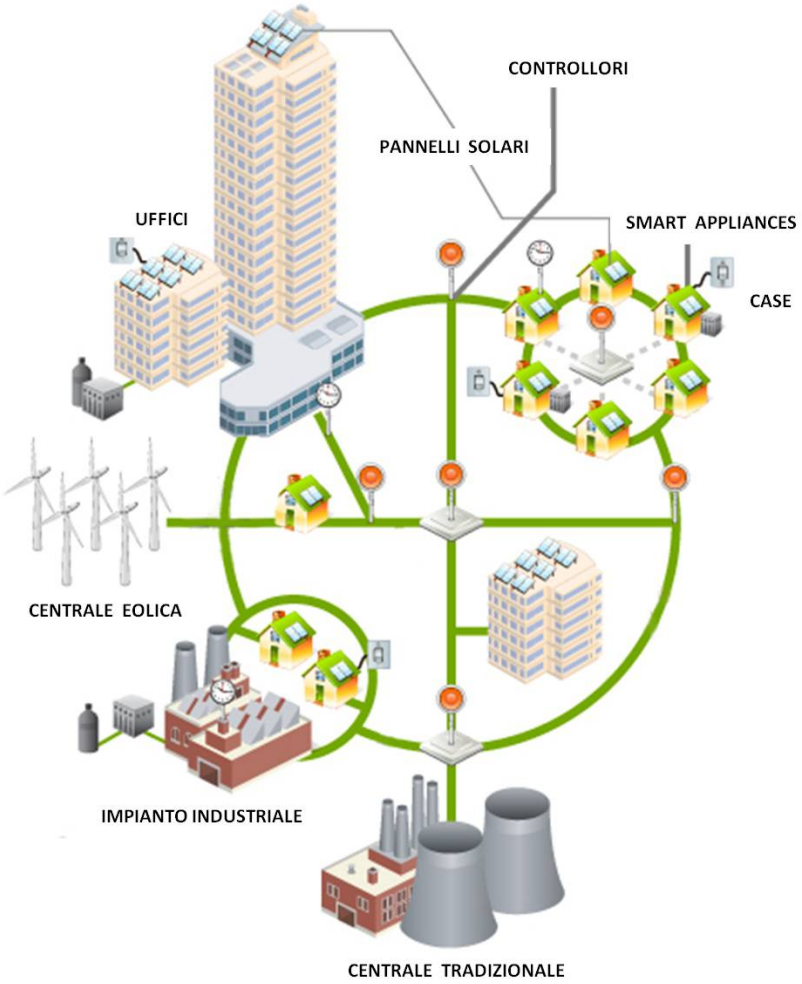


Figura 1.1: Esempio di Smart Microgrid.

quali attività dovranno essere svolte e in quali intervalli di tempo nell'arco della giornata. Come già accennato, si tratta di una grandezza dinamica e non definibile a priori in modo esatto, tuttavia può essere in qualche modo prevista sfruttando ragionevolmente l'abitualità nel comportamento degli utenti: la storia dei profili viene memorizzata dal controllore per essere fornita in input a una rete neurale, assieme ad altri parametri che influenzano la richiesta energetica, come ad esempio le condizioni metereologiche e le caratteristiche dell'edificio. L'output della rete neurale è il profilo previsto per il giorno successivo. In caso di conflitto tra la previsione e la reale richiesta, viene naturalmente salvaguardato il comfort dell'utente, correggendo in tempo reale lo scheduling del controllore.

La stessa previsione deve essere tradotta in una richiesta di energia complessiva da tra-

smettere, insieme alle richieste previste di tutti gli altri nodi, al controllore globale. Il tempo è suddiviso in slot, i problemi di ottimizzazione di entrambi i controllori sono formulati come ILP (Programmazione Lineare Intera) e risolti all'inizio di ogni slot.

Una limitazione del modello riportato è che il profilo energetico previsto, una volta stimato, non possa "evolvere" durante il periodo in cui viene utilizzato. Si tratta di un profilo deterministico, che non può pertanto adattarsi a improvvisi cambiamenti non previsti dalla rete neurale. Esso inoltre viene calcolato inglobando indistintamente tutte le attività che richiedono energia, incluse quelle difficilmente controllabili, ovvero quelle che il consumatore deve poter svolgere in qualunque momento, a seconda delle proprie necessità.

Il modello proposto nella Tesi cerca di migliorare questi aspetti, rappresentando il profilo dell'utente con una Catena di Markov (*MC, Markov Chain*) per mantenerne la natura aleatoria, e inserendo nello scheduling solo le attività effettivamente controllabili; con riferimento all'ambiente domestico, esse coincidono con i grandi elettrodomestici, che oltretutto influiscono maggiormente sui consumi quotidiani. Si distingue pertanto il profilo (o richiesta energetica) non controllabile, che sarà quello rappresentato mediante la MC, dal profilo inseribile all'interno dello scheduling. Sebbene l'obiettivo principale sia gestire in modo ottimo quest'ultimo, il profilo aleatorio deve essere inserito come vincolo all'interno del problema, poiché in ogni istante deve essere rispettato il limite di potenza assorbita dalla rete elettrica (che nella maggior parte dei casi, ad uso domestico, è pari a 3.3 kW).

Gli autori di [6] si concentrano, come nel nostro caso, esclusivamente sul problema di controllo locale dei consumi, proponendo però ancora una volta un modello deterministico per lo scheduling. Si introduce inoltre un "costo di attesa", che cresce esponenzialmente all'aumentare del ritardo, in numero di slot, con cui il controllore inserisce le attività nel piano. A nostro avviso, è più utile specificare delle scadenze precise per ciascuna attività e assegnare un costo molto elevato solo al loro superamento (come si vedrà nel paragrafo 2.5.3). In [6] il costo dell'energia non è un parametro arbitrario e statico, ma il costo monetario reale all'interno della Smart Grid, e si definisce un semplice ma efficace modello di previsione per tenere conto delle sue variazioni nel tempo. Tale modello sarà ripresentato nel paragrafo 2.5.4.

1.3 Schema di controllo proposto

Nella trattazione si farà riferimento a una Smart Microgrid in cui alcuni nodi hanno a disposizione una struttura per la produzione di energia rinnovabile (ad esempio pannelli solari o turbine eoliche). Si suppone che la quantità di energia prodotta da tali nodi, ma non utilizzata immediatamente all'interno della Smart Microgrid, non possa essere immagazzinata e utilizzata successivamente. Questo eccesso energetico pertanto verrà considerato come "sprecato".

La quantità di energia rinnovabile è una grandezza aleatoria, dunque verrà definita anch'essa ricorrendo alla Teoria delle MC, affinché il suo andamento possa essere in qualche modo previsto.

Ogni nodo è in grado di rilevare, in tempo reale, la potenza elettrica che consuma e quella che può sfruttare dalle fonti di energia rinnovabile, prodotta dal nodo stesso o dai nodi vicini. Ciò può essere ottenuto utilizzando un cosiddetto "*Smart Meter*", un contatore digitale che, a differenza di quelli tradizionali, consente l'acquisizione di misure di potenza elettrica in tempo reale e la loro trasmissione per mezzo di una rete (cablata o wireless) [7]. Per questo motivo, una parte del nostro lavoro è stata dedicata anche alla realizzazione di uno Smart Meter e del suo interfacciamento con l'algoritmo di scheduling. Possiamo supporre che, alla base del dispositivo, sia presente uno strumento software con funzionalità di "*signature detection*" o un meccanismo di "switching" periodico, per fare in modo che un unico Smart Meter sia sufficiente a raccogliere informazioni provenienti sia dalla rete elettrica interna (consumo locale) sia dalla rete esterna (energia rinnovabile disponibile).

Nella Tesi ci concentreremo esclusivamente sulla parte di controllo locale, trascurando la gestione globale delle risorse sull'intera Griglia. I dati rilevati dal Meter sono utilizzati in primo luogo per aggiornare le MC associate all'energia rinnovabile e al profilo non controllabile; inoltre essi costituiscono alcuni dei parametri di ingresso dell'algoritmo che si occupa di pianificare il funzionamento dei grandi elettrodomestici in modo ottimale, a partire dall'istante desiderato e all'interno di una finestra della durata massima di 24 ore. Anche nel nostro caso il tempo è suddiviso in slot, e l'algoritmo viene eseguito all'inizio di ciascun intervallo, per fare in modo che lo scheduling possa essere corretto "al volo" qualora gli andamenti previsti della richiesta non controllabile e dell'energia rinnovabile differiscano da quelli correntemente misurati. Per risolvere il problema di ottimizzazione si ricorre alla tecnica di Programmazione Dinamica (*DP, Dynamic Programming*).

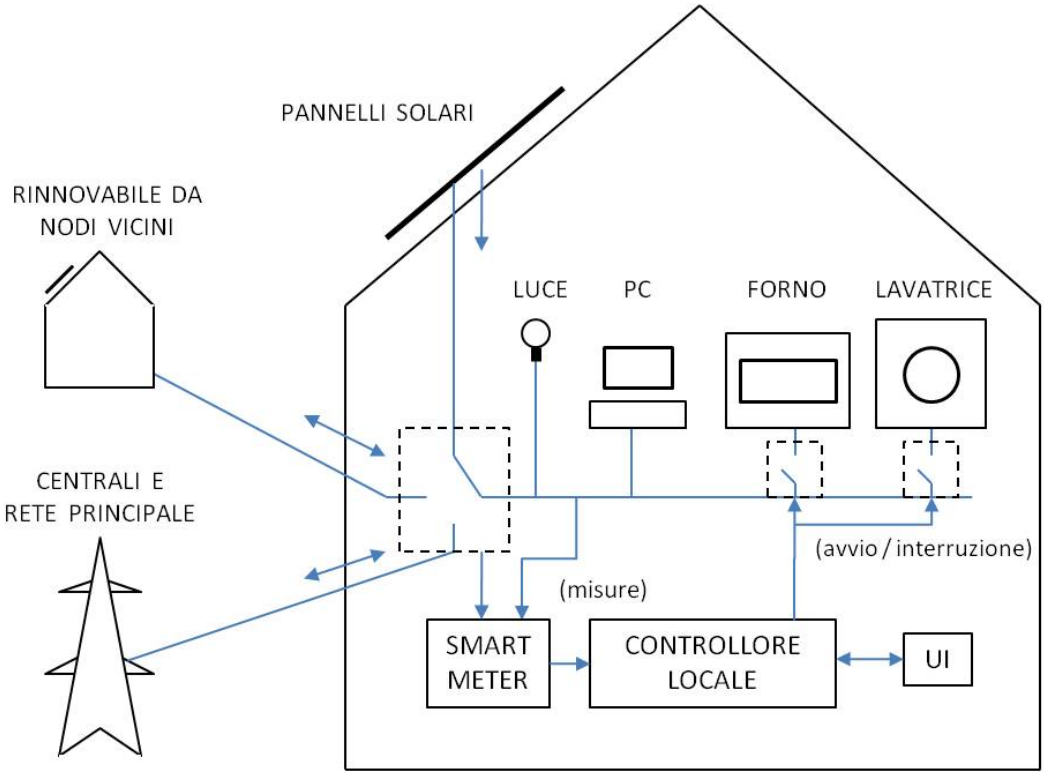


Figura 1.2: Schema di controllo locale.

Si può supporre che lo scheduler agisca direttamente sugli apparecchi tramite una piccola rete HAN (*Home Area Network*), vista soprattutto la recente introduzione sul mercato di elettrodomestici predisposti a tale connessione, allo scopo di poter gestire i consumi in maniera del tutto autonoma. L'utente può impostare le attività da pianificare e l'orario di inizio del piano tramite un'apposita interfaccia (*UI*, *User Interface*). La Figura 1.2 illustra lo scenario appena descritto.

Capitolo 2

Modelli per l'ottimizzazione

L'algoritmo di scheduling si basa sulla tecnica di Programmazione Dinamica (DP), che illustreremo a livello generale nel paragrafo 2.4 e successivamente nell'implementazione specifica del nostro caso.

L'obiettivo primario dell'algoritmo è quello di pianificare l'esecuzione di determinate attività (che indicheremo equivalentemente con il termine "*appliance*") a partire da un certo istante, ciascuna entro le scadenze imposte dall'utente, minimizzandone i costi. Il piano ottimale prevede l'esecuzione delle attività preferibilmente nelle fasce orarie del giorno in cui le risorse rinnovabili (prodotte all'interno della Smart Grid) sono disponibili in quantità sufficiente, garantendo un costo nullo; se ciò non è (in parte o del tutto) possibile, esso deve concentrare l'esecuzione nelle fasce orarie in cui l'energia non rinnovabile (erogata dalla società distributrice) ha un costo minore. In Italia ciò si traduce nel preferire l'intervallo di tempo tra le ore 19 e le ore 8 del giorno successivo, se l'utente ha aderito al contratto con "tariffa bioraria". Solo quando entrambe le condizioni non sono realizzabili, l'algoritmo deve allocare l'energia spesa dalle appliance nelle altre fasce orarie per terminare comunque tutte le attività entro le rispettive scadenze.

Dopo un breve richiamo alla teoria delle Catene di Markov [8], a partire dal paragrafo 2.2 analizzeremo prima di tutto gli elementi essenziali utilizzati dall'algoritmo.

2.1 Catene di Markov

2.1.1 Processi di Markov

Un processo aleatorio $s(t)$ si definisce processo di Markov se, noto il valore assunto in un certo istante t_0, la sua descrizione statistica completa negli istanti successivi dipende esclusivamente da tale valore, mentre non dipende dai valori assunti negli istanti che precedono t_0. Supposto che $I \subseteq \mathbf{R}$ sia il dominio temporale di $s(t)$, se $t_0 \in I$ rappresenta l'istante presente, $s(t_0) = s_0$ è il valore assunto dal processo, o "evento" presente. L'evento futuro è costituito di conseguenza dai valori assunti negli istanti successivi a t_0, mentre l'evento passato dai valori assunti negli istanti precedenti. La proprietà inizialmente descritta può essere allora riformulata, ricorrendo alla probabilità condizionata, nella cosiddetta "*condizione di markovianità*":

$$P(futuro \mid s_0,\, passato) = P(futuro \mid s_0).$$

I valori assunti da $s(t)$ sono in genere chiamati "stati". Un processo di Markov ad ampiezze discrete (in cui l'insieme degli stati è discreto) viene detto "Catena di Markov"(*Markov Chain*, *MC*). Per il nostro modello ci interessano in particolare le Catene di Markov a stati finiti e a tempo discreto.

2.1.2 Catene di Markov a stati finiti e a tempo discreto

Una MC a stati finiti e a tempo discreto è caratterizzata da uno spazio degli stati S di carinalità finita (che indichiamo con $J+1$) e un dominio temporale I discreto. Indichiamo con n l'istante presente, in cui il processo si trova nello stato s_n, con m un qualche istante futuro e con $n_1 > n_2 > ... > n_K$ alcuni istanti passati. La condizione di markovianità diventa:

$$P(s_m = j | s_n = i, s_{n_1} = i_1, ..., s_{n_K} = i_K) = P(s_m = j | s_n = i), \qquad j, i, i_1, ...i_K \in S.$$

Particolarmente importanti risultano le cosiddette "*probabilità di transizione*", definite come le probabilità condizionate

$$p_{ij}^{(n+k)} = P(s_{n+k} = j | s_n = i),$$

per ogni intero $k > 0$ e per ogni coppia (i, j) di stati appartenenti a S. Esse esprimono la probabilità che la MC si trovi nello stato futuro j all'istante $n+k$, dato che si trova nello stato i all'istante presente n. In letteratura, i k istanti successivi all'istante presente vengono detti

"*passi*", e molto spesso (come nel nostro caso) interessano probabilità di transizione con $k = 1$. È possibile raccogliere le varie probabilità di transizione in una "*matrice di transizione*" $J \times J$

$$\begin{pmatrix} p_{00}^{(n\ k)} & \cdots & p_{0J}^{(n\ k)} \\ \vdots & \ddots & \vdots \\ p_{J0}^{(n\ k)} & \cdots & p_{JJ}^{(n\ k)} \end{pmatrix},$$

che gode delle seguenti proprietà:

1) tutti i suoi elementi sono non negativi;

2) la somma degli elementi di ciascuna riga è pari a 1.

La prima proprietà risulta ovvia. La seconda proprietà deriva dal Teorema della Probabilità Totale:

$$\sum_{j=0}^{J} p_{ij} = \sum_{j=0}^{J} P(s_{n+k} = j | s_n = i) = P(s_{n+k} \in S | s_n = i) = 1.$$

Una MC si dice "*omogenea*" se tutte le sue matrici di transizione ad un passo ($k = 1$) sono indipendenti dall'istante di partenza n. Le MC definite nel nostro modello non sono omogenee, pertanto tralasciamo il richiamo di alcuni teoremi e proprietà fondamentali derivanti dall'omogeneità.

Durante la trattazione indicheremo, per semplicità, le probabilità di transizione con

$$p_{ij}(n+1) = p_{ij}^{(n+1)},$$

specificando quindi l'istante di arrivo e sottointendendo invece l'istante di partenza n e il passo.

2.2 Richiesta energetica non controllabile ed energia rinnovabile

Definiamo "profilo non controllabile" o "richiesta non controllabile" l'energia che deve essere assorbita in ogni istante dalla rete, durante un certo periodo di tempo, per alimentare tutti i dispositivi che non possono essere inseriti nello scheduling, ovvero quelli che devono essere a disposizione immediata dell'utente qualora abbia la necessità di utilizzarli. Nel contesto domestico, esempi di appliance non controllabili sono l'illuminazione, il funzionamento di un PC, di una stampante o di un televisore.

Nel nostro modello il tempo viene suddiviso in slot da 30 minuti ciascuno. È ragionevole supporre che il profilo non controllabile tenda a ripetersi ciclicamente dopo 24 ore, ovvero

dopo 48 slot, nel corso della settimana, fatta eccezione per il week-end, in cui i comportamenti dell'utenza differiscono con alta probabilità da quelli abituali. Per semplicità si considera la potenza *media* richiesta in ciascuno slot, in modo tale da avere un valore costante nel generico slot k, che indichiamo con r_k.

La potenza media $\{r\}$ è una grandezza aleatoria, che possiamo modellare come una MC a stati finiti e a tempo discreto. Indicato con R l'insieme degli stati assumibili da $\{r\}$, il passaggio dal k-esimo al $(k+1)$-esimo slot corrisponde all'istante di transizione della MC da un certo stato i a un altro stato j, entrambi appartenenti a R, che avviene con una probabilità

$$p'_{ij}(k+1) \;=\; P(r_{k+1} = j | r_k = i).$$

Le probabilità di transizione dipendono dal tempo (MC non omogenea) in quanto una parte degli stati possibili viene raggiunta più facilmente in determinate fasce orarie piuttosto che in altre. Ad esempio, le probabilità del tipo p'_{i0} sono sicuramente più elevate durante la notte, quando in genere si registra una scarsa attività da parte dei consumatori, mentre sono molto più basse nelle ore destinate ai pasti. Ne consegue che si deve disporre complessivamente di 48 matrici di transizione, una per ogni slot, più altre 48 considerando il comportamento diverso nei week-end.

Il limite superiore della potenza prelevabile dalla rete, stabilito in un tipico contratto per uso domestico, è pari a $r_{MAX} = 3.3$ kW. Nel modello abbiamo ridotto tale limite a 3 kW per evitare il rischio di raggiungimento della soglia critica, che comporta l'immediato distacco di corrente da parte delle centraline e dunque un rimarcabile disagio per l'utente. Attualmente il numero di stati possibili di $\{r\}$, ovvero la cardinalità di R, è pari a 10, pertanto i valori di potenza (media) sono quantizzati secondo multipli di 300 W (come vedremo meglio nel paragrafo 3.2).

La potenza derivata dall'energia rinnovabile disponibile in un qualsiasi nodo della Smart Microgrid, sia essa prodotta dal nodo stesso o prelevata dalla Griglia, è anch'essa una grandezza aleatoria modellabile come una MC a stati finiti e a tempo discreto. Con ragionamenti analoghi a quelli visti per la richiesta, in ogni slot facciamo riferimento al valore medio di tale grandezza, e nel generico slot k la indichiamo con e_{R_k}. Definiamo come R' lo spazio degli stati di $\{e_R\}$. La probabilità di transizione tra due generici stati i e j appartenenti a R', che indichiamo con

$$p''_{ij}(k+1) = P(e_{R_{k+1}} = j | e_{R_k} = i),$$

dipende dal tempo, poiché anche in questo caso una parte degli stati possibili viene raggiunta più facilmente in determinate fasce orarie piuttosto che in altre. Ad esempio, se supponiamo di utilizzare esclusivamente dei pannelli solari come fonte di produzione di $\{e_R\}$, le probabilità del tipo p''_{i0} sono praticamente unitarie durante la notte, in totale assenza di luce solare, mentre sono praticamente nulle durante la tarda mattinata o il pomeriggio. Anche per la completa caratterizzazione di $\{e_R\}$ si deve pertanto disporre di 48 matrici di transizione.

La cardinalità dell'insieme R' e la granularità della quantizzazione di $\{e_R\}$ sono le medesime fissate per $\{r\}$.

La costruzione e l'aggiornamento di ciascuna delle matrici di transizione dei due processi aleatori appena descritti possono essere compiuti nel modo seguente: se indichiamo con n_{TOT_i} il numero totale di transizioni effettuate dallo stato i all'inizio di uno stesso slot e con n_{ij} il numero di transizioni effettuate solo dallo stato i allo stato j nel medesimo istante,

$$p'_{ij} = \frac{n_{ij}}{n_{TOT_i}} = \frac{n_{ij}}{\sum_{h \in R} n_{ih}}. \tag{2.1}$$

Nell'espressione si è fatto riferimento alle transizioni di $\{r\}$, ma naturalmente analoghi calcoli valgono per $\{e_R\}$.

2.3 Appliance controllabili

Nel paragrafo precedente abbiamo definito il significato di "non controllabilità" nel nostro contesto, intesa come la proprietà delle appliance tale per cui esse difficilmente possono essere inserite in uno scheduling. Viceversa, le appliance "controllabili" sono le attività, sostanzialmente dei grandi elettrodomestici (forno, lavatrice, lavastoviglie, etc.), di cui l'algoritmo di ottimizzazione può gestire i consumi. Indichiamo il loro insieme con $\{a_n\}_{n=0,..N-1}$, dove N è il numero totale di appliance controllabili. Ogni attività deve essere definita da:

- *profilo energetico* $\underline{\mathbf{e}}_n$: un vettore contenente la potenza media richiesta dall'n-esima appliance in ogni slot della sua attività. La lunghezza del vettore corrisponde di conseguenza alla durata dell'appliance, espressa come numero di slot s_n;

- *dead-line* D_n: l'istante entro cui l'attività deve essere terminata;

- permesso di *interrompibilità* INT_n: l'attività di alcune appliance può essere interrotta più volte o può invece dover essere eseguita in un blocco unico. INT_n è quindi rappresentabile mediante una variabile binaria;

- *picco di potenza iniziale* P_n: il massimo livello di potenza assorbita dall'appliance nel primo slot del proprio funzionamento. Sperimentalmente si osserva che la maggior parte dei grandi elettrodomestici tende ad assorbire un picco molto elevato di potenza nei primi minuti della propria attività, per consentire il raggiungimento dell'elevata temperatura richiesta (riscaldamento dell'acqua nel caso della lavatrice, riscaldamento uniforme di un forno elettrico, etc.). Una volta raggiunta la temperatura, il livello di potenza cala rapidamente (nel caso specifico del forno in realtà si riscontrano periodicamente degli altri picchi, ogni qualvolta si abbassi la temperatura interna).

Abbiamo visto che i profili energetici delle varie appliance ne definiscono implicitamente le durate, intese come numero di slot necessario al loro completamento. Le durate possono essere raccolte all'interno di un unico vettore, che indichiamo con $\underline{\mathbf{s}}_{ATT} = [s_0, s_1, ..., s_{N-1}]$. L'algoritmo di scheduling, come vedremo nei paragrafi successivi, procede per iterazioni successive, effettuate all'inizio di ciascuno slot temporale. Ad ogni iterazione, l'algoritmo decide quali appliance mantenere in attività, allocando o meno l'energia specificata dagli elementi dei rispettivi profili. Il vettore delle durate verrà pertanto aggiornato ad ogni iterazione, decrementando di un'unità l'elemento associato all'appliance inserita nell'iterazione precedente, finché tutti i suoi elementi saranno nulli (se il piano andrà a buon fine) o si verificherà un errore. Nel primo caso tutte le appliance saranno state portate a termine nel tempo stabilito.

Analogamente si possono raccogliere le dead-line nel vettore $\underline{\mathbf{D}} = [D_0, D_1, ..., D_{N-1}]$, i permessi di interrompibilità nel vettore $\underline{\mathbf{I}}_{ATT} = [INT_0, INT_1, ..., INT_{N-1}]$ e i picchi di potenza assorbita nel vettore $\underline{\mathbf{P}}_{ATT} = [P_0, P_1, ..., P_{N-1}]$.

2.4 Programmazione dinamica

La tecnica di Programmazione Dinamica (DP) è spesso adottata negli scenari in cui occorre intraprendere delle decisioni in un certo numero di istanti temporali (*stage*), con l'obiettivo di minimizzare un determinato costo finale, derivato dalle conseguenze di tali decisioni. Esse non possono essere valutate nella loro singolarità, separatamente dalle altre, poiché un'azione ottima

in un certo istante (che produce cioè il costo minimo per l'intervallo di tempo che termina con l'inizio del successivo), può determinare una serie di decisioni non ottime nel futuro, ovvero un costo finale complessivamente elevato. La Programmazione Dinamica tiene conto di questo aspetto, calcolando il costo della decisione di ogni stage come somma del costo presente (dallo stage corrente al successivo) e del costo atteso futuro, supponendo ottime le decisioni di tutti gli istanti seguenti.

La formulazione di base di un problema risolvibile mediante DP prevede due elementi essenziali: un sistema dinamico, a tempo continuo o discreto, e una funzione di costo [9]. Nel nostro caso, il sistema è a tempo discreto e la funzione di costo è additiva nel tempo [9].

Lo stato del sistema, che indichiamo con la variabile x, evolve nel generico $(k+1)$-esimo stage secondo una certa funzione, dipendente dallo stato x_k nello stage corrente (k), dalla decisione (o azione di controllo) applicata, che indichiamo con u_k, e dall'eventuale disturbo aleatorio w_k. Se il problema è cosiddetto "a Orizzonte Finito", il numero di stage in cui intraprendere le decisioni è finito: si tratta di una "finestra temporale" composta da W slot. Volendo formalizzare quanto appena descritto, si ha che

$$x_{k+1} = f_k(x_k, u_k, w_k), \qquad k = 0, 1, ..., W-1.$$

Lo stato x_k è un elemento di un certo spazio S_k, l'azione u_k è un elemento di uno spazio C_k e il disturbo w_k appartiene a uno spazio D_k. Il controllo u_k dipende a sua volta dallo stato x_k, per cui i suoi valori si troveranno in un sottoinsieme $U(x_k) \subset C_k$, per ogni $x_k \in S_k$ e per ogni k.

La funzione di costo del k-esimo stage, che indichiamo con g_k, dipende anch'essa dallo stato corrente, dalla decisione compiuta e dal disturbo nello stesso stage. Essendo additiva, il costo finale del processo di ottimizzazione risulta:

$$g_W(x_W) + \sum_{k=0}^{W-1} g_k(x_k, u_k, w_k),$$

dove $g_W(x_W)$ è un costo aggiunto al termine del processo, in genere nullo. A causa dell'aleatorietà di w_k, tuttavia, non è possibile individuare con certezza il costo complessivo, e si deve calcolare il costo atteso, pesato rispetto alla distribuzione di probabilità di w_k:

$$E\left\{g_W(x_W) + \sum_{k=0}^{W-1} g_k(x_k, u_k, w_k)\right\}.$$

Possiamo considerare l'insieme di tutte le W decisioni (la cosiddetta "politica di controllo") prese durante il processo con:

$$\pi = \{u_0, u_1, ..., u_{W-1}\},$$

che dovrà naturalmente essere una politica ammissibile, formata solo da elementi appartenenti al sottoinsieme $U(x_k)$. Definiamo Π l'insieme contenente tutte le politiche ammissibili. Il costo finale risultante da una qualsiasi sequenza di controlli, a partire dallo stato iniziale x_0, si può indicare con:

$$J_\pi(x_0) = E\left\{g_W(x_W) + \sum_{k=0}^{W-1} g_k(x_k, u_k, w_k)\right\}. \tag{2.2}$$

L'obiettivo è quello di individuare la politica ottima π^* in modo tale da trovare:

$$J^*(x_0) = J_{\pi^*}(x_0) = \min_{\pi \in \Pi} J_\pi(x_0). \tag{2.3}$$

Analizziamo ora la tecnica vera e propria attraverso cui raggiungere l'obiettivo proposto. Il concetto fondamentale su cui si basa la Programmazione Dinamica è il *Principio di Ottimalità* [9], che illustriamo di seguito.

Principio di Ottimalità Sia $\pi^* = \{u_0^*, u_1^*, ..., u_{W-1}^*\}$ una politica ottima per la risoluzione del problema, e tale per cui si raggiunga un certo stato x_i all'istante i con probabilità non nulla. Si consideri il sotto-problema in cui lo stato iniziale è x_i all'istante i e si voglia minimizzare il costo atteso finale dallo stage i allo stage $W-1$:

$$E\left\{g_W(x_W) + \sum_{k=i}^{W-1} g_k(x_k, u_k, w_k)\right\}.$$

Allora la politica di controllo "troncata" $\{u_i^*, u_{i+1}^*, ..., u_{W-1}^*\}$ è ottima per tale sotto-problema.

È facile fornire una giustificazione intuitiva al Principio: se, per assurdo, la politica troncata non fosse ottima, si potrebbe ridurre ulteriormente il costo finale cambiando politica una volta raggiunto lo stato x_i all'istante i. Ma ciò significherebbe che la politica individuata inizialmente per l'intero problema non era effettivamente ottima. La contraddizione dimostra la tesi.

Il Principio di Ottimalità suggerisce un metodo di minimizzazione del costo procedendo a ritroso, ovvero: valutare l'azione di controllo ottima per l'ultimo stage, quindi passare al sotto-problema che inizia al penultimo stage ed espandere la politica aggiungendo l'azione ottima per il penultimo stage, e così via... Formalizziamo ora il metodo esposto, giungendo così alla definizione dell'algoritmo di Programmazione Dinamica [9] per la risoluzione del problema di base descritto in precedenza.

Algoritmo DP Per qualunque stato iniziale x_0, il costo ottimo $J^*(x_0)$ equivale al costo $J(x_0)$ ottenuto all'ultima iterazione del seguente algoritmo, che procede a ritroso dall'istante $W-1$ all'istante 0:

$$J(x_W) = g_W(x_W),$$

$$J(x_k) = \min_{u_k \in U_k(x_k)} E\left\{g_k(x_k, u_k, w_k) + J(x_{k+1})\right\}, \qquad k = 0, 1, ..., W-1, \tag{2.4}$$

dove l'aspettazione è calcolata rispetto alla distribuzione di probabilità di w_k. Inoltre, detta u_k^* l'azione che minimizza il termine a destra dell'equazione per qualsiasi x_k e k, la sequenza $\pi^* = \{u_0^*, u_1^*, ..., u_{W-1}^*\}$ è ottima.

Dim: Data una qualunque politica di controllo ammissibile $\pi = \{u_0, u_1, ..., u_{W-1}\}$, indichiamo con $\pi^k = \{u_k, u_{k+1}, ..., u_{W-1}\}$ la politica corrispondente troncata al sotto-problema con inizio in k. Ricordando le equazioni (2.2, 2.3), possiamo scrivere:

$$J^*(x_k) = \min_{\pi^k} E\left\{g_W(x_W) + \sum_{i=k}^{W-1} g_i(x_i, u_i, w_i)\right\}.$$

Per $k = W$, $J^*(x_W) = g_W(x_W)$. Dimostriamo per induzione che tutte le funzioni $J^*(x_k)$, al variare di k, sono equivalenti alle $J(x_k)$ generate dall'algoritmo DP, e quindi anche per $k = 0$, da cui giungiamo alla tesi.

Supponiamo che, per un certo k e per tutti i possibili stati x_{k+1}, valga $J^*(x_{k+1}) = J(x_{k+1})$.

Dato che $\pi^k = (u_k, \pi^{k+1})$, per tutti gli stati x_k si ha che

$$\begin{aligned}
J^*(x_k) &= \min_{(u_k, \pi^{k+1})} E\Big\{ g_k(x_k, u_k, w_k) + g_W(x_W) + \sum_{i=k+1}^{W-1} g_i(x_i, u_i, w_i) \Big\} \\
&= \min_{u_k} E\Big\{ g_k(x_k, u_k, w_k) + \min_{\pi^{k+1}} E\{ g_W(x_W) + \sum_{i=k+1}^{W-1} g_i(x_i, u_i, w_i) \} \Big\} \\
&= \min_{u_k} E\Big\{ g_k(x_k, u_k, w_k) + J^*(x_{k+1}) \Big\} \\
&= \min_{u_k} E\Big\{ g_k(x_k, u_k, w_k) + J(x_{k+1}) \Big\} \\
&= J(x_k).
\end{aligned}$$

Q.E.D.

2.5 Scheduler di appliance controllabili basato sull'algoritmo DP

Applichiamo ora quanto visto nel paragrafo precedente per costruire il nostro algoritmo di scheduling, in grado di pianificare le attività controllabili. Gli istanti in cui vengono compiute le azioni di controllo (stage) coincideranno con l'inizio dei vari slot.

2.5.1 Stato del sistema

A differenza del problema di base descritto in [9], nel nostro scenario lo stato in un generico stage $k+1$ è costituito da due parti: una deterministica, che evolve secondo una funzione $f_k(\cdot)$ e che possiamo indicare con x_{k+1}, e una aleatoria, formata a sua volta dai processi aleatori che abbiamo indicato con r_{k+1} ed $e_{R_{k+1}}$ (richiesta non controllabile ed energia rinnovabile), e che pertanto evolve secondo le rispettive probabilità di transizione.

Indicando la parte aleatoria con y_{k+1}, è necessario considerare la distribuzione di probabilità congiunta dei due processi per determinare l'evoluzione di y_k. Dato che $\{r\}$ ed $\{e_R\}$ sono statisticamente indipendenti, la distribuzione congiunta è determinata dal prodotto delle loro probabilità. Ciò significa che è possibile vedere y_k come una MC, che evolve in "macro-stati" derivati dall'intersezione degli stati di $\{r\}$ ed $\{e_R\}$. Indichiamo con $R = R' \times R''$ l'insieme dei macro-stati possibili di y. L'evoluzione da un generico macro-stato i a un macro-stato j è determinata dalla probabilità di transizione

$$p_{ij}(k+1) = p'_{hl}(k+1) \cdot p''_{mn}(k+1), \tag{2.5}$$

dove i macro-stati i, j appartengono a R, gli stati h, l appartengono a $R = R'$ e m, n a R''.

La parte deterministica dello stato, nello stage $k+1$, è costituita da due componenti: la potenza (media) allocata nel k-esimo slot, che corrisponde alla somma dei contributi di tutte le appliance attivate nel medesimo slot, e il vettore delle durate (residue) $\underline{\mathbf{s}}_{ATT_{k+1}}$, che viene aggiornato a seconda delle appliance inserite nello slot k. Come si vedrà meglio in seguito, l'allocazione di potenza rappresenta l'azione di controllo, pertanto la indichiamo con u_k. Possiamo dunque scrivere

$$x_{k+1} = \big(u_k,\ \underline{\mathbf{s}}_{ATT_{k+1}}\big).$$

Nel nostro modello inoltre la funzione di stato $f_k(\cdot)$ dipende dal controllo u_k (come nel caso generale visto nel paragrafo 2.4) e dal vettore $\underline{\mathbf{s}}_{ATT_k}$:

$$x_{k+1} = f_k\big(u_k,\ \underline{\mathbf{s}}_{ATT_k}\big).$$

In questo caso, dunque, una componente dello stato non solo dipende dal controllo effettuato nello slot precedente, ma coincide proprio con esso. Confrontando i termini a destra delle due ultime equazioni si può giungere alla seguente semplificazione:

$$x_{k+1} = \big(u_k,\ f_k(\underline{\mathbf{s}}_{ATT_k})\big),$$

dove la funzione di stato si riduce a svolgere una semplice operazione: indicato con A_k l'insieme delle appliance attivate nello stage k, essa decrementa il numero di slot residui di ogni attività appartenente ad A_k:

$$\underline{\mathbf{s}}_{ATT_{k+1}}(n) = \underline{\mathbf{s}}_{ATT_k}(n) - 1, \qquad \forall n \in A_k.$$

2.5.2 Controllo

Ad ogni stage, l'algoritmo di scheduling deve decidere se inserire o meno una determinata appliance nel suo piano per i successivi 30 minuti. Indicando con "1" l'inserimento e con "0" l'azione opposta, possiamo definire un vettore binario $\underline{\mathbf{b}}$ di lunghezza N, che contenga un elemento per ogni attività controllabile. L'azione scelta dall'ottimizzatore allo stage k corrisponde, come osservato prima, alla potenza media da allocare durante tutto il medesimo slot, la quale dipende dalle appliance che verranno aggiunte. In poche parole, tutte le attività poste a "1" contribuiranno all'aumento del livello di potenza, nella misura specificata dal proprio

profilo energetico $\underline{\mathbf{e}}_n$. Per selezionare di volta in volta l'elemento corretto del profilo di una determinata appliance, si sottrae la sua durata residua nel k-esimo slot alla sua durata totale, dichiarata in $\underline{\mathbf{s}}_{ATT_0}$; il risultato sarà l'indice dell'elemento cercato:

$$u_k = \sum_{n \in A_k} \underline{\mathbf{e}}_n \big(\underline{\mathbf{s}}_{ATT_0}(n) - \underline{\mathbf{s}}_{ATT_k}(n) \big) \underline{\mathbf{b}}_k(n). \tag{2.6}$$

Volendo fare un esempio, supponiamo di dover inserire nello scheduling una appliance a_0 della durata complessiva di 4 slot. Supponiamo inoltre di essere nello stage 2, e di aver già allocato in precedenza la potenza richiesta nei primi 30 minuti di attività ($\underline{\mathbf{e}}_0(0)$), per cui $\underline{\mathbf{s}}_{ATT_2}(0) = 3$. Se l'algoritmo decide di inserire nuovamente l'appliance, dovrà allocare nel secondo slot una potenza media pari al valore di

$$\underline{\mathbf{e}}_0\big(\underline{\mathbf{s}}_{ATT_0}(0) - \underline{\mathbf{s}}_{ATT_2}(0)\big) = \underline{\mathbf{e}}_0\big(4-3\big) = \underline{\mathbf{e}}_0\big(1\big),$$

che corrisponde proprio al secondo elemento del profilo energetico. Quando, per qualche $k \leq W$, $\underline{\mathbf{s}}_{ATT_k}(0) = \underline{\mathbf{s}}_{ATT_0}(0)$, l'appliance sarà considerata terminata e pertanto non potrà più essere inserita nel piano.

Ricordando il limite di potenza istantanea prelevabile dalla rete, da noi fissato a $r_{MAX} = 3$ kW, è necessario scegliere il controllo u_k tale per cui la potenza complessiva, data dal contributo delle attività inserite e di quelle simultanee non controllabili, non superi tale limite. La condizione dovrebbe valere in ogni istante, tuttavia nel modello essa viene riformulata in termini di potenza media su uno slot, pertanto non può essere pienamente garantita. Ciò detto, avendo comunque fissato la soglia critica a un valore inferiore rispetto a quello reale (3.3 kW) e supponendo ragionevolmente di non avere una grande varianza di potenza (sia per u_k che per r_k) nell'intervallo, possiamo concludere che la riformulazione sia adeguata:

$$0 \leq u_k \leq r_{MAX} - r_k, \qquad k = 0, 1, ..., W-1. \tag{2.7}$$

Durante la fase di minimizzazione del costo, verranno calcolati i vari u_k possibili, a partire da $u_k = 0$ (corrispondente al vettore binario interamente nullo). Se una certa azione di controllo non rispetta il limite superiore appena visto, essa non appartiene all'insieme $U(x_k)$ e quindi viene scartata, non potendo essere inserita in una politica π ammissibile. A parità di costo tra due decisioni, viene data la priorità a quella che coinvolge l'appliance con la dead-line più stringente.

Nel primo slot di "vita" di un'appliance in realtà non è, in generale, accettabile l'ipotesi fatta in merito alla ridotta varianza di u_k nel tempo, a causa del picco di potenza assorbita nei primi minuti (par. 2.3). Quando si verifica la condizione (2.7), per tutte le appliance inserite per la prima volta nello scheduling si decide allora di sostituire il valore medio di potenza contenuto in $\underline{\mathbf{e}}(0)$ con il valore di picco contenuto in $\underline{\mathbf{P}}_{ATT}$ (par. 2.3). Indichiamo con $A_k^{(1)}$ il sottoinsieme delle appliance inserite per la prima volta e con $A_k^{(2)}$ il sottoinsieme delle altre appliance inserite nello stesso slot k, tali che $A_k^{(1)} \cup A_k^{(2)} = A_k$. In pratica, occorre calcolare:

$$u_{PEAK_k} = \sum_{n \in A_k^{(1)}} \underline{\mathbf{P}}_{ATT}(n)\, \underline{\mathbf{b}}_k(n) + \sum_{n \in A_k^{(2)}} \underline{\mathbf{e}}_n \big(\underline{\mathbf{s}}_{ATT_0}(n) - \underline{\mathbf{s}}_{ATT_k}(n) \big)\, \underline{\mathbf{b}}_k(n).$$

Ripetiamo comunque che tale grandezza viene utilizzata solo per verificare la condizione (2.7), che diventa:

$$0 \leq u_{PEAK_k} \leq r_{MAX} - r_k, \qquad k = 0, 1, ..., W-1.$$

Quando $A_k^{(1)} = \oslash$, ovvero $A_k = A_k^{(2)}$, si ha che $u_{PEAK_k} = u_k$. Nel calcolo del costo invece si ricorre sempre al valore di u_k definito in (2.6).

Un ulteriore vincolo sul controllo è imposto dal permesso di interrompibilità: se una certa appliance a_m non è interrompibile, una volta avviata dovrà essere inserita nel piano ad ogni slot, fino al termine della sua durata. In altre parole, se siamo nello stage k e $\underline{\mathbf{s}}_{ATT_k}(m) < \underline{\mathbf{s}}_{ATT_0}(m)$, si deve avere:

$$\underline{\mathbf{b}}_k(m) \;=\; \underline{\mathbf{b}}_{k+1}(m) \;=\; ... \;=\; \underline{\mathbf{b}}_{k+\underline{\mathbf{s}}_{ATT_k}(m)-1}(m) \;=\; 1. \tag{2.8}$$

Si potrebbe pensare di ampliare la componente deterministica dello stato x_k con un terzo elemento, un vettore che tenga conto di quali appliance non interrompibili sono state già avviate. Tuttavia questo complicherebbe inutilmente il modello, dato che tale informazione è ricavabile dalla lettura del permesso di interrompibilità INT di ciascuna appliance e dal semplice confronto tra $\underline{\mathbf{s}}_{ATT_k}$ e $\underline{\mathbf{s}}_{ATT_0}$ visto sopra.

Riassumendo, una qualsiasi politica di controllo $\pi = \{u_0, u_1, ..., u_{W-1}\}$ è ammissibile se e solo se ogni decisione rispetta la condizione (2.7), il vincolo (2.8) nel caso di appliance non interrompibili e non include ovviamente appliance già terminate. È corretto precisare dunque che l'insieme delle azioni possibili non dipende esclusivamente dallo stato, ma anche da alcuni

parametri statici:

$$\big(\, r_{MAX},\, \underline{\mathbf{I}}_{ATT}(n),\, \underline{\mathbf{P}}_{ATT}(n)\,\big), \qquad n = 0, 1, ...N-1,$$

dove ricordiamo che $\underline{\mathbf{I}}_{ATT}(n)$ rappresenta il vettore contenente i permessi di interrompibilità e $\underline{\mathbf{P}}_{ATT}(n)$ quello contenente i picchi di potenza (par. 2.3).

2.5.3 Funzione di Costo

Definiamo la funzione di costo $g_k(x_k, y_k, u_k)$, le cui due componenti fondamentali sono associate rispettivamente al costo dell'energia offerta dalla società distributrice e al costo dell'energia rinnovabile, prodotta dagli utenti, ma sprecata, cioè non sfruttata per il funzionamento delle appliance controllabili nonostante la disponibilità. Per semplicità indicheremo l'energia fornita dalle centrali elettriche come "non rinnovabile", e il relativo costo unitario (al kilowattora, kWh) con c_{N_k}. Si assume che, nel generico slot k, essa venga utilizzata solo quando quella rinnovabile non è sufficiente a soddisfare il fabbisogno totale dell'utente, dato da $u_k + r_k$. Viceversa, si spreca energia rinnovabile quando essa è superiore al fabbisogno. Il compito dello scheduler è di conseguenza quello di allocare nello slot la minor potenza possibile qualora vi siano scarse risorse rinnovabili a disposizione, e viceversa la maggior potenza possibile nel caso opposto, per ridurre gli sprechi.

Nel nostro modello assumiamo che il costo unitario dell'energia sprecata sia esattamente uguale a quello dell'energia non rinnovabile. Tuttavia l'energia è considerata "sprecata" nello slot corrente solo se l'utente non la utilizza ma ne avrà effettivamente bisogno in un momento successivo, poiché non tutte le attività controllabili sono giunte al termine. Porre il costo unitario dello spreco pari a quello dell'energia non rinnovabile significa supporre implicitamente che l'energia rinnovabile non sfruttata all'istante k verrà recuperata poi in un qualche istante $k + m$ tramite energia non rinnovabile. Naturalmente ciò non è vero se in $k+m$ sarà ancora disponibile energia rinnovabile. Si tratta quindi di un approccio al caso peggiore, che tende a penalizzare maggiormente lo spreco.

Definiamo la funzione:

$$\{x\}^+ = \begin{cases} x \ \text{ se } x > 0 \\ 0 \ \text{ altrimenti} \end{cases},$$

e indichiamo con $\mu(x)$ la funzione *gradino*:

$$\mu(x) = \begin{cases} 1 \text{ se } x \geq 0 \\ 0 \text{ altrimenti} \end{cases}.$$

Definiamo inoltre l'energia *residua* res_k, nel k-esimo slot, come la somma delle energie ancora da allocare di tutte le appliance controllabili, calcolabile a partire dagli N profili energetici fissati all'inizio:

$$res_k = \sum_{n=0}^{N-1} \sum_{m=\underline{s}_{ATT_0}(n)-\underline{s}_{ATT_k}(n)}^{\underline{s}_{ATT_0}(n)} \underline{\mathbf{e}}_n(m).$$

A questo punto possiamo esprimere la somma dei due contributi di costo descritti sopra, relativi all'energia pagata e a quella sprecata, mediante la seguente espressione, che coinciderà con la prima parte della nostra funzione di costo:

$$c_{N_k}\{u_k + r_k - e_{R_k}\}^+ + c_{N_k} \min\Big\{\{e_{R_k} - u_k - r_k\}^+, res_k\, \mu\big(e_{R_k} - u_k - r_k\big)\Big\}.$$

Si noti che per minimizzare l'espressione è sufficiente fare in modo che $u_k + r_k = e_{R_k}$, ovvero annullare il costo. Di conseguenza, se $r_k \geq e_{R_k}$, il controllore cercherà di abbassare quanto più possibile u_k, mentre se $r_k < e_{R_k}$ cercherà di alzare u_k, in modo da avvicinarsi sempre alla situazione di equilibrio ideale. Questa semplice politica di controllo tuttavia non può portare al raggiungimento dell'obiettivo proposto, poiché si basa esclusivamente sulla situazione nello slot corrente. Al contrario, è necessaria una visione "globale" del piano ad ogni stage, per contrastare i problemi che ora esamineremo.

La politica ottimale adottata nello scheduling deve innanzitutto tenere conto delle scadenze delle varie appliance: il controllore può allocare poca energia, se necessario, per ridurre il costo nello slot corrente, ma deve "accertarsi" di poter allocare l'energia rimanente nei limiti di tempo stabiliti. Per tale motivo abbiamo introdotto nella funzione di costo una terza componente di penalità M, ovvero un ulteriore costo, molto elevato, che viene aggiunto nel caso in cui un'attività non sia terminata entro la propria dead-line.

Il costo totale diventa così:

$$c_{N_k}\{u_k + r_k - e_{R_k}\}^+ + c_{N_k} \min\Big\{\{e_{R_k} - u_k - r_k\}^+, res_k\, \mu\big(e_{R_k} - u_k - r_k\big)\Big\} +$$
$$+ M\mu\big(\underline{\mathbf{s}}_{ATT_k}(n) - 1\big)\mu\big(k - D_n\big), \qquad n = 0, 1, ..., N-1,$$

dove le due funzioni gradino sono entrambe non nulle quando almeno una delle appliance non è ancora terminata ($\underline{\mathbf{s}}_{ATT_k}(n) \geq 1$) e, contemporaneamente, l'istante di decisione corrente k ne ha superato la scadenza ($k \geq D_n$).

Come anticipato all'inizio del capitolo, se l'algoritmo è costretto ad utilizzare l'energia non rinnovabile, per minimizzare il costo finale della politica di controllo esso deve inserire le appliance nelle fasce orarie in cui tale energia viene offerta a una tariffa inferiore (in Italia, dalle ore 19 alle ore 8 del giorno successivo). Una visione "globale" della finestra in cui si colloca il piano consente di determinare se, a partire dall'istante corrente k, sarà disponibile un numero di slot con energia non rinnovabile a basso costo sufficiente al funzionamento delle appliance ancora non terminate. In caso affermativo, l'algoritmo dovrà ritardare il loro inserimento (sempre se le dead-line lo consentono) nel piano previsto. A tale scopo si introduce un'altra penalità M', che viene aggiunta se lo scheduler non ha in quel momento risorse rinnovabili a disposizione e alloca energia prima di arrivare all'inizio dell'intervallo di costo inferiore. Indichiamo con N_{L_k} il numero di slot "*low-cost*" a disposizione a partire dall'istante k, e definiamo la funzione:

$$\big\{x(k)\big\}_{k\in I} = \begin{cases} x(k) \ \text{ se } k \in I \\ 0 \ \text{ altrimenti} \end{cases} .$$

Indicando con H l'intervallo contenente gli slot "*high-cost*", possiamo finalmente giungere alla formulazione completa della funzione di costo:

$$\begin{aligned} & g_k(x_k, y_k, u_k) = \\ & = c_{N_k}\big\{u_k + r_k - e_{R_k}\big\}^+ + \ c_{N_k} \min\Big\{\big\{e_{R_k} - u_k - r_k\big\}^+, res_k\, \mu\big(e_{R_k} - u_k - r_k\big)\Big\} + \\ & + M\mu\big(\underline{\mathbf{s}}_{ATT_k}(n) - 1\big)\, \mu\big(k - D_n\big) + \\ & + M'\left\{ \frac{\{u_k + r_k - e_{R_k}\}^+}{(u_k + r_k - e_{R_k})}\, \mu\big(N_{L_k} - 1\big)\, \underline{\mathbf{b}}_k(n) \right\}_{k\in H}, \qquad n = 0, 1, ..., N-1. \end{aligned} \tag{2.9}$$

L'ultimo termine introdotto è non nullo quando si verificano contemporaneamente le seguenti condizioni: l'istante corrente k si trova all'interno della fascia alta di costo ($k \in H$), la potenza assorbita è superiore a quella derivata dalle risorse rinnovabili ($u_k + r_k > e_{R_k}$), ci sono slot "low-cost" nella finestra ($N_{L_k} \geq 1$) e viene inserita almeno una appliance ($\underline{\mathbf{b}}_k(n) = 1$).

Nell'implementazione dell'algoritmo abbiamo posto $M' < M$, per assegnare la maggior priorità al rispetto delle scadenze piuttosto che alla scelta degli slot low-cost.

Il modello può essere applicato anche nel caso in cui non esistano due fasce di prezzo energetico (che abbiamo indicato come "low-cost" e "high-cost") ben definite, ma si abbiano continue variazioni durante la giornata (vedi par. 2.5.4). È sufficiente fissare nell'implementazione un valore di soglia C_{th} tale per cui tutti i valori del costo unitario previsto maggiori di C_{th} sono considerati come appartenenti alla fascia "high-cost", mentre i valori minori apparterranno alla fascia complementare. Nello stesso caso è possibile anche estendere il modello, considerando F fasce di costo crescente $f_0, f_1, ..., f_{F-1}$, a cui viene associata a ciascuna una penalità M_i, in modo che $M_0 < M_1 < ... < M_{F-1}$, dove $M_0 = 0$ (f_0 è la fascia "low-cost"). Risulta facile generalizzare la (2.9), indicando con N_i il numero di slot appartenenti alla fascia f_i e sostituendo all'ultimo contributo, dipendente da M', il seguente termine:

$$\ldots + \sum_{l=1}^{F-1} M_l \left\{ \frac{\{u_k + r_k - e_{R_k}\}^+}{(u_k + r_k - e_{R_k})} \, \mu\left(\sum_{i=0}^{l-1} N_i \; - 1 \right) \underline{\mathbf{b}}_k(n) \right\}_{k \in f_l} \qquad n = 0, 1, ..., N-1.$$

A questo punto, usando le equazioni (2.4, 2.9), per risolvere il problema proposto dovremo calcolare il costo minimo atteso in ogni stage procedendo a ritroso, secondo le probabilità di transizione di $\{y\}$ definite in (2.5). Il calcolo va eseguito per ogni macro-stato (x_k, y_k). Si osservi che, una volta fissati lo stato (x_k, y_k) e il controllo u_k, la funzione di costo risulta deterministica, poiché nel nostro modello viene a mancare la componente di disturbo w_k. Pertanto, posto $(x_k, y_k) = i$ e $(x_{k+1}, y_{k+1}) = j$, il costo minimo diventa:

$$J(x_W, y_W) = g_W(x_W, y_W),$$

$$\begin{aligned} J(x_k, y_k) &= \min_{u_k} \; E\big\{ g_k(x_k, y_k, u_k) + J(x_{k+1}, y_{k+1}) \big\} = \\ J(x_k, y_k) &= \min_{u_k} \left\{ g_k(x_k, y_k, u_k) + E\left\{ J(x_{k+1}, y_{k+1}) \right\} \right\} = \\ J(i) &= \min_{u_k} \left\{ g_k(i, u_k) + \sum_{j \in R} p_{ij}(k+1) \, J(j) \right\}, \qquad k = 0, 1, ..., W-1. \end{aligned} \tag{2.10}$$

Durante la procedura a ritroso, nel generico slot k è necessario considerare, per ogni possibile macro-stato (x_k, y_k), tutte le possibili azioni u_k (ammissibili), al fine di individuare quella che minimizza $J(i)$. Ciò significa considerare tutti i possibili valori (ammissibili) del

vettore binario di allocazione $\underline{\mathbf{b}}_k$, a partire dal vettore interamente nullo. Una volta determinato il costo minimo per ogni macro-stato, si procede nello stesso modo per lo slot precedente $k-1$, azzerando nuovamente il vettore binario ($\underline{\mathbf{b}}_{k-1}$) alla valutazione di ogni macro-stato (x_{k-1}, y_{k-1}) e cercando ancora il costo minimo associato. Si ripetono le stesse operazioni procedendo gradualmente fino allo slot 0, dove si ha un unico macro-stato possibile (x_0, y_0), quello di partenza.

2.5.4 Costo unitario dell'energia

Il costo unitario c_N subisce delle variazioni dipendenti dal tempo. Si deve pertanto supporre che il controllore locale abbia costantemente a disposizione un vettore di costi unitari $\underline{\mathbf{c}}_N$ per l'intera durata dello scheduling (un elemento per ogni slot), e che questo possa essere aggiornato utilizzando informazioni offerte dal Mercato, ad esempio attraverso una rete locale (*LAN*). Si può pensare tuttavia che tali informazioni (aggiornate) non siano disponibili in ogni slot temporale, e di conseguenza è necessario disporre di un meccanismo di previsione del costo.

Gli autori di [6] propongono un sistema di previsione basato su Media Mobile Pesata (*WMA*, *Weighted Moving Average*), in quanto facile da implementare, a bassa complessità computazionale e in grado di fornire risultati accurati. È stato esaminato l'andamento dei prezzi (in Cent/kWh) dell'energia stabiliti dalla società Statunitense "Illinois Power Company" tra Gennaio 2007 e Dicembre 2009. Analizzando la correlazione temporale del prezzo $\Phi(t)$, dove t indica la distanza tra due valori del prezzo, espressa in numero di giorni, si osservano dei "picchi" in corrispondenza di $t = 1$, $t = 2$ e $t = 7$ (Figura 2.1). Questo indica un'elevata somiglianza del prezzo di "oggi" rispettivamente con quello di "ieri", di due giorni prima e di una settimana prima, che costituiranno quindi gli elementi essenziali del filtro WMA. Indicati con α, β, γ i coefficienti del filtro, il modello di previsione del costo unitario può essere così definito:

$$c_{N_k} = \alpha\, c_{N_{k-W}} + \beta\, c_{N_{k-2W}} + \gamma\, c_{N_{k-7W}}, \qquad k = 0, 1, ..., W-1,$$

dove tutte le sottrazioni tra indici, di tipo $(k-x)$, sono ovviamente da intendersi in modulo W.

In [6] si osservano risultati migliori utilizzando coefficienti del filtro variabili nel corso dei giorni piuttosto che costanti, soprattutto a causa della riduzione del prezzo durante i week-end, che determina una variazione nell'andamento rispetto agli altri giorni della settimana.

In Italia l'andamento del prezzo imposto non subisce variazioni durante la settimana, mentre è possibile usufruire del contratto a tariffa "bioraria", in cui si distingue una fascia di costo

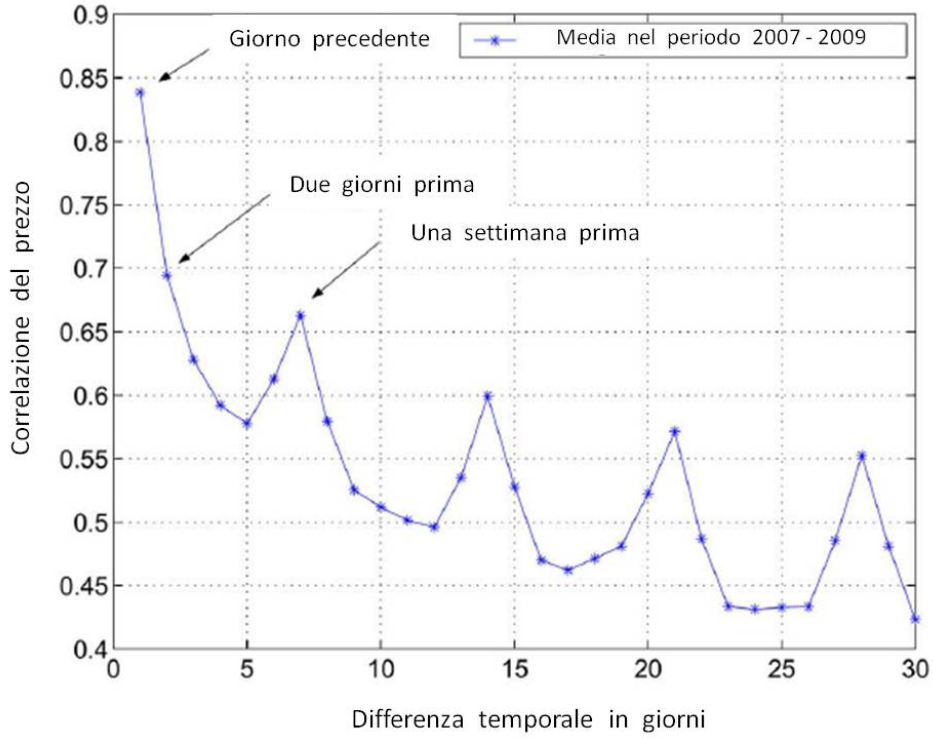


Figura 2.1: Correlazione del prezzo al variare dei giorni.

elevato (ogni giorno dalle ore 8 alle 19, escluso il week-end) e una di costo inferiore (dalle 19 alle 8 del giorno seguente e nel week-end). Le simulazioni del nostro algoritmo di scheduling tengono conto di tale contesto, pertanto non richiedono il sistema di previsione appena descritto, che non è stato implementato.

2.5.5 Durata dell'algoritmo

Abbiamo già specificato che il nostro algoritmo DP è a "Orizzonte Finito", poiché la sua durata massima, in termini di slot in cui si effettuano le decisioni di controllo, è pari a $W = 48$. La durata effettiva in realtà, che possiamo indicare con W', dipende dalla dead-line più lontana $\tilde{D}$ a partire dall'istante di partenza. Superata questa infatti, tutte le appliance devono essere state inserite nello scheduling, pertanto in ciascuno degli slot successivi, fino a $k = W - 1$, il controllo dovrà essere sempre $u_k = 0$ (onde evitare la penalità per la mancata scadenza). Si decide quindi ragionevolmente di sospendere l'algoritmo una volta compiuta l'ultima decisione utile nello stage $k = W' - 1$, che precede $\tilde{D}$, segnalando un errore nel caso almeno una appliance non fosse condotta a termine entro l'ultimo intervallo.

Capitolo 3

Implementazione dell'algoritmo di scheduling

3.1 Schema generale

L'algoritmo di scheduling presentato nel capitolo precedente è realizzato interamente in linguaggio C. Si riporta lo schema generale in Figura 3.1, dove sono illustrati i vari moduli e file necessari all'esecuzione, che verranno presentati in maggior dettaglio nei paragrafi successivi, e le relative interconnessioni.

La funzione principale è `optimal_scheduler`, contenuta nel file "optimal_scheduler.c", che, a partire da determinati parametri di ingresso, tra cui l'istante di avvio, esegue la procedura a ritroso prevista dalla tecnica di Programmazione Dinamica e restituisce in output un valore indicante quali appliance devono essere inserite nello slot corrente (i primi 30 minuti successivi all'avvio della procedura). Il file eseguibile, compilato da "main.c", richiama tale funzione all'inizio di ogni slot della finestra temporale, finché tutte le appliance giungono al termine o comunque si raggiunge lo stage che precede l'ultima dead-line specificata (segnalando le appliance che non sono state completate nel periodo stabilito).

In alternativa, sarebbe possibile eseguire la procedura DP solo una volta, all'istante di avvio, prevedere l'evoluzione delle due MC (richiesta non controllabile ed energia rinnovabile) e selezionare quindi le azioni di controllo ottime in ogni slot associate ai macro-stati previsti, già individuate durante la procedura a ritroso. La politica di controllo ottima risultante determinerebbe lo schedule definitivo. La previsione dell'evoluzione di una MC può essere compiuta,

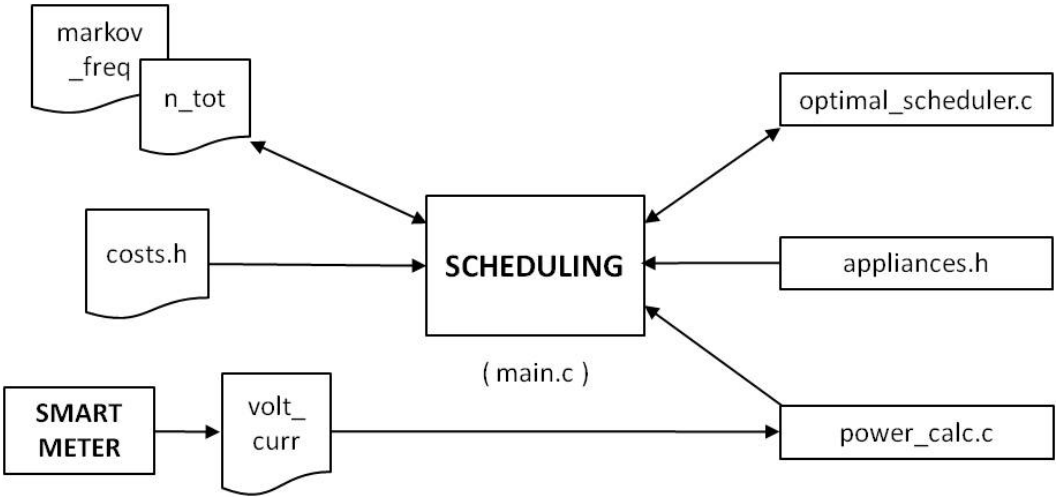


Figura 3.1: Moduli utilizzati nell'implementazione dell'algoritmo.

partendo dallo stato i assunto nello slot k, calcolando di volta in volta l'aspettazione dello stato successivo j secondo le probabilità di transizione $p_{ij}(k+1)$ (e arrotondando a un valore intero il risultato).

Questa alternativa, tuttavia, appare molto poco robusta rispetto ai possibili errori di previsione che l'algoritmo può commettere. In altre parole, i valori dei due processi aleatori misurati all'inizio dello slot potrebbero differire anche di molto da quelli previsti all'avvio, e ciò rischierebbe di compromettere la fattibilità dello schedule prodotto, o comunque la sua ottimalità. Per tale motivo, abbiamo scelto di eseguire la procedura DP, come già detto, all'inizio di ogni slot, in modo da riuscire a correggere lo schedule in tempo reale in caso di previsioni errate.

Tutte le appliance che devono essere inserite nello scheduling sono specificate all'interno del file "appliances.h", che ne raccoglie i profili energetici, le dead-line, i permessi di interrompibilità e i valori di picco della potenza assorbita.

Prima di ogni chiamata a `optimal_scheduler`, si procede con il calcolo e la lettura della potenza attiva richiesta dall'utente e della potenza attiva derivata dall'energia rinnovabile. Sarebbe tuttavia poco sensato basarsi esclusivamente sul primo valore, determinato all'inizio dello slot, e supporre che esso coincida con il valore medio della potenza attiva durante l'intero intervallo. Entrambe le misure sono pertanto "corrette" tramite una combinazione lineare con i rispettivi valori medi registrati 24 ore prima. Indichiamo con α il coefficiente di combinazione lineare per la richiesta e con β quello per l'energia rinnovabile; indichiamo poi con $r^{(i)}$ e $e_R^{(i)}$ i rispettivi i-esimi campioni di potenza calcolati all'interno di uno slot, e infine con $\bar{r}_k$ e $\bar{e}_{R_k}$ i

valori medi delle due grandezze. Definito K il numero totale di questi campioni (che assumiamo uguale per entrambe le grandezze) in un qualunque slot, formalizziamo quanto appena detto nelle seguenti relazioni:

$$\bar{r}_k = \frac{1}{K}\sum_{i=0}^{K-1} r_k^{(i)};$$

$$r_{k+W} = \alpha\, r_{k+W}^{(0)} + (1-\alpha)\bar{r}_k;$$

$$\bar{e}_{R_k} = \frac{1}{K}\sum_{i=0}^{K-1} e_{R_k}^{(i)};$$

$$e_{R_{k+W}} = \beta\, e_{R_{k+W}}^{(0)} + (1-\beta)\bar{e}_{R_k};$$

I coefficienti α, β determinano il "peso" assegnato al campione iniziale o alla media registrata in passato: se essi sono fissati a un valore maggiore di 0.5, il campione presente avrà importanza maggiore rispetto al passato.

L'algoritmo assumerà costanti le due misure r_{k+W} e $e_{R_{k+W}}$ appena calcolate durante tutto l'intervallo di 30 minuti, mentre si continueranno a produrre periodicamente i campioni che, una volta mediati, saranno utilizzati in maniera analoga per le 24 ore successive. La Figura 3.2 illustra il procedimento appena descritto, con riferimento alla sola r_k.

Lo Smart Meter nel nostro caso rileva separatamente la tensione e la corrente istantanee. I dati, una volta ricevuti dal server, vengono salvati sul file "volt_curr". Il calcolo delle potenze, nella realizzazione attuale dello Smart Meter, non viene quindi effettuato direttamente dal dispositivo, ma dal lato server mediante la funzione `power_calc`, definita all'interno del modulo "power_calc.c", a partire dalla lettura di "volt_curr".

Dopo ogni chiamata a `optimal_scheduler` (e prima della successiva) si deve aggiornare il profilo energetico di una determinata attività e, conseguentemente, il numero di slot rimasti, se questa è stata inserita nello scheduling.

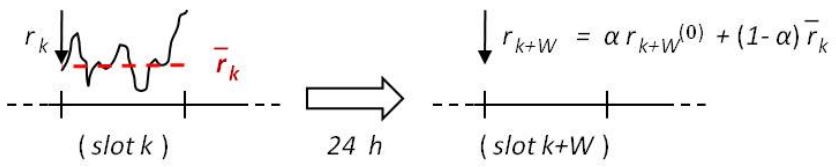


Figura 3.2: Calcolo della richiesta non controllabile di potenza all'inizio dello slot.

Il file "costs.h" contiene il vettore dei costi unitari dell'energia, espressi in euro/kWh. Come visto nel paragrafo 2.5.4, nell'implementazione attuale si tratta di un vettore statico, in cui gli elementi non evolvono in base a un qualche meccanismo di previsione. Nello stesso modulo sono definite le penalità utilizzate nel calcolo della funzione di costo $g_k(\cdot)$ (2.9).

I file "markov_freq" e "n_tot" sono necessari per l'aggiornamento delle MC relative ai processi $\{r\}$ e $\{e_R\}$. Come già visto, al valore di potenza media calcolata nel generico slot k è associato un certo stato i della Catena, e analogamente, al valore nello slot successivo $(k+1)$ è associato uno stato j. Il file "markov_freq" contiene il numero di volte in cui si è verificata la transizione n_{ij}, per ogni coppia (i, j) appartenente allo spazio degli stati, mentre "n_tot" contiene il numero complessivo di transizioni che hanno i come stato iniziale, per ogni i appartenente allo spazio degli stati. Dopo ogni transizione è possibile ricorrere alla (2.1) per aggiornare le due MC. Nella versione dell'algoritmo effettivamente utilizzata, su cui sono stati eseguiti i vari test, non è stata tuttavia implementata la procedura di aggiornamento, come vedremo meglio nel paragrafo 3.5.

3.2 Quantizzazione della potenza media

Nell'implementazione attuale dell'algoritmo, l'intervallo di valori assumibili dalla potenza media assorbita (o disponibile, nel caso dell'energia rinnovabile) in ciascuno slot temporale è stato definito tra 0 e 3000 W (par. 2.2), e quantizzato uniformemente in dieci possibili stati. La Tabella 3.1 specifica i valori di potenza media P compresi in ogni stato, che indichiamo con s.

L'algoritmo svolge tutte le sue operazioni utilizzando i valori interi adimensionali associati ai vari stati (che possiamo definire valori "nominali"), fatta eccezione per il calcolo del costo monetario e dell'energia residua di ciascuna appliance (l'energia ancora da allocare nei vari slot di attività, espressa in kWh). Tali grandezze sono visualizzabili al termine di ogni chiamata a `optimal_scheduler`. In entrambi i casi, è sufficiente moltiplicare il valore nominale dello stato per una costante opportuna. Il risultato del calcolo non può comunque essere esatto, dato che a ciascun valore nominale corrispondono infiniti valori di potenza media. Riducendo la granularità degli stati si migliora naturalmente la precisione del calcolo, a scapito della complessità computazionale.

Nell'implementazione si è deciso di sovrastimare i valori del costo e dell'energia residua: se γ e ϵ sono le rispettive costanti opportune da moltiplicare per i valori nominali dello stato (s_c

Stato	Potenza media (W)
0	$0 \leq P < 300$
1	$300 \leq P < 600$
2	$600 \leq P < 900$
3	$900 \leq P < 1200$
4	$1200 \leq P < 1500$
5	$1500 \leq P < 1800$
6	$1800 \leq P < 2100$
7	$2100 \leq P < 2400$
8	$2400 \leq P < 2700$
9	$2700 \leq P < 3000$

Tabella 3.1: Definizione degli stati rappresentanti la potenza media in ogni slot.

per il costo, s_e per l'energia residua), si ha che:

$$costo\ monetario = \gamma\ (s_c + 1),$$

$$energia\ residua = \epsilon\ (s_e + 1).$$

La costante γ è quella che abbiamo indicato con c_N nel paragrafo 2.5.3, normalizzata però secondo la durata degli slot (30 minuti) e l'unità di misura convenzionale dell'energia (kWh). Nel caso dell'Italia, dobbiamo distinguere tra la costante $\gamma^{(l)}$, relativa alla fascia di prezzo inferiore, e $\gamma^{(h)}$, relativa alla fascia più alta. Il costo unitario c_N nelle due fasce di prezzo è attualmente di circa 0.14 e 0.16 euro al kWh, pertanto definiamo

$$\gamma^{(l)} = \frac{c_N^{(l)} \cdot 300 \cdot 10^{-3}}{2} \approx 0.021 \quad [\,euro\,],$$

$$\gamma^{(h)} = \frac{c_N^{(h)} \cdot 300 \cdot 10^{-3}}{2} \approx 0.024 \quad [\,euro\,],$$

che esprimono i costi unitari associati alla durata dello slot e alla granularità di 0.3 kW. Volendo fare un esempio, supponiamo che la potenza media assorbita in un certo intervallo, appartenente alla fascia "low-cost", sia tale da trovarsi nello stato 2. Il costo stimato per lo slot è allora pari a $0.021\ (2 + 1) \approx 0.063$ euro. In pratica, è stato associato allo stato 2 un consumo di $(2+1) \cdot 300 = 900$ W, indipendentemente dal fatto che la potenza assorbita fosse effettivamente

molto vicina al limite superiore dello stato (quasi 900 W) o invece molto vicina al limite inferiore (600 W o poco più). Ciò comporta di fatto un approccio al caso peggiore, e quindi una stima in eccesso dei costi, come spiegato sopra. A nostro avviso, una stima in difetto sarebbe meno idonea, poiché porterebbe l'utente a sottovalutare maggiormente gli eccessi di potenza assorbita.

In modo simile a quanto visto per γ, la costante ϵ è definita come

$$\epsilon = \frac{300 \cdot 10^{-3}}{2} = 0.15 \quad [\, kWh \,].$$

Supponiamo, ad esempio, che in un certo istante siano rimasti, per l'appliance a_0, due slot di attività da inserire nello scheduling, e che gli stati della potenza media previsti nei due slot siano 1 e 0. L'energia residua stimata è pari a $0.15 \cdot (1 + 1) = 0.3$ kWh per il primo slot e $0.15 \cdot (0 + 1) = 0.15$ kWh per il secondo. Anche in questo caso, il valore è stimato in eccesso rispetto a quello reale.

Il limite superiore di potenza prelevabile dalla rete, che abbiamo precedentemente indicato con r_{MAX}, è rappresentato da un ulteriore stato $s = 10$, che secondo il criterio di quantizzazione adottato ha come valore minimo di potenza media 3000 W e come valore massimo (ma non raggiungibile) 3300 W.

3.3 Analisi degli elementi essenziali

3.3.1 Funzioni ausiliarie

Sono state definite innanzitutto alcune funzioni fondamentali, utilizzate in vari punti dei moduli software, di cui presentiamo ora brevemente le più importanti.

max_24 È importante osservare che gli operatori relazionali classici ($<, >$) non possono essere adottati per confrontare istanti temporali espressi nel formato delle 24 ore, poiché non solo i loro valori si ripetono ciclicamente col trascorrere di un giorno, ma soprattutto la relazione tra due istanti dipende dal giorno in cui ciascuno di essi si colloca. A titolo di esempio, consideriamo i due istanti $t_1 = 22:00$ e $t_2 = 7:30$. Se supponiamo che i due orari si riferiscano allo stesso giorno, è evidente che $t_2 < t_1$, nel senso che t_2 precede t_1. Ma se supponiamo che il primo orario si riferisca al giorno precedente, risulta $t_2 > t_1$.

La funzione `max_24` individua il massimo (cioè il più lontano nel tempo) tra due o più istanti (disposti in un vettore). Per evitare l'ambiguità descritta sopra, oltre all'array di elementi da

confrontare, viene passato come parametro l'istante di partenza t_{start} (che può anche coincidere con uno degli elementi). Esso costituisce un riferimento: tutti gli istanti maggiori o uguali a t_{start} precedenti le ore $00:00$ sono considerati come appartenenti al medesimo giorno; tutti gli istanti inferiori sono considerati come appartenenti al giorno successivo (e quindi più lontani ancora, sebbene il loro valore sia, appunto, più piccolo). Tornando all'esempio visto sopra, se passiamo alla funzione $t_{start} = t_2$ e il vettore $[\, t_1,\ t_2\,]$, l'output prodotto sarà t_1, mentre se passiamo $t_{start} = t_1$ e lo stesso vettore, l'output sarà t_2.

La funzione viene utilizzata per effettuare confronti tra due dead-line o tra l'istante in cui si trova l'algoritmo durante la procedura a ritroso di DP e una dead-line. Se viene chiamata iterativamente può inoltre essere sfruttata per un ordinamento di istanti in formato 24 ore, come avviene nella funzione `sort_dl`, che ora esamineremo.

sort_dl Ordina le dead-line delle N appliance controllabili, passate in un unico vettore, restituendo per ogni elemento la posizione corrispondente nell'ordine corretto, il tutto nell'array `sorted_ind` . Non viene restituito direttamente il vettore delle dead-line ordinato in quanto gli stessi indici di `sorted_ind` sono usati anche per ordinare i vettori contenenti le durate, i permessi di interrompibilità e i nomi delle appliance, nonché le righe della matrice che ne raccoglie i profili energetici.

Consideriamo, ad esempio, le scadenze di cinque appliance disposte nel seguente array:

$$\texttt{dl} = [\, 9:00,\ 6:30,\ 19:30,\ 23:30,\ 20:00\,],$$

e supponiamo di voler avviare l'algoritmo di scheduling alle ore $17:00$. La funzione `sort_dl` chiamerà ripetutamente `max_24` fissando $t_{start} = 17:00$. Alla prima iterazione, l'istante più lontano nel tempo da t_{start} risulterà quello delle ore $9:00$, a cui verrà associato pertanto l'indice 4. Alla seconda iterazione verrà escluso l'elemento già selezionato, pertanto `max_24` individuerà come nuovo "massimo" l'istante delle ore $6:30$, a cui verrà assegnato l'indice 3. Si procederà in maniera analoga finché si otterrà l'array

$$\texttt{sorted\_ind} = [\, 4,\ 3,\ 0,\ 2,\ 1\,],$$

e seguendo le indicazioni degli indici, il vettore ordinato delle dead-line sarà (in senso crescente da sinistra verso destra):

$$\texttt{sorted\_dl} = [\, 19:30,\ 20:00,\ 23:30,\ 6:30,\ 9:00\,].$$

L'ordinamento delle attività secondo le rispettive dead-line consente allo scheduler di riconoscere i livelli di priorità e dunque, a parità di costo, di allocare sempre prima l'energia delle attività più urgenti.

dec2bin Converte un intero (positivo o nullo) nel *vettore* binario corrispondente, i cui elementi coincidono cioè con i simboli dell'equivalente binario. La funzione è usata innanzitutto dall'algoritmo per la creazione del vettore $\underline{\mathbf{b}}$, che indica l'inserimento o meno di ciascuna appliance nelle azioni di controllo (vedi par. 2.5.2). Inoltre, la funzione `optimal_scheduler` restituisce, al termine della procedura DP, un intero, che deve essere codificato nel main da `dec2bin` per evidenziare il corrispondente vettore ottimo $\underline{\mathbf{b}}_0^*$, ovvero le appliance da inserire nell'istante corrente.

Volendo fare un esempio, supponiamo che il numero di appliance schedulabili sia $N = 6$ e che, alla fine di una certa iterazione, avviata alle ore $11 : 00$, `optimal_scheduler` restituisca il valore 26. Il vettore binario che otteniamo chiamando `dec2bin(26)` sarà $\underline{\mathbf{b}}_0 = [\,0,\ 1,\ 1,\ 0,\ 1,\ 0\,]$, pertanto dalle $11 : 00$ alle $11 : 30$ sarà programmato il funzionamento delle attività a_1, a_2, a_4, associate rispettivamente agli indici degli elementi non nulli di $\underline{\mathbf{b}}_0$.

3.3.2 La funzione "power_calc"

Calcola la potenza istantanea (attiva) a partire dai valori di tensione e corrente trasmessi dallo Smart Meter. Indichiamo i due segnali sinusoidali di tensione e corrente campionati con

$$v(t) = V_M \sin(2\pi f_0 t + \alpha);$$

$$i(t) = I_M \sin(2\pi f_0 t + \beta).$$

Se ci limitiamo a considerare la rete elettrica Europea, la frequenza f_0 deve essere fissata a 50 Hz. Se indichiamo con $V = V_M/\sqrt{2}$ e $I = I_M/\sqrt{2}$ i valori efficaci delle rispettive ampiezze, e con $\phi = \alpha - \beta$ lo sfasamento tra tensione e corrente, la potenza attiva è pari a

$$P = VI\cos\phi.$$

La funzione determina innanzitutto lo sfasamento ϕ ricorrendo all'analisi spettrale dei due segnali. Viene calcolata dunque la *DFFT* (*Discrete Fast Fourier Transform*) delle sinusoidi

di tensione e corrente. Nel caso ideale, il contenuto in frequenza di entrambe è costituito esclusivamente da due componenti principali centrate in $\pm f_0$, di ampiezza esattamente pari a $V_M/2$ nel caso della tensione e $I_M/2$ nel caso della corrente. Nella pratica, a causa del rumore e della discretizzazione delle frequenze, tali ampiezze possono essere soggette ad errore, e si aggiungono alcune componenti spurie in altre frequenze. È opportuno ricordare che la DFFT deve essere calcolata a partire da un segnale periodico, e quindi su un multiplo intero K di periodi della sinusoide di ingresso. Per aumentare l'accuratezza del calcolo, non ci siamo limitati ad un solo periodo, ma abbiamo posto $K = 40$.

Una volta individuate le componenti principali in frequenza di $v(t)$ e $i(t)$, che indichiamo con i valori z_v e z_i , entrambi appartenenti al piano complesso, la funzione ne calcola la fase a partire dalla parte reale e dalla parte immaginaria:

$$\phi_v = \arctan 2\left(\frac{\Im\, z_v}{\Re\, z_v}\right),$$

$$\phi_i = \arctan 2\left(\frac{\Im\, z_i}{\Re\, z_i}\right),$$

dove ricordiamo che la funzione "arcotangente2" è in grado di determinare univocamente, a differenza della semplice arcotangente, le coordinate di un punto del piano (x, y) tali per cui y/x rappresenta il valore della tangente associata. In questo caso y rappresenta la parte immaginaria del numero complesso e x quella reale.

A questo punto è facile calcolare lo sfasamento dei due segnali:

$$\phi = \phi_v - \phi_i.$$

I valori efficaci si ricavano dal modulo delle due componenti principali, peraltro già determinati durante la ricerca di queste ultime. Se indichiamo con N_c il numero di campioni della sinusoide data in ingresso alla trasformata, si ha:

$$V_M = \frac{2}{N_c}\sqrt{\Re\, {z_v}^2 + \Im\, {z_v}^2},$$

$$I_M = \frac{2}{N_c}\sqrt{\Re\, {z_i}^2 + \Im\, {z_i}^2},$$

da cui $VI = V_M V_I/2$.

3.4 Il metodo Main

3.4.1 Struttura

All'inizio del metodo main vengono create tutte le matrici di transizione necessarie per caratterizzare gli andamenti della richiesta non controllabile dell'utente e dell'energia rinnovabile disponibile durante la giornata. Il file `markov_freq` contiene le frequenze di tutte le possibili transizioni per ogni matrice, cioè i valori che nel paragrafo 2.2 abbiamo indicato con n_{ij}. Il file `n_tot` contiene invece le frequenze di tutte le transizioni a partire da uno stesso stato, che al paragrafo 2.2 abbiamo indicato con n_{TOT_i}. Dall'espressione (2.1) si possono ricavare così le probabilità di transizione vere e proprie.

Dopo aver caricato in memoria le matrici di transizione, viene effettuato un semplice controllo in modo da verificarne la correttezza, ricordando che la somma degli elementi di ogni riga deve essere pari a 1: per ogni stato i e per ogni slot k della finestra di previsione, indicato con S l'insieme degli stati, deve valere

$$\sum_{j \in S} p_{ij}(k) = 1.$$

In caso contrario, se per almeno una riga di qualche matrice la condizione non è soddisfatta, l'algoritmo si interrompe generando un errore.

Successivamente l'utente può scegliere l'ora in cui iniziare la pianificazione delle appliance. In base all'ora scelta, si ordinano le attività privilegiando quelle con scadenze più "stringenti", come già visto. A questo punto si entra nel ciclo principale, di cui si illustrano le operazioni nello pseudo-codice sotto riportato. Le variabili $contr, rich, rinn$ e s_{att} indicano rispettivamente il controllo effettuato nello slot precedente, la richiesta non controllabile e l'energia rinnovabile misurate all'inizio dello slot corrente, e il vettore contenente le durate delle attività $\underline{\mathbf{s}}_{ATT}$. Dopo la chiamata alla funzione `optimal_scheduler`, $contr$ viene aggiornato secondo il valore del controllo u_0, poiché il main, ad ogni iterazione del ciclo principale, considera solo la politica di controllo per lo slot immediatamente successivo.

La funzione `optimal_scheduler` genera immediatamente un errore nel caso una o più appliance non abbiano il tempo sufficiente per essere completate entro il limite stabilito. Se l'errore viene rilevato alla prima iterazione del ciclo principale, si assume che l'utente abbia specificato male qualche dead-line e quindi gli viene data la possibilità di correggere l'inserimento dei dati, sospendendo il processo. Se l'errore viene rilevato successivamente, significa che in

qualche slot l'energia effettiva a disposizione è stata di molto inferiore a quella prevista: ciò ha ritardato l'inserimento di una o più attività, che quindi non riusciranno a terminare in tempo. Appare tuttavia poco sensato interrompere lo scheduling a questo punto, pertanto si preferisce rimuovere dal piano le attività che hanno causato l'errore e proseguire. Al termine del processo tutto ciò viene comunque segnalato all'utente.

Main (ciclo principale):

$completed = 0$;
$contr = 0$
while (! $completed$) **do**
  $aggiorna\ stato \leftarrow (contr, rich, rinn, s_{att})$;
  $optimal\_scheduler(ora\_avvio, stato, ...)$;
  if ($\exists\ deadline\ non\ rispettabile$) **then**
    if ($prima\ iterazione$) **then**
      $interrompi \rightarrow errore$!
    else
      $elimina\ appliance\ con\ scadenza\ non\ rispettabile$;
      $ripeti\ iterazione$;
    end if
  end if
  $aggiorna\ contr$;
  $aggiorna\ s_{att}$;
  $aggiorna\ profili\ energetici$;
  if ($appliance\ tutte\ terminate$) **then**
    $completed = 1$;
  end if
  $segnala\ eventuali\ appliance\ rimosse$;
  $attendi\ 30\ min$;
  $aggiorna\ ora\_avvio$;
end while

La funzione `optimal_scheduler`, all'inizio della sua esecuzione, controlla che tutte le appliance siano potenzialmente terminabili entro le proprie scadenze, ovvero, usando la notazione dello pseudo-codice e indicando con D_n una generica dead-line:

$$ora\_avvio + s_{att}(n) \leq D_n, \qquad n = 0, 1, ..., N-1,$$

dove l'operazione di somma è ovviamente da intendersi nel formato 24 ore, e la relazione "$\leq$" indica la precedenza (o al più coincidenza) tra i due istanti.

Nel caso il controllo non andasse a buon fine, `optimal_scheduler` restituisce il valore "-1", e salva su un array (accessibile anche dal main) gli indici delle appliance che hanno provocato l'errore. Una volta ricevuto l'errore, l'algoritmo si comporta come illustrato nello pseudo-codice, rimuovendo tali attività dall'insieme di partenza. Si tratta di una rimozione "fittizia", poiché le corrispondenti durate sono poste a zero e le dead-line sono poste al valore dell'ultima dead-line della finestra. In questo modo le appliance "eliminate" continueranno ad essere presenti nella lista specificata dall'utente, ma verranno ignorate.

3.4.2 Osservazioni sull'esecuzione iterata dell'algoritmo DP

È opportuno fare, in primo luogo, alcune precisazioni riguardanti il vettore contenente le durate delle appliance schedulabili. Dal punto di vista del main, adottando ancora la notazione dello pseudo-codice, esso è rappresentato da s_{att}, i cui elementi vengono decrementati di un'unità ad ogni iterazione del ciclo `while` principale nel caso in cui le corrispondenti attività siano inserite nello slot corrente. Se lo scheduling va a buon fine, si esce dal ciclo quando tutti gli elementi di s_{att} sono nulli.

Dal punto di vista della funzione `optimal_scheduler`, ad ogni chiamata s_{att}, passato come parametro, coincide con quello che nel paragrafo (2.5) abbiamo indicato con $\underline{\mathbf{s}}_{ATT_0}$, cioè con il vettore delle durate di partenza. Dobbiamo pensare che la funzione procede a ritroso, seguendo i passi dell'algoritmo DP, pertanto dovrà utilizzare un vettore temporaneo $\underline{\mathbf{s}}\,'_{ATT}$. All'inizio viene calcolata la durata effettiva W' dell'algoritmo, come visto nel paragrafo (2.5.5), e viene posto $\underline{\mathbf{s}}\,'_{ATT_{W'}} = \underline{\mathbf{0}}$ (vettore nullo). Man mano che si procede verso l'istante iniziale ($k = 0$), gli elementi di $\underline{\mathbf{s}}\,'_{ATT_k}$ vengono incrementati nel caso in cui le corrispondenti attività siano inserite nello slot. Se $\underline{\mathbf{s}}\,'_{ATT_0} = s_{att}$ la procedura è andata a buon fine, e nel main l'operazione $s_{att} = \underline{\mathbf{s}}_{ATT_1}$ aggiornerà il vettore prima della successiva chiamata a

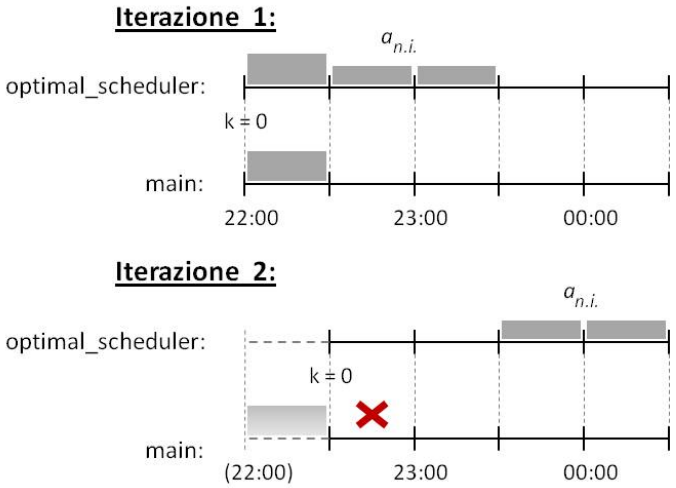


Figura 3.3: Nell'iterazione 1 viene aggiunta l'attività non interrompibile $a_{n.i.}$, a partire da $k = 0$, pertanto essa viene inserita anche nel main; nell'iterazione 2 ne viene rispettata la non-interrompibilità durante l'algoritmo DP, ma essa non viene più aggiunta a partire da $k = 0$, quindi non viene inserita nel main e la non-interrompibilità risulta comunque violata. La politica di controllo dell'iterazione 2 non è ammissibile.

`optimal_scheduler`. In caso contrario, viene inviato un messaggio di errore senza sospendere il processo (confidando in uno scheduling più "fortunato" alla prossima chiamata).

Un'altra importante osservazione riguarda l'inserimento delle appliance non interrompibili. Abbiamo già ricavato nel paragrafo 2.5.2 la condizione (2.8), che deve essere imposta sulle azioni di controllo quando una di queste attività viene inserita nello scheduling. Tale condizione deve valere naturalmente anche a ritroso, durante l'esecuzione dell'algoritmo DP. Quando, nella parte finale di una chiamata a `optimal_scheduler`, l'energia di una appliance non interrompibile $a_{n.i.}$ viene allocata nel primo slot della finestra ($k = 0$), nel main $a_{n.i.}$ viene effettivamente considerata "avviata", pertanto, nell'iterazione successiva del ciclo principale, essa dovrà essere lasciata in funzione per rispettare la "non-interrompibilità". È necessario allora che, in questa seconda chiamata a `optimal_scheduler`, l'insieme delle azioni di controllo ammissibili venga ulteriormente ristretto, in modo da inserire nuovamente l'appliance a partire da $k = 0$. In altre parole, si deve garantire la "non-interrompibilità" non solo all'interno dello scheduling provvisorio generato dall'algoritmo DP, ma anche nello scheduling vero e proprio (vedi Figura 3.3), che si viene a formare col susseguirsi delle chiamate nel main.

Nell'implementazione si utilizza a tale scopo il vettore binario `nonInt_on`, in cui, dopo la

chiamata a `optimal_scheduler`, vengono posti a "1" gli elementi corrispondenti alle attività non interrompibili inserite nell'iterazione corrente del `while` . Il vettore viene quindi passato come parametro a `optimal_scheduler` nella chiamata successiva: a questo punto l'algoritmo DP, partendo da $k = W' - 1$, deve attendere lo slot opportuno per inserire le appliance "marcate". In particolare, per ogni n tale che `nonInt_on`$[\, n \,] = 1$, deve valere la condizione:

$$\underline{\mathbf{b}}_k(n) = \begin{cases} 1 \text{ se } k < s_{att}(n) \\ 0 \text{ altrimenti} \end{cases} .$$

Supponiamo ad esempio che l'appliance non interrompibile avviata nel main sia a_0, e che il numero di slot rimasti per il completamento sia pari a 3. Nell'iterazione successiva la funzione `optimal_scheduler` può inserire a_0 solo a partire da $k = s_{att}(0) - 1 = 2$. Spostandosi ancora a ritroso, in $k = 1$ e $k = 0$ allocherà tutta l'energia rimasta.

Dobbiamo ricordare infine quanto detto a proposito dei picchi di potenza assorbiti dalle appliance nel primo slot di attività (par. 2.3, 2.5.2): per rispettare la condizione (2.7) sulla massima potenza prelevabile dalla rete, si deve tenere conto del valore di picco $\underline{\mathbf{P}}_{ATT}(n)$ per ogni a_n inserita per la prima volta nel piano. Anche in questo caso occorre "contestualizzare" la funzione `optimal_scheduler` nel main. Supponiamo che l'energia di una certa appliance a_m venga allocata per la prima volta a partire dall'istante $k = 0$ durante l'algoritmo DP. Nel main essa risulta così avviata per la prima volta. All'esecuzione successiva, l'algoritmo DP deve "ricordare" l'allocazione effettuata in precedenza e quindi non deve più utilizzare il valore $\underline{\mathbf{P}}_{ATT}(m)$ nella verifica del limite superiore di potenza.

Per tenere conto di questo, tutti gli elementi di $\underline{\mathbf{P}}_{ATT}$ associati ad attività inserite per la prima volta vengono posti successivamente a zero. La funzione `optimal_scheduler` considererà quindi come "già avviate in passato" tutte le appliance con picco nullo.

In conclusione, ribadiamo (come al par. 2.5.2) che l'insieme U delle azioni di controllo "lecite" dipende non solo dallo stato del sistema, ma anche da un certo numero di parametri esterni, a cui si deve aggiungere il vettore indicante le appliance non interrompibili già avviate nell'esecuzione precedente dell'algoritmo DP, che annotiamo con $\underline{\mathbf{N}}_{ON}$:

$$\left(r_{MAX},\, \underline{\mathbf{I}}_{ATT}(n),\, \underline{\mathbf{P}}_{ATT}(n),\, \underline{\mathbf{N}}_{ON}(n) \right) \qquad n = 0, 1, ...N-1.$$

Tutti questi parametri sono visti effettivamente come "statici" dall'algoritmo DP, nel senso che non evolvono durante una *stessa* esecuzione, pertanto risulterebbe poco ragionevole inglobarli all'interno dello stato del sistema. Tuttavia, dal punto di vista del main, gli

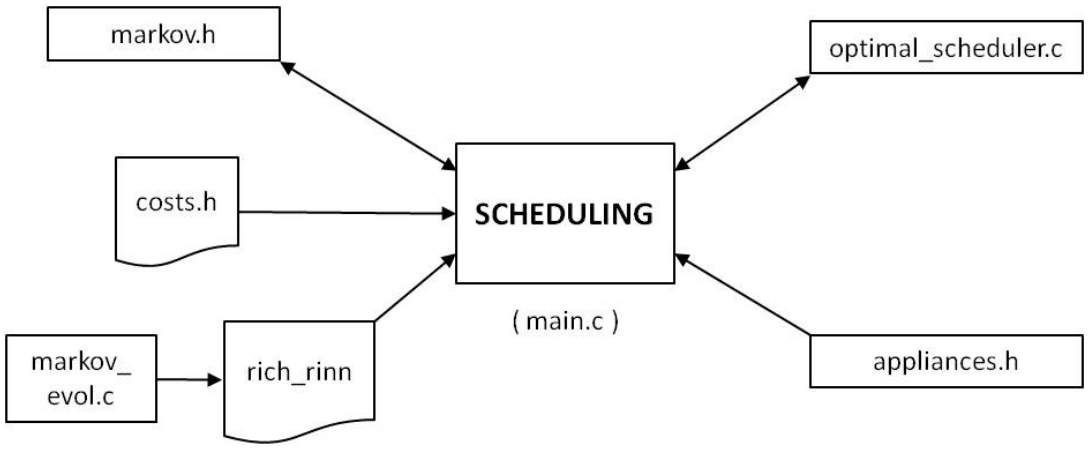


Figura 3.4: Moduli adottati nella versione dell'algoritmo usata per le simulazioni.

ultimi due ($\underline{\mathbf{P}}_{ATT}$, $\underline{\mathbf{N}}_{ON}$) possono considerarsi "dinamici", poiché in chiamate diverse di `optimal_scheduler`, cioè in esecuzioni *diverse* dell'algoritmo DP, i loro elementi possono variare.

3.5 Versione per le simulazioni

Lo schema generale dell'algoritmo di scheduling usato nelle simulazioni presenta alcune modifiche, come si osserva dalla Figura 3.4.

Innanzitutto si utilizzano delle matrici di transizione supponendo che le MC di $\{r\}$ e $\{e_R\}$ siano già "a regime", ovvero siano il risultato di un numero sufficientemente elevato di iterazioni dell'algoritmo (nel corso di più giorni), tale da poter ritenere che da quel momento in poi gli elementi delle matrici subirebbero modifiche trascurabili. Questa osservazione ci consente di evitare la procedura di aggiornamento delle matrici ed eseguire i test ignorando la fase di assestamento iniziale delle MC, in cui esse non modellano correttamente le due grandezze aleatorie e pertanto possono dare maggiori errori nella previsione del loro comportamento.

Le matrici a regime si trovano nel file "markov.h", che viene caricato direttamente dal main.

In secondo luogo, non si utilizzano le misure reali di potenza, che richiederebbero tra l'altro un'attesa di 30 minuti tra un'iterazione e la successiva (imposta dal main), ma delle misure "fittizie" ricavate in modo coerente con le matrici di transizione delle MC, come vedremo nel paragrafo seguente.

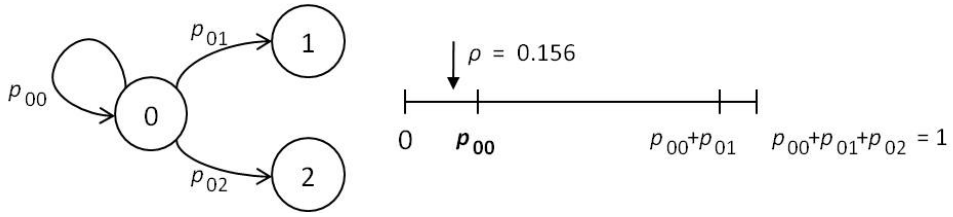


Figura 3.5: Criterio di scelta dello stato successivo durante l'evoluzione "forzata" della MC: il valore casuale estratto (ρ) appartiene all'intervallo [0, p_{00}], quindi il prossimo stato sarà ancora 0.

3.5.1 La funzione "markov_evol"

Nel modulo "markov_evol.c" viene definita l'omonima funzione, il cui scopo è quello di far evolvere le MC relative a $\{r\}$ e $\{e_R\}$, in modo da produrre una particolare istanza dei due processi aleatori nell'arco di una giornata. La funzione pertanto restituisce 48 valori di ciascun processo, uno per ogni slot della finestra di previsione, e li raccoglie all'interno del file "rich_rinn". Nel main, ad ogni iterazione del ciclo `while` principale, vengono utilizzati i valori dello slot associato all'ora di avvio dell'algoritmo DP.

Far evolvere una MC significa, in generale, a partire da uno stato fissato i, individuare gli stati di arrivo negli istanti di transizione successivi. Lo stato j raggiunto con la transizione dipende dalla distribuzione di probabilità associata allo stato di partenza, cioè dall'insieme

$$\{\, p_{i0},\, p_{i1},\, ...,\, p_{iM} \,\},$$

se il numero degli stati della MC è $M + 1$. Per semplicità di notazione non è stata riportata la dipendenza dall'istante di transizione.

Si ricorre alla funzione `srand48` della libreria standard C per generare il "seed" della sequenza pseudo-casuale restituita poi dalla funzione `drand48`. I valori di tale sequenza appartengono all'intervallo [0, 1] e sono utilizzati per selezionare lo stato di arrivo di una determinata transizione. In particolare, si suddivide idealmente l'intervallo [0, 1] in $M + 1$ sotto-intervalli: il primo è [0, p_{i0}], il secondo è [p_{i0}, $p_{i0} + p_{i1}$], il terzo è [$p_{i0} + p_{i1}$, $p_{i0} + p_{i1} + p_{i2}$], e così via. Indicando con ρ il valore preso dalla sequenza aleatoria, `markov_evol` seleziona j come stato di arrivo se e solo se

$$p_{i0} + p_{i1} + ... + p_{i(j-1)} \leq \rho \leq p_{i0} + p_{i1} + ... + p_{i(j-1)} + p_{ij}.$$

Vediamo un semplice esempio, illustrato in Figura 3.5. Supponiamo di avere una MC a tre stati, di essere nello stato 0 all'istante k e di voler determinare una possibile evoluzione nel

prossimo step $k+1$. Supponiamo che le probabilità di transizione siano $\{p_{00} = 0.2,\ p_{01} = 0.7,\ p_{02} = 0.1\}$. Risulta evidente che l'estensione dei sotto-intervalli rispecchia la facilità (e quindi la probabilità) con cui ρ può capitare al loro interno: il 70% dei valori pseudo-casuali generabili apparterrà al secondo sotto-intervallo, pertanto con una probabilità del 70% verrà effettivamente selezionato lo stato $j = 1$ (se ricorriamo alla definizione frequentistica di probabilità). Supponiamo tuttavia di avere all'istante corrente $\rho = 0.156$: si ha che $0 < \rho < 0.2$, quindi lo stato di arrivo sarà $j = 0$.

3.6 Complessità computazionale

Valutiamo ora la complessità computazionale del nostro algoritmo, ricorrendo ad un'analisi asintotica al caso pessimo. Per semplificare la notazione, all'interno dello pseudo-codice indicheremo i parametri di un ciclo riportando solo il numero di iterazioni da esso compiute:

for $(i = 0;\ i < N;\ i$++$)$ **do**		**for** (N) **do**
...	$\equiv$	...
end for		**end for**

Ci concentriamo innanzitutto sulla funzione `optimal_scheduler`, che implementa l'algoritmo DP. Non riportiamo tutte le varie operazioni elementari, che danno un contributo costante in termini di complessità e sono quindi trascurabili. Non riportiamo inoltre cicli con un numero di iterazioni pari a N (a meno che non siano annidati), cioè il numero delle appliance controllabili: si assume ragionevolmente che N sia sempre piuttosto basso (tipicamente non superiore a 4 o 5 nell'arco di 24 ore), quindi anche questi cicli sono trascurabili.

Indichiamo con R il numero di stati di $\{r\}$ e $\{e_R\}$, con W la durata della procedura DP (che al caso pessimo coincide appunto con la durata massima della finestra di previsione) e con M il massimo numero di stati che il controllo u_k può assumere. Ricordiamo che lo stato complessivo $(x_k,\ y_k)$ del nostro modello è determinato da $\{r_k\}$, $\{e_{R_k}\}$, u_{k-1} e $\underline{\mathbf{s}}_{ATT_k}$ (par. 2.5.1), pertanto, al caso peggiore, il numero totale di stati da considerare nel procedimento di minimizzazione del costo sarà sempre R^2M.

Lo pseudo-codice (semplificato) di `optimal_scheduler` è il seguente:

optimal_scheduler:

for (R^2M) **do**

$inizializza\ vettore\ costi$;

end for

for (R^2M) **do**

for (N) **do**

$inizializza\ s_{ATT'_W}$;

end for

end for

for (W) **do** // ciclo I

for (R^2M) **do**

$inizializza\ contatore\ deadline\ mancate$;

end for

for (W) **do**

$conta\ slot\ low\_cost$;

end for

for (R^2M) **do** // ciclo II

for (R) **do**

for (R) **do**

$calcola\ p_{ij}\ dei\ macrostati\ per\ ogni\ j$;

end for

end for

for (2^N) **do** // ciclo III

for (R^2) **do**

$calcola\ J(j)\ per\ ogni\ j$;

end for

for (R^2) **do**

$calcola\ costo\ atteso$;

end for

end for

end for

for (R^2M) **do**

for (N) **do**

$controlla\ se\ hai\ già\ completato\ tutte\ le\ attività$;

end for

end for

end for

Si osserva che il contributo più consistente, in termini di complessità, è dato dall'annidamento dei cicli denominati I, II e III. Il ciclo I determina lo spostamento a ritroso di uno slot alla volta; il ciclo II, all'inizio del generico slot k, permette di valutare tutti gli stati (x_k, y_k) possibili per il calcolo del costo atteso; il ciclo III consente di valutare tutte le azioni di controllo u_k, per individuare quella di costo minimo. Per quest'ultimo ciclo contiamo 2^N iterazioni (tutti i possibili valori del vettore binario **b**), con all'interno $R^2 + R^2$ operazioni, quindi in totale $2R^2 2^N$ operazioni.

Per il ciclo II contiamo R^2M iterazioni, con all'interno il contributo appena determinato, più R^2 operazioni, quindi:

$$R^2M(2R^2 2^N + R^2) = R^4M(2 \cdot 2^N + 1).$$

Per il ciclo I abbiamo W iterazioni, con all'interno il contributo calcolato sopra, più $R^2M + W$ operazioni:

$$W\Big[R^2M + W + R^4M(2 \cdot 2^N + 1)\Big].$$

Il termine introdotto dal ciclo II è approssimabile con $O\big(R^4M2^N\big)$, pertanto il termine $R^2M + W$ è trascurabile nell'analisi asintotica. La complessità dei tre cicli annidati risulta quindi $O\big(WR^4M2^N\big)$. A questo vanno aggiunte le R^2M e R^2MN iterazioni dei cicli iniziali della funzione, che però divengono anch'esse trascurabili. Possiamo allora concludere che la complessità di `optimal_scheduler` è $O\big(WR^4M2^N\big)$.

N	1	2	3	4	5	6	7	8	9	10
2^N	2	4	8	16	32	64	128	256	512	1024
N^2	1	4	9	16	25	36	49	64	81	100
N^3	1	8	27	64	125	216	343	512	729	1000

Tabella 3.2: Confronto tra le funzioni $2^N, N^2, N^3$ per $N \leq 10$

Generalmente, complessità asintotiche dipendenti da funzioni esponenziali sono indice di algoritmi molto poco efficienti. Tuttavia, dato che ci si aspetta un numero di appliance controllabili, come osservato in precedenza, tipicamente non superiore a 4 o 5, la funzione 2^N nel nostro caso non grava eccessivamente sulla complessità. Essa è anzi abbondantemente sovra-dominata dalla funzione cubica fino a $N = 9$, e l'andamento si avvicina molto a quello della funzione quadratica fino a $N = 5$, come si può osservare nella Tabella 3.2.

Analizziamo ora la complessità computazionale del main, riportandone lo pseudo-codice semplificato secondo le stesse considerazioni fatte per la funzione `optimal_scheduler` . Tralasciamo per il momento le istruzioni eseguite dalla funzione `power_calc` per il calcolo della potenza o da `markov_evol` per l'evoluzione delle MC (nel caso si utilizzi la versione per le simulazioni), che danno comunque un contributo trascurabile nella complessità totale, come vedremo in seguito.

Indichiamo con S la durata massima (in slot) di una qualunque attività schedulabile. Al caso pessimo tutte le N attività avranno ovviamente durata pari a S. Non teniamo conto del tempo di attesa forzato tra due chiamate successive a `optimal_scheduler` (30 minuti), nel caso si utilizzi la versione normale, che non riguarda l'efficienza dell'algoritmo.

Main:

for (R^2W) **do**

$carica\ matrici\ di\ transizione\ di\ r$;

end for

for (R^2W) **do**

$carica\ matrici\ di\ transizione\ di\ e_R$;

end for

for (R^2W) **do**

controlla correttezza matrici di r;

controlla correttezza matrici di e_R;

end for

for (N) **do**

for (N) **do**

ordina deadline;

end for

end for

for (N) **do**

for (S) **do**

inizializza a 0 *matrice dei profili energetici*;

end for

end for

for (N) **do**

for (S) **do**

riempi matrice dei profili energetici;

end for

end for

while do // ciclo IV

esegui `optimal_scheduler`;

for (N) **do**

for (S) **do**

aggiorna variabili;

end for

end for

end while

La parte iniziale del main comprende $3R^2W$ iterazioni per il caricamento e il controllo delle matrici di transizione dei processi $\{r\}$ e $\{e_R\}$. Seguono le istruzioni per ordinare le appliance

in base alle rispettive dead-line e riempire la matrice $N \times S$ che ne raccoglie i profili energetici, aggiungendo così $N^2 + 2NS$ iterazioni.

Il ciclo IV comporta, nel caso peggiore, W esecuzioni di `optimal_scheduler` e aggiornamenti vari (trascurabili rispetto al contributo di queste ultime). Si deve però osservare che, ad ogni esecuzione, la finestra su cui agisce l'algoritmo DP diminuisce di uno slot, quindi in realtà la complessità di `optimal_scheduler` non è fissa: se indichiamo con I il numero delle iterazioni compiute prima della sua chiamata, essa è pari a $O\big((W-I)R^4M2^N\big)$. Ciò significa che, alla prima iterazione del `while` l'algoritmo DP richiederà un tempo $O\big(W^2R^4M2^N\big)$, mentre diverrà sempre più veloce nelle iterazioni successive, finché all'ultima richiederà un tempo W volte inferiore, cioè $O\big(WR^4M2^N\big)$.

La complessità introdotta dalla parte iniziale diventa ovviamente trascurabile, pertanto la complessità totale del main, dopo un numero generico I di iterazioni, risulta $O\big(W(W-I)R^4M2^N\big)$.

Analizziamo molto rapidamente le complessità di `power_calc` e `markov_evol`, dimostrando che esse sono effettivamente trascurabili.

In `power_calc` (par.3.3.2), ciò che "appesantisce" la complessità è il calcolo della DFFT per i segnali di tensione e corrente. Se indichiamo con F il numero di campioni in cui è discretizzato il dominio della Trasformata (che deve essere una potenza di 2), la complessità di tale calcolo è $O\big(F\log_2 F\big)$ [10]. Attualmente si utilizza un numero di campioni pari a $F = 512$, pertanto $O\big(F\log_2 F\big) < O\big(W(W-I)R^4M2^N\big)$.

Per quanto riguarda `markov_evol`, il contributo maggiore nel tempo di esecuzione è dato dal calcolo dei sotto-intervalli in $[0,1]$ (par.3.5.1) mediante una somma di probabilità che richiede ogni volta un numero di accessi a una determinata riga della matrice di transizione pari a:

$$\sum_{i=1}^{R} i = \frac{R(R+1)}{2} = \frac{R^2}{2} + \frac{R}{2}\,.$$

Questo avviene sia per il processo $\{r\}$ sia per $\{e_R\}$, e viene ripetuto per tutti i W slot della finestra di previsione. Il numero di iterazioni risulta quindi $W2R^2$, da cui la complessità $O(WR^2)$.

3.7 Risultati delle simulazioni

Sono state effettuate numerose simulazioni per verificare i risultati prodotti dal nostro algoritmo, sia in termini di efficacia (osservando la bontà degli *schedule* proposti) che di efficienza (misurando il tempo necessario per arrivare alla soluzione).

Il numero massimo di appliance controllabili che abbiamo dato in ingresso all'algoritmo è stato $N = 10$. Si tratta di un valore piuttosto elevato rispetto alle aspettative di consumo dell'utenza media, con cui sono stati comunque riscontrati ottimi risultati.

Riportiamo nella trattazione i risultati di due simulazioni, in cui vengono impostate dall'utente le stesse appliance, ma con dead-line differenti e a partire da orari diversi. All'inizio dell'esecuzione è possibile decidere se usufruire o meno delle risorse rinnovabili. In caso negativo, nella modalità di esecuzione che indicheremo con "N", alcuni parametri saranno inizializzati in modo tale da abilitare una politica di controllo basata esclusivamente sulla valutazione delle fasce di costo energetico. La modalità alternativa sarà indicata con "R". È interessante confrontare gli schedule così prodotti con quelli pianificati invece tenendo conto dell'energia rinnovabile.

Mediante la funzione `markov_evol` è stata creata un'ipotetica evoluzione del profilo di consumo $\{r\}$ e dell'energia rinnovabile disponibile $\{e_R\}$. Le stesse evoluzioni sono adottate in tutte e quattro le esecuzioni dell'algoritmo, in modo da analizzare i cambiamenti delle politiche di controllo solo al variare della finestra temporale e della scelta di utilizzo o meno delle fonti rinnovabili, a parità di scenario.

Si suppone che l'energia rinnovabile sia interamente prodotta nella Microgrid da pannelli solari. Le relative MC sono state definite tenendo conto di tale scenario, inserendo nelle matrici di transizione delle probabilità tali per cui si ottengono mediamente livelli alti di energia negli orari corrispondenti alla tarda mattinata e al pomeriggio, mentre si hanno livelli nulli verso sera e durante la notte.

Per la determinazione dei profili energetici delle varie appliance abbiamo fatto riferimento a caratteristiche di elettrodomestici reali (di Classe Energetica A). Ne abbiamo misurato la potenza istantanea assorbita tramite la lettura dei contatori digitali tradizionali e abbiamo integrato tali informazioni con i dati offerti dalle Case Produttrici e da *ENEA* ("Ente nazionale per le Nuove tecnologie, l'Energia e l'Ambiente") [11].

3.7.1 Simulazione A

La Tabella 3.3 mostra le appliance controllabili inserite in ingresso dall'utente, indicando per ciascuna il numero associato (e dunque la priorità), il tipo, la dead-line, la durata in numero di slot, il permesso di interrompibilità ("0" indica "non interrompibile"), il picco assorbito nei primi minuti e il profilo energetico. Per comodità sono elencate già secondo l'ordine con cui l'algoritmo le dispone nella fase iniziale di caricamento. I livelli di potenza assorbita sono specificati dagli stati corrispondenti, in base a quanto visto al paragrafo 3.2.

i	**Tipo**	$\underline{\mathbf{D}}(i)$	$\underline{\mathbf{s}}_{ATT}(i)$	$\underline{\mathbf{I}}_{ATT}(i)$	$\underline{\mathbf{P}}_{ATT}(i)$	$\underline{\mathbf{e}}_i$
0	Forno	19:30	2	0	7	$\{3, 3\}$
1	Microonde	20:00	2	0	4	$\{1, 1\}$
2	Asciugatrice	23:30	5	1	7	$\{4, 3, 2, 3, 2\}$
3	Lavastoviglie	6:30	5	1	7	$\{2, 1, 1, 1, 2\}$
4	Lavatrice	9:00	4	1	6	$\{2, 1, 1, 1\}$

Tabella 3.3: Definizione delle appliance da inserire nello scheduling (Simulazione A).

La durata di ogni slot è di 30 minuti. L'inizio della pianificazione è fissato alle ore 17:30, e le scadenze delle ultime due appliance sono da intendersi ovviamente come relative al giorno successivo, pertanto la finestra effettiva in cui opera l'algoritmo è pari a $W' = 31$ (slot).

La Figura 3.6 illustra lo schedule prodotto senza considerare le eventuali risorse rinnovabili a disposizione (modalità N). Le unità colorate corrispondono all'energia assorbita dalle varie appliance. Si utilizzano cinque stili diversi in modo da distinguere le attività (per maggior chiarezza sono riportati fra parentesi i numeri corrispondenti). Il profilo bianco contornato dalla linea tratteggiata corrisponde alla richiesta non controllabile dell'utente (il processo aleatorio $\{r\}$). Nella parte superiore della Figura è indicato il limite superiore r_{MAX}, mentre nella parte inferiore sono indicate le scadenze di ciascuna appliance ($d_0, d_1, ..., d_4$).

Si può notare innanzitutto che il vincolo (2.7) sulla potenza massima assorbita è rispettato in ogni slot dello schedule, e che tutte le appliance vengono terminate entro la dead-line stabilita. L'ultima attività inoltre ("Lavatrice") viene terminata con un'ora di anticipo, per evitare di allocare energia dopo le ore 8, ovvero nella fascia di costo più elevato. Il costo complessivo del piano è pari a 1.28 Euro.

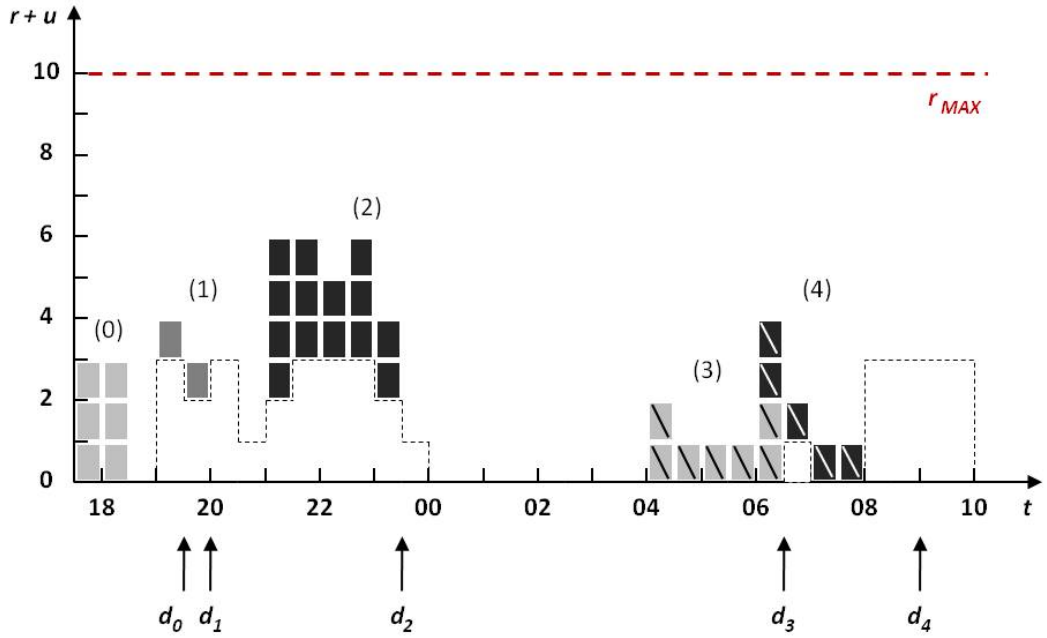


Figura 3.6: Pianificazione delle appliance nella simulazione A senza utilizzo delle risorse rinnovabili.

Va ricordato che l'algoritmo termina o in caso di errore o quando tutte le appliance sono state inserite nello schedule, pertanto, come in questo esempio, non è detto che l'istante di terminazione coincida con lo stage che precede l'ultima dead-line. Va ricordato inoltre che il costo di ogni slot k dipende sia dall'energia controllata u_k sia dall'energia non controllabile r_k. In virtù di queste due osservazioni, abbiamo deciso di considerare sempre il costo totale delle simulazioni fino all'ultima scadenza fissata, indipendentemente dalla durata effettiva, per rendere confrontabili i vari costi. Dove non specificato altrimenti, si intenderà che il costo fino alla terminazione dell'algoritmo e quello fino all'ultima dead-line coincidono. Nel caso della Simulazione A senza energia rinnovabile, il costo finale diventa così pari a 1.41 Euro.

La Figura 3.7 (a) illustra invece lo schedule prodotto usufruendo dell'energia rinnovabile (modalità R). Si osserva che, rispetto al caso precedente, l'algoritmo tende a spostare l'allocazione di energia verso le fasce in cui è prevista effettivamente una maggiore disponibilità di risorse rinnovabili. La finestra in cui si svolge la pianificazione tuttavia è centrata prevalentemente nelle ore serali e notturne, pertanto non è molto evidente la convenienza, in termini di costo monetario, rispetto alla precedente esecuzione. Il costo complessivo del piano è infatti pari a 1.05 Euro. La Figura 3.7 (b) tiene conto esclusivamente dell'energia non rinnovabile, comprata dalla società distributrice, sottraendo dallo schedule il consumo a costo zero. In altre parole,

essa mostra solo le unità energetiche che contribuiscono alla spesa dell'utente.

Si verifica infine facilmente che viene mantenuta la condizione di "non-interrompibilità" di "Forno" e "Microonde" in entrambi i casi.

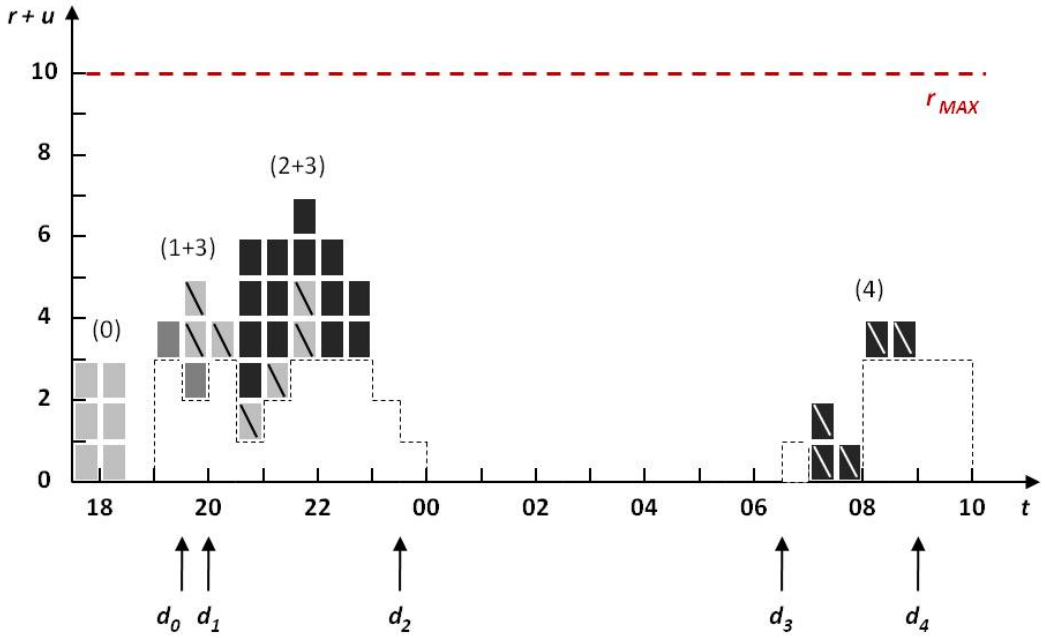
Abbiamo successivamente avviato le due modalità di simulazione a partire dalle ore 9:00, mantenendo le stesse dead-line. In tal modo ci siamo ricondotti al caso peggiore dal punto di vista della finestra su cui opera l'algoritmo ($W = W' = 48$ slot, ovvero 24 ore). Lo schedule ottenuto nella modalità N all'interno della finestra estesa è sostanzialmente identico a quello calcolato all'interno della finestra originaria ($W' = 31$). La sola differenza riguarda l'allocazione di "Forno" nelle prime due iterazioni, ovvero dalle ore 9 alle ore 10. Lo schedule ottenuto nella modalità R invece presenta invece come differenza principale, con l'estensione della finestra, il fatto che alcune attività vengano inserite nelle ore del pomeriggio, in cui si prevedono elevati livelli di potenza prodotta dai pannelli solari della Microgrid.

Per avere un riscontro con quanto visto teoricamente a proposito della complessità computazionale (par. 3.6), abbiamo misurato, in entrambe le modalità, il tempo totale di esecuzione e i tempi parziali tra l'inizio di una chiamata alla funzione `optimal_scheduler` e la sua terminazione. L'algoritmo è stato eseguito su un calcolatore dotato di processore da 1.5 GHz e memoria RAM da 1.5 GB. Per una stima più accurata sono stati calcolati i tempi medi valutati su 10 esecuzioni dello stesso tipo.

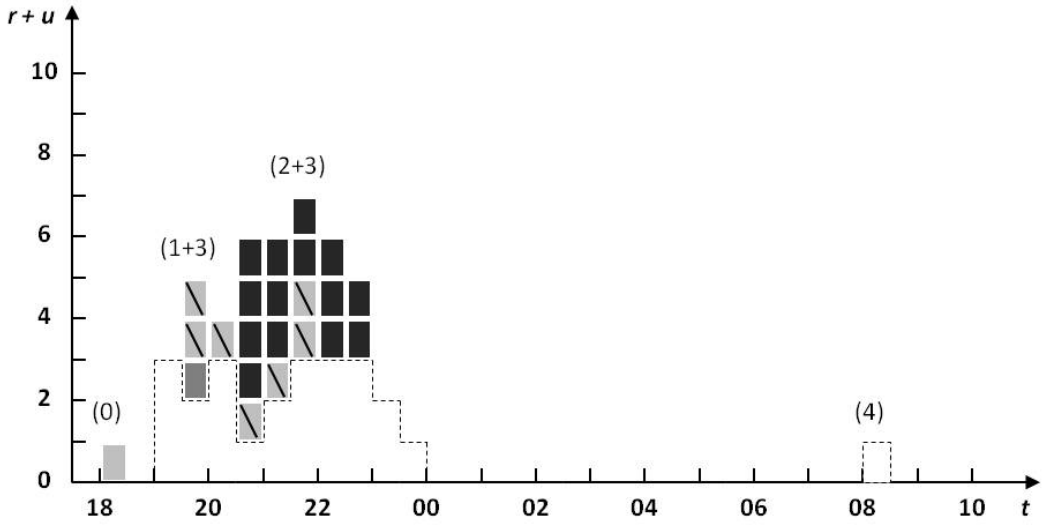
La fase di caricamento iniziale delle matrici di transizione e di ordinamento delle attività richiede circa 1 ms. Il tempo di esecuzione totale è sostanzialmente dato dalla somma dei tempi parziali, dato che le operazioni intermedie tra una chiamata e la successiva di `optimal_scheduler` danno un contributo trascurabile. In realtà il tempo totale non è molto significativo, poiché nell'applicazione pratica le iterazioni dell'Algoritmo DP avvengono a distanza di 30 minuti l'una dall'altra.

Il grafico di Figura 3.8 consente invece di verificare la riduzione dei tempi (medi) parziali dopo ogni chiamata, dovuta principalmente al restringimento della finestra W' e alla diminuzione delle attività da schedulare man mano che vengono inserite nel piano. È possibile notare una riduzione quasi lineare del tempo negli intervalli in cui non viene allocata energia, alternata da cali improvvisi, in corrispondenza degli istanti in cui l'algoritmo inserisce invece una o più appliance.

Una ragionevole misura dell'efficienza del nostro algoritmo può essere definita come il



(a)



(b)

Figura 3.7: Pianificazione delle appliance nella simulazione A con utilizzo delle risorse rinnovabili (a). L'energia pagata alla società distributrice è di fatto quella evidenziata in (b).

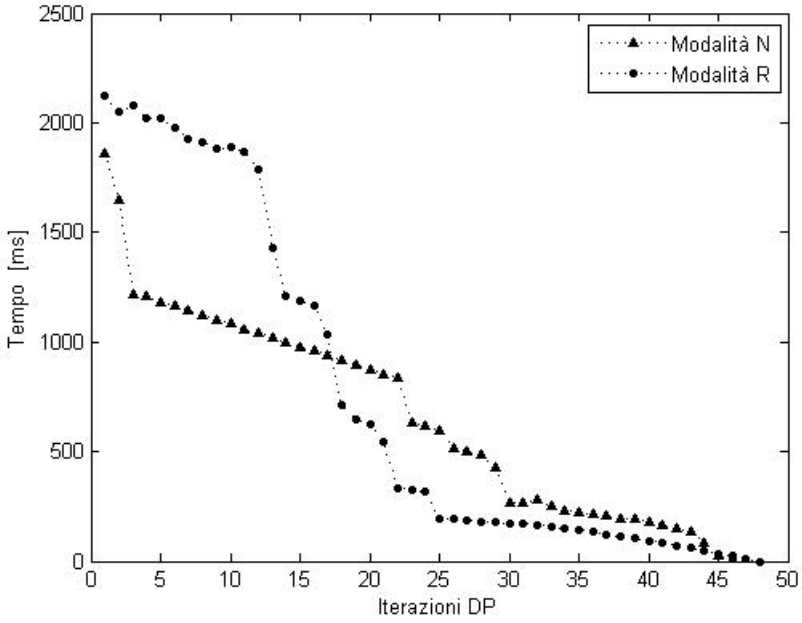


Figura 3.8: Simulazione A. Tempo di esecuzione dell'Algoritmo DP (optimal_scheduler) al crescere del numero di iterazioni effettuate.

rapporto tra il massimo tempo parziale t^* di un'esecuzione (che in generale è quello relativo alla prima chiamata a `optimal_scheduler`) e il tempo a disposizione per pianificare l'allocazione energetica in uno slot (cioè la durata dello slot stesso, pari a 30 minuti). Indichiamo tale rapporto con Γ: il caso $\Gamma > 1$ è chiaramente improponibile, perché significherebbe che lo scheduler non è in grado di individuare la soluzione per lo slot corrente prima di iniziare la pianificazione nello slot successivo. L'algoritmo è invece tanto più efficiente quanto più piccolo è Γ rispetto all'unità. Con l'implementazione attuale si ottiene sempre, anche al caso peggiore, $\Gamma << 1$, ma è un parametro che può risultare utile qualora si voglia cercare di ridurre al minimo la durata degli slot senza andare al di sotto di un determinato valore di efficienza.

Per la Simulazione A, in modalità N si ha $t^*_N = 1.862$ s , da cui $\Gamma_N \approx 0.00103$, in modalità R $t^*_R = 2.122$ s, pertanto $\Gamma_R \approx 0.00118$.

3.7.2 Simulazione B

Nella seconda simulazione viene mantenuto lo stesso insieme di appliance controllabili, ma le dead-line sono fissate tutte alle ore 00:00, mentre l'avvio dello scheduling è impostato alle ore 9:00 (del giorno precedente). La finestra, di durata $W' = 30$, risulta così centrata

prevalentemente in pieno giorno. Questo ci consente di analizzare l'efficacia dell'algoritmo su un intervallo di tempo completamente diverso rispetto a quello della Simulazione A, e soprattutto di apprezzare meglio le due diverse "strategie" nel caso si ricorra o meno all'energia rinnovabile.

L'impostazione di una stessa scadenza per tutte le appliance da un lato rilassa i vincoli temporali a cui è soggetto lo scheduling, dall'altro però costringe il controllore a gestire l'allocazione energetica di ciascuna attività sull'intera finestra anziché su varie "sotto-finestre".

La Tabella 3.4 riporta le appliance secondo l'ordine definito dall'algoritmo nella fase iniziale.

i	**Tipo**	$\underline{\mathbf{D}}(i)$	$\underline{\mathbf{s}}_{ATT}(i)$	$\underline{\mathbf{I}}_{ATT}(i)$	$\underline{\mathbf{P}}_{ATT}(i)$	$\underline{\mathbf{e}}_i$
0	Asciugatrice	00:00	5	1	7	$\{4, 3, 2, 3, 2\}$
1	Microonde	00:00	2	0	4	$\{1, 1\}$
2	Forno	00:00	2	0	7	$\{3, 3\}$
3	Lavastoviglie	00:00	5	1	7	$\{2, 1, 1, 1, 2\}$
4	Lavatrice	00:00	4	1	6	$\{2, 1, 1, 1\}$

Tabella 3.4: Definizione delle appliance da inserire nello scheduling (Simulazione B).

In Figura 3.9 si riporta lo schedule prodotto ignorando ancora una volta le fonti rinnovabili (modalità N). La rappresentazione delle unità allocate per ogni appliance è la stessa della Simulazione A. Possiamo notare che l'algoritmo, per minimizzare il costo totale, deve concentrare tutte le attività controllabili nella fascia "low-cost", cioè dalle ore 19. Il costo finale è di 1.79 Euro.

La Figura 3.10 (a) mostra il risultato dell'esecuzione in cui è previsto l'utilizzo di energia rinnovabile (modalità R). A differenza del primo caso, buona parte delle attività viene inserita nelle ore del pomeriggio. Nella Figura 3.10 (b) si osserva infatti che tale strategia riduce notevolmente il numero di unità energetiche che gravano sulla spesa complessiva. Il costo totale fino alla terminazione dell'algoritmo è pari a 0.42 Euro, mentre il costo fino alla scadenza fissata è di 0.73 Euro, inferiore alla metà del costo ottenuto senza sfruttare le fonti rinnovabili.

In entrambe le modalità si può verificare che gli schedule non portano al superamento del limite r_{MAX} e rispettano la "non-interrompibilità" di "Forno" e "Microonde".

Anche per la Simulazione B sono stati misurati i tempi di esecuzione totali e parziali nelle modalità N e R, estendendo la finestra alla massima durata possibile $W = 48$. In quest'ultimo

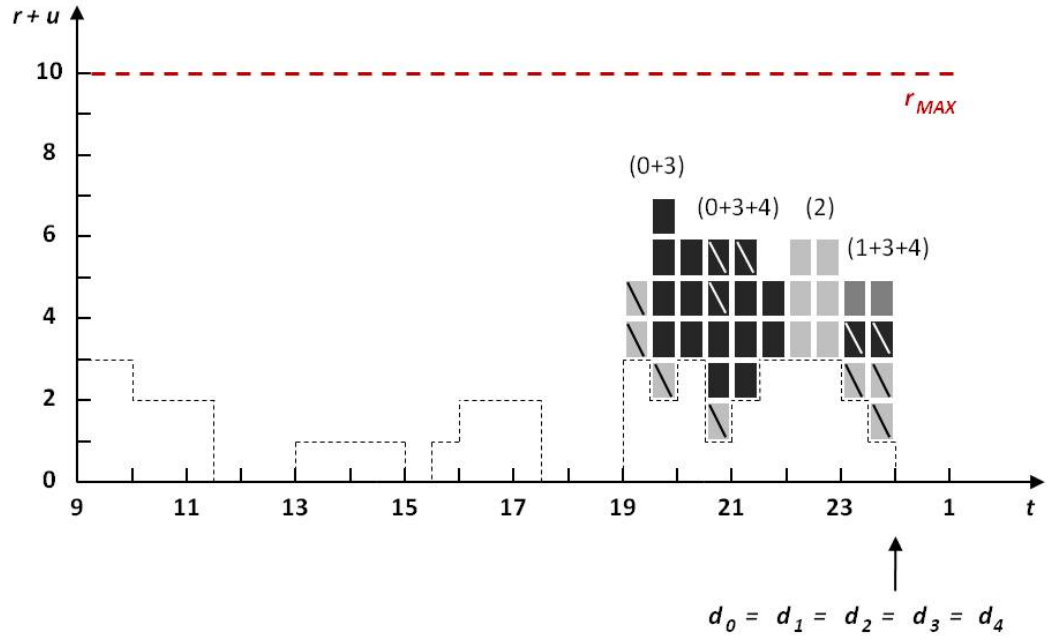
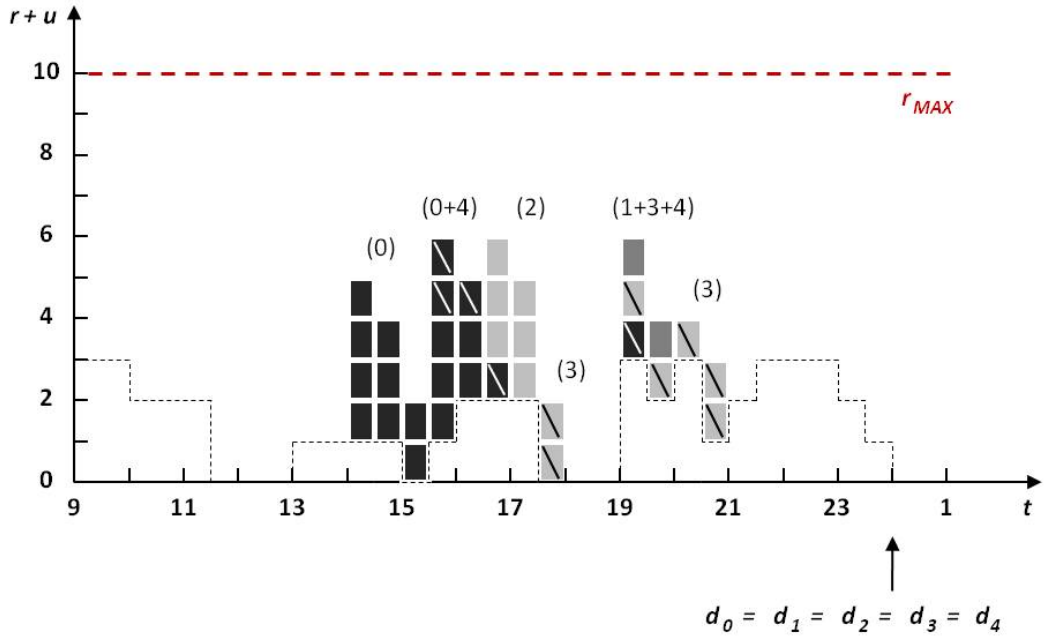


Figura 3.9: Pianificazione delle appliance nella simulazione B senza utilizzo delle risorse rinnovabili.

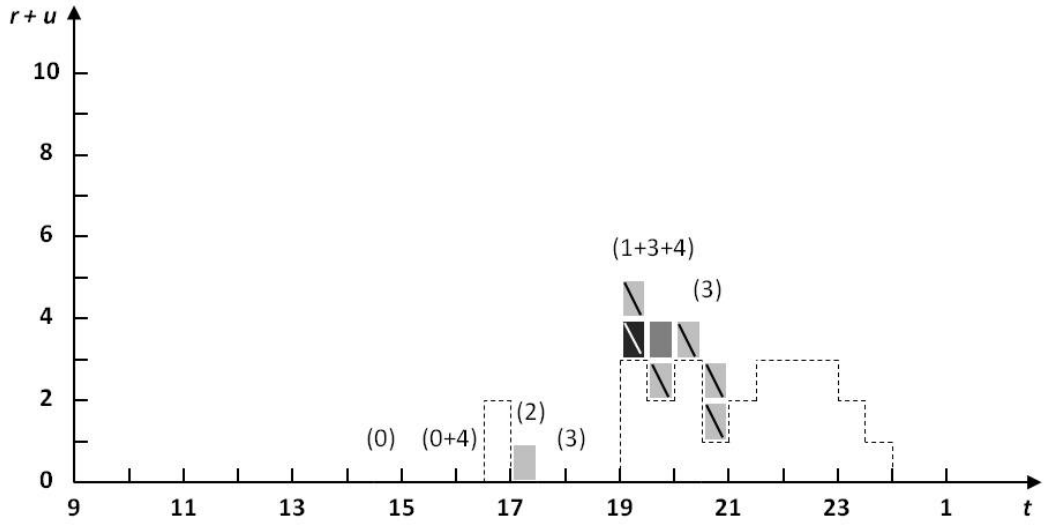
caso l'istante di avvio corrisponde pertanto alle ore 00:00. Gli schedule ottenuti con la nuova finestra sono identici a quelli illustrati sopra con la finestra originaria. L'andamento del tempo in funzione del numero di iterazioni effettuate è riportato nel grafico di Figura 3.11.

È possibile osservare, come per la Simulazione A, una riduzione quasi lineare del tempo durante il periodo in cui l'algoritmo non alloca energia. Nella modalità N tale andamento si protrae per quasi tutta la durata della finestra, poiché le appliance vengono inserite nello schedule solo negli slot finali.

Nella modalità N il massimo tempo parziale è pari a $t_N^* = 1.468$ s, e l'efficienza risulta di conseguenza $\Gamma_N \approx 0.00081$, mentre nella modalità R abbiamo $t_R^* = 2.208$ s, da cui $\Gamma_R \approx 0.00123$.



(a)



(b)

Figura 3.10: Pianificazione delle appliance nella simulazione B con utilizzo delle risorse rinnovabili (a). L'energia pagata alla società distributrice è di fatto quella evidenziata in (b).

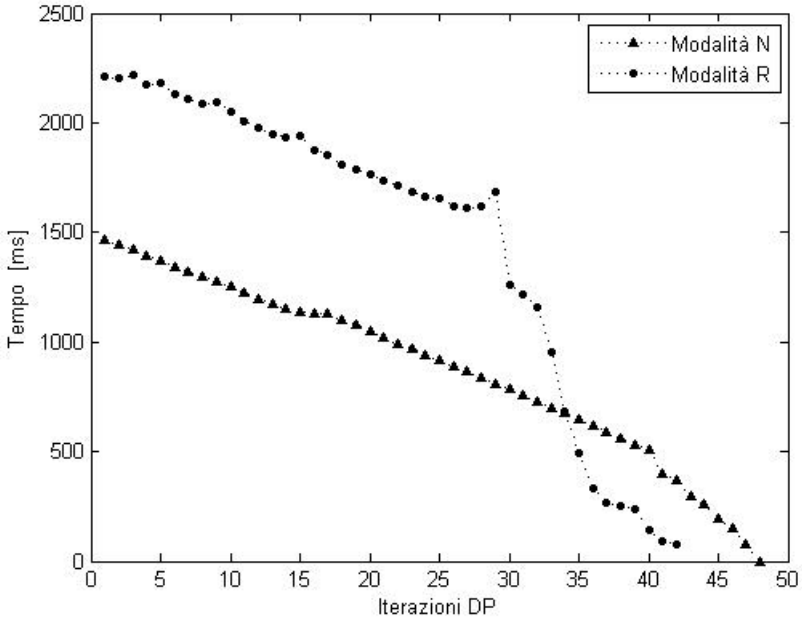


Figura 3.11: Simulazione B. Tempo di esecuzione dell'Algoritmo DP (optimal_scheduler) al crescere del numero di iterazioni effettuate.

Capitolo 4

Sviluppo del Radio Smart Meter

4.1 Schema generale

Lo Smart Meter progettato nella Tesi si compone essenzialmente di tre elementi:

- *Circuito Integrato MCP3909 (Microchip Technology)* [12], per la misura in tempo reale di tensione e corrente elettrica;
- *Scheda Programmabile LPC1768-H (NGX Technologies)* [13] [14], per la raccolta e la gestione dei dati;
- *Radio AT86RF231 (Atmel)* [15], per la trasmissione dei dati al server, che li utilizza per elaborazioni successive e per l'esecuzione dell'algoritmo di scheduling.

Lo schema generale di funzionamento è riportato in Figura 4.1. Il dispositivo MCP3909 rileva separatamente i segnali analogici di tensione e corrente provenienti dalla linea elettrica, li converte nel dominio digitale, li codifica in due parole da 16 bit ciascuna e li trasmette alla Board LPC1768-H mediante il protocollo standard di comunicazione seriale *SPI* (*Serial Peripheral Interface*), che verrà esaminato brevemente nel paragrafo successivo. Alla ricezione di tutti i 32 bit associati allo stesso rilevamento, essi vengono inseriti in un *buffer* apposito. Raggiunta la capienza massima, tutti i dati sono inviati dalla Board alla Radio mediante comunicazione *SPI* su un'altra linea. Da qui sono infine trasmessi al server usando il protocollo standard IEEE 802.15.4, definito per reti wireless di piccole dimensioni (*WPAN, Wireless Personal Area Network*).

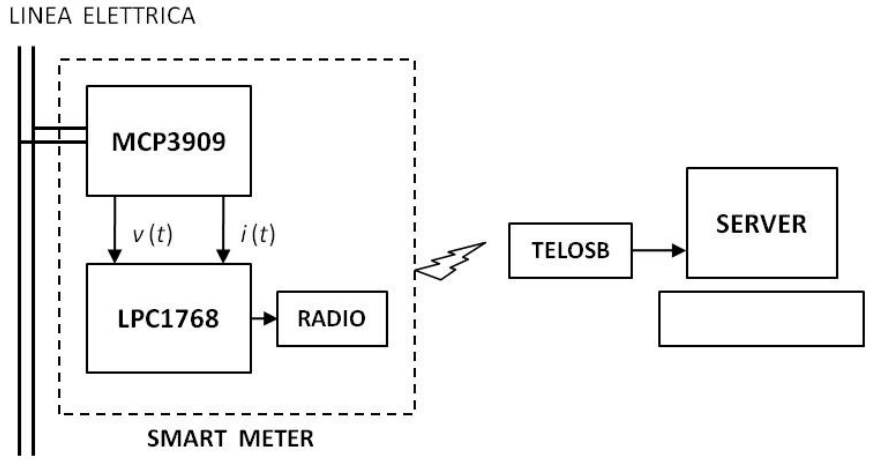


Figura 4.1: Schema generale di funzionamento dello Smart Meter e comunicazione wireless con il lato server.

Il server utilizza come ricevitore il nodo sensore *TelosB* (*Crossbow Technology*) [16], dotato di sistema operativo *Tiny OS*. Il sensore è in grado di leggere i vari pacchetti e di trasmetterne il contenuto alla porta seriale USB, codificato in formato esadecimale. Il payload è così scritto sul file "volt_curr", a cui accede periodicamente in lettura la funzione `power_calc` per determinare la potenza attiva, come visto nel paragrafo 3.1.

4.2 Serial Peripheral Interface (SPI)

L'interfaccia SPI è un sistema di comunicazione seriale tra un microcontrollore (*MCU*) e altri circuiti integrati o tra microcontrollori differenti. Si tratta di un sistema *full duplex* ad alta velocità, in cui uno dei due dispositivi svolge il ruolo di "*Master*" e ha il compito di gestire la sincronizzazione delle trasmissioni verso il secondo dispositivo, che svolge invece il ruolo di "*Slave*". Il bus di comunicazione è costituito da quattro segnali, denominati:

- *Clock* (*CLK*): segnale generato dal Master e utilizzato per regolare i tempi di lettura e scrittura dei singoli bit;
- *Master Output-Slave Input* (*MOSI*): segnale di uscita del dispositivo "Master", noto anche come *Serial Data In* (*SDI*);
- *Master Input-Slave Output* (*MISO*): segnale di ingresso del "Master", noto anche come *Serial Data Out* (*SDO*);

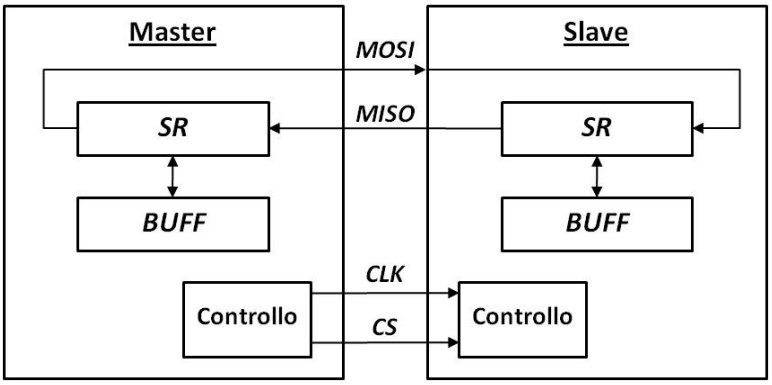


Figura 4.2: I quattro segnali su cui si basa il funzionamento dell'interfaccia SPI.

- *Chip Select* (*CS* o *CSEL*): rappresenta una sorta di "interruttore". Quando il segnale viene abbassato allo "0" logico, lo "Slave" si sincronizza col "Master" e può iniziare la comunicazione; quando viene alzato ("1" logico) si sospende la comunicazione. Esso è noto anche come *Slave Select* (*SS* o *SSEL*).

Una volta configurato il clock, il Master abbassa il CS associato allo Slave desiderato. Durante la comunicazione vengono utilizzati due Registri a Scorrimento (*Shift Register, SR*) per la ricezione e la trasmissione dei vari bit della sequenza: il Master invia sulla linea MOSI un bit, inserito dallo Slave nel proprio SR, determinando così lo scorrimento che a sua volta comporta l'uscita di un bit e il suo invio sulla linea MISO. Il bit provoca uno scorrimento dello SR del Master e così via (vedi Figura 4.2). Quando i due dispositivi si sono scambiati un intero byte, esso viene copiato dallo SR al *Buffer Seriale* (*BUFF*), per essere utilizzato dal software in esecuzione.

In pratica, si stabilisce un ciclo di trasferimento dati tra i due SR, che termina quando il Master alza il CS. A questo punto verrà ignorato qualsiasi segnale inviato sulle linee di Input/Output. È importante osservare che, senza la trasmissione di un dispositivo, non può avvenire la ricezione dell'altro. Quando il Master ha bisogno pertanto esclusivamente di ricevere dati utili, come nel nostro caso, deve comunque inviare dei dati "fittizi"(ad esempio la sequenza nulla) per far scorrere lo SR dello Slave.

La configurazione del CLK iniziale non coincide solo con la scelta della frequenza, ma anche delle cosiddette "polarità" (*CPOL*) e "fase" (*CPHA*). La polarità determina il livello logico (alto o basso) in cui il clock è considerato "attivo", mentre la fase indica su quali fronti (di salita o discesa) avvengono la scrittura e la lettura dei dati sul bus. Secondo la convenzione,

la modalità con CPOL $= 0$ e CPHA $= 0$, ad esempio, significa che il clock è attivo quando si trova a "1" (e di conseguenza "inattivo" a "0"), mentre i dati vengono inviati sul MOSI al secondo fronte di attività del clock (cioè quello di discesa) e ricevuti sul MISO al primo fronte (quello di salita).

4.3 Circuito Integrato MCP3909

4.3.1 Funzionamento generale

Il misuratore di energia elettrica MCP3909 è in grado di generare simultaneamente in uscita due funzioni: un'onda quadra con frequenza proporzionale al valore della potenza attiva rilevata, e un'onda sinusoidale corrispondente all'andamento della potenza istantanea, i cui campioni possono essere recuperati tramite l'interfaccia SPI. Utilizzando la medesima interfaccia, è possibile inoltre ottenere separatamente i valori delle sinusoidi di tensione e corrente in ingresso al circuito, come abbiamo scelto per la realizzazione del nostro Smart Meter, in modo da recuperare una maggiore quantità di informazione utile. La tensione di alimentazione deve essere compresa tra 4 e 5 Volt.

Analizziamo brevemente lo schema di funzionamento del dispositivo, illustrato in Figura 4.3. I segnali di corrente e tensione sono ricevuti in continuazione su due canali differenziali separati, rispettivamente *Canale 0* (*CH0*) e *Canale 1* (*CH1*). Il Canale 0 comprende un amplificatore a guadagno programmabile (*PGA*) per favorire la misurazione di segnali con ampiezza poco elevata, che nel nostro caso non viene utilizzato (guadagno unitario). I due segnali sono passati a due convertitori analogico-digitali di tipo "delta-sigma" ($\Delta - \Sigma\ ADC$), ciascuno con una risoluzione di 16 bit, e successivamente a due filtri passa-alto che eliminano la componente continua di offset introdotta dal circuito (stadio I). Un moltiplicatore calcola il prodotto dei due canali filtrati per determinare la potenza istantanea (stadio II). Il segnale viene quindi filtrato da un passa-basso di tipo IIR del primo ordine, in modo da estrarne la componente reale (cioè la potenza attiva), passata infine a un convertitore tensione-frequenza (*Digital-To-Frequency, DTF*) per il calcolo della frequenza proporzionale, indicata in Figura con HF_{OUT}, e la generazione dell'onda quadra alla medesima frequenza (stadio III). Il convertitore DTF accumula i campioni di potenza attiva finché l'energia misurata raggiunge un determinato valore di soglia e viene generato in risposta un impulso rettangolare. Il convertitore calcola altre

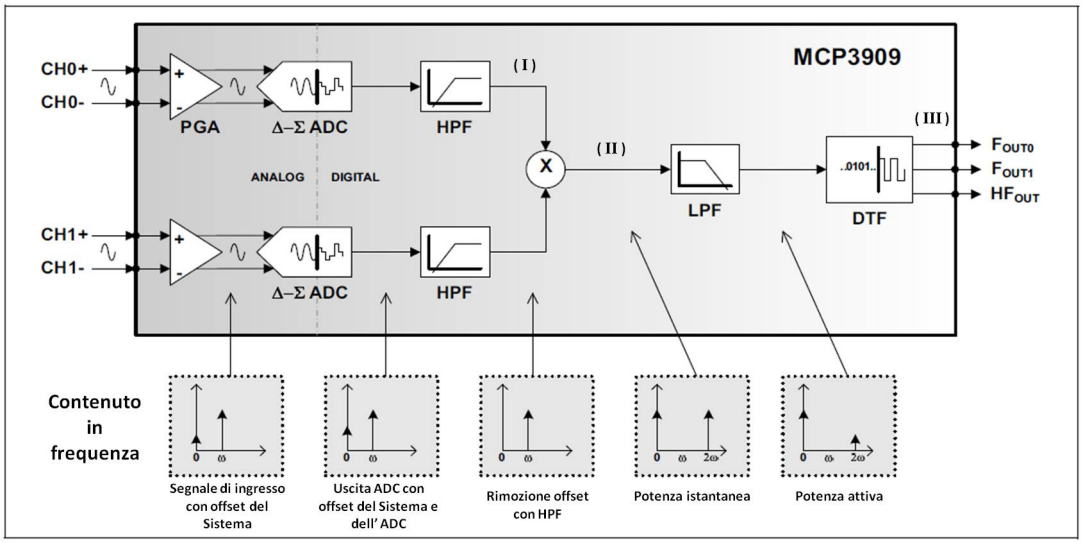


Figura 4.3: Elaborazione dei segnali di tensione e corrente in ingresso al circuito

due frequenze, F_{OUT0} e F_{OUT1}, utilizzabili per altri scopi, adottando una soglia diversa dalla precedente per la generazione degli impulsi corrispondenti.

4.3.2 Descrizione e funzionamento dell'interfaccia SPI

Il dispositivo MCP3909 consente di utilizzare tre diverse modalità di comunicazione SPI, cambiando la funzionalità dei pin coinvolti:

- *Multiplier Output*: permette di recuperare l'uscita del moltiplicatore, ovvero la potenza istantanea (stadio II). I dati vengono codificati in 20 bit (19 bit più segno) in complemento a due;
- *Dual Channel Output*: permette di recuperare l'informazione di ciascun canale (corrente e tensione) separatamente, in uscita dai convertitori ADC. I dati vengono codificati in 16 bit per ciascun canale (15 bit più segno) in complemento a due. È possibile acquisire i campioni prima o dopo il filtraggio di HPF;
- *Filter Input*: permette di passare un qualunque segnale di input direttamente al filtro LPF.

Per specificare la modalità SPI che si desidera utilizzare, è necessario trasmettere al dispositivo un opportuno byte di comando dopo la sua accensione (o un reset), all'interno di una finestra temporale ben precisa. Se indichiamo con $MCLK$ il clock del dispositivo, pari a circa 3.58

MHz, la sequenza deve essere trasmessa dopo un tempo $t_{WINSET} = 1/MCLK \approx 279.33$ ns ed entro un tempo $t_{WINDOW} = 32/MCLK \approx 8.94$ μs.

Il reset del dispositivo, con conseguente cancellazione di tutti i registri, può essere eseguito portando il pin *MCLR* ("*Master Clear*") allo "0" logico. È invece essenziale che esso sia mantenuto a "1" durante l'intero periodo di funzionamento.

Nel nostro caso è stata scelta la modalità "Dual Channel Output" in post-filtraggio (stadio I). Il dispositivo svolge il ruolo di Slave, controllato dalla scheda LPC1768-H. I dati sono raccolti in un'unica sequenza di 32 bit, dove i 16 bit più significativi rappresentano il Canale 1, mentre i restanti 16 bit il Canale 0. Ogni 256 cicli del clock interno (cioè alla frequenza di $MCLK/256 \approx 14$ kHz) viene inviato, sul pin associato al MISO, un bit indicato come DR (*Data Ready*), per segnalare la disponibilità di una nuova uscita prodotta dall'ADC. Questo definisce pertanto la frequenza di campionamento dei segnali di corrente e tensione in ingresso, pari a 14 kHz appunto, e di conseguenza il periodo di campionamento, pari a circa 71,5 μs.

Si osserva sperimentalmente tuttavia che, a causa delle operazioni svolte dalla scheda LPC1768-H per la corretta gestione dei dati, la distanza temporale tra un campione salvato e il successivo è maggiore. In particolare, ciascun periodo delle sinusodi di tensione e corrente risulta descritto da $N_T = 21$ campioni. La frequenza delle sinusoidi è pari a 50 Hz, da cui deriva un periodo di $T = 20$ ms. La frequenza di campionamento effettiva è quindi pari a $N_T/T = 1.05$ kHz.

Indichiamo con $CH0_+$ e $CH0_-$ gli estremi della tensione differenziale in ingresso al canale di corrente, con $CH1_+$ e $CH1_-$ gli estremi in ingresso al canale di tensione, e con $V_{REF} = 2.4$ V la tensione di riferimento interna del circuito. Indichiamo inoltre con $CH_0^{(16)}$ e $CH_1^{(16)}$ rispettivamente i 16 bit dei canali *CH0* e *CH1*. È possibile ricavare le tensioni misurate ai loro ingressi dalle seguenti equazioni, riportate nel *data-sheet* [12]:

$$CH0_+ - CH0_- = \frac{CH_0^{(16)} \cdot V_{REF}}{32768 \cdot \sqrt{8.06 \cdot \frac{0.66}{0.47}}},$$

$$CH1_+ - CH1_- = \frac{CH_1^{(16)} \cdot V_{REF}}{32768 \cdot \sqrt{8.06 \cdot \frac{0.47}{0.66}}},$$

dove $CH0_- = CH1_- = 0$ V, mentre i valori 0.47 e 0.66 sono legati alla massima tensione differenziale misurabile (prima di entrare in saturazione) rispettivamente nel Canale 0 ($\pm$470

mV) e nel Canale 1 ($\pm$660 mV). È evidente la necessità di utilizzare dei trasformatori che precedano entrambi gli ingressi, per poter effettuare la lettura di segnali con ampiezze molto più elevate (provenienti dalla rete elettrica), come vedremo nel paragrafo 4.6.1.

Il segnale CLK deve essere configurato con polarità CPOL$= 0$ e con fase CPHA$= 1$: esso è considerato attivo quando si trova al livello logico alto, e il misuratore MCP3909 trasmette il dato sulla linea MISO alla transizione $0 \rightarrow 1$ (fronte di salita). Di conseguenza il "Master" legge il dato sulla stessa linea alla transizione $1 \rightarrow 0$.

Il fatto che il pin dedicato alla funzione di MISO sia utilizzato anche per trasmettere il "DR bit" comporta, dal lato del "Master", una gestione accurata del segnale, poiché tale informazione deve essere riconosciuta effettivamente come una notifica di disponibilità del dato e non come dato utile. Esamineremo meglio il problema nel paragrafo successivo.

4.4 Scheda "BlueBoard" LPC1768-H

4.4.1 Funzionamento generale

La scheda configurabile "BlueBoard" LPC1768-H (Figura 4.4) è supportata dal microprocessore ARM LPC1768 della serie *Cortex-M3*, caratterizzata da elevate prestazioni e consumi estremamente ridotti, in grado di operare fino a una frequenza di clock pari a 100 MHz. Per la programmazione e il debug è stato utilizzato il sistema "*OpenOCD*" (*Open On Chip Debugger*), costituito dall'adattatore JTAG, per la comunicazione tra il PC (mediante porta USB) e la scheda, e dai relativi driver di supporto. L'adattatore è un piccolo modulo hardware che consente lo scambio di dati mediante il protocollo usato dall'ambiente "*GDB*" (*Gnu DeBugger*), convertendo opportunamente i segnali elettrici provenienti dal PC verso la scheda e viceversa.

La scheda può essere alimentata sfruttando la porta USB o i pin opportuni (in quest'ultimo caso la tensione deve essere compresa tra 4.5 e 9 Volt).

Esaminiamo ora le operazioni principali svolte dalla scheda. All'accensione dello Smart Meter, essa deve innanzitutto configurare alcuni pin e parametri essenziali per: inizializzare il sistema (frequenza di clock del microprocessore, abilitazione degli interrupt, led utilizzati, etc.); abilitare la linea di comunicazione SPI con il misuratore MCP3909 (denominata SPI0) e la linea SPI con la radio (SPI1); inizializzare la radio (canale di trasmissione, abilitazione dei relativi interrupt, etc.). Il microprocessore ha a disposizione tre oscillatori indipendenti per la

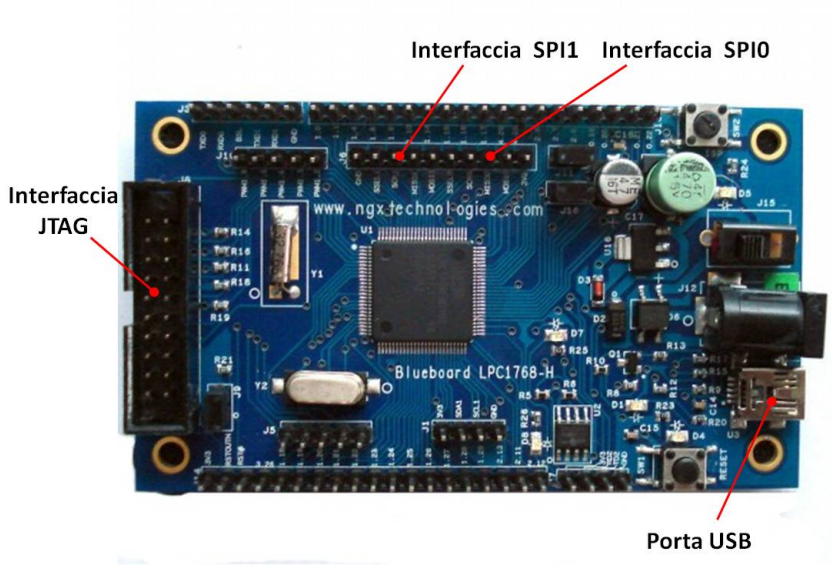


Figura 4.4: Scheda programmabile LPC1768-H e interfacce principali utilizzate.

generazione del segnale di clock, denominati "*IRC*" (di tipo RC), "Principale" e "RTC" (di tipo RTC, *Real Time Clock*). La frequenza di clock della CPU viene impostata a $f_{cpu} = 48$ MHz; ciò richiede l'utilizzo dell'Oscillatore Principale e del circuito integrato PLL (*Phase Locked Loop*). La frequenza di clock f_{s0} dell'interfaccia SPI0 deve essere pari a $f_{cpu}/8$, quindi 6 MHz nel nostro caso. La frequenza f_{s1} dell'interfaccia SPI1 non deve soddisfare una precisa relazione, e viene fissata al valore massimo tollerabile dalla radio, ovvero 8 MHz.

Dopo questa fase di inizializzazione, viene portato a "1" il pin di tipo GPIO (*General Purpose Input Output*) collegato al pin MCLR del misuratore. Viene quindi inviato il byte di comando al misuratore stesso, sulla linea MOSI, per selezionare la modalità SPI desiderata ("Dual Channel Output Mode" nel nostro caso). Abbiamo visto che il comando deve essere inviato all'interno di una finestra temporale ben precisa. In particolare, è necessario iniziare il trasferimento del byte dopo circa 280 ns dall'istante di attivazione del MCLR, e disporre di una frequenza (f_{s0}) sufficientemente elevata per riuscire a trasmettere l'intera sequenza entro 8.9 μs circa. Abbiamo sperimentalmente osservato che, a tal fine, non è necessario misurare il tempo a partire dall'accensione, poiché il microprocessore, dopo la fase di inizializzazione, si trova già all'interno della finestra desiderata. Inoltre, avendo impostato $f_{s0} = 6$ MHz, anche il secondo vincolo risulta soddisfatto.

Si configurano poi le interfacce dello stack protocollare necessario per la comunicazione con il ricevitore TelosB, si definiscono gli *header* dei relativi pacchetti e si imposta la dimensione

massima del *payload*, attualmente pari a 60 Byte. Tenendo conto che ogni campione di tensione o corrente inviato dal MCP3909 è rappresentato da due byte, nello stesso pacchetto sono inviati 15 campioni di entrambi i segnali.

A questo punto viene ripetuto ciclicamente un blocco di istruzioni, fino allo spegnimento del sistema. Vengono abilitati i segnali di interrupt sul pin MISO di SPI0, e lo stesso pin viene temporaneamente adibito alla funzionalità di GPIO. Come abbiamo visto nel paragrafo precedente infatti, l'arrivo di un nuovo dato sul MISO dal MCP3909 è preceduto dall'invio del bit DR sulla *stessa* linea, posto al valore logico "1". Il Master deve quindi utilizzare lo stesso pin dapprima per riconoscere il bit DR (GPIO attivato alla ricezione di un "1"), per poi avviare di conseguenza la *subroutine* che si occupa dell'acquisizione del dato vero e proprio. All'inizio della subroutine, oltre a disabilitare momentaneamente i segnali di interrupt, il pin viene immediatamente adibito alla funzionalità di MISO, per cominciare la ricezione sulla linea SPI0. Una volta memorizzati i quattro byte necessari (due per la tensione e due per la corrente), la subroutine termina e si rientra nel ciclo principale.

Il dato viene salvato all'interno di un buffer di trasmissione. Se questo non risulta ancora pieno, il ciclo riparte daccapo, riabilitando gli interrupt sul pin MISO e predisponendolo nuovamente per la funzionalità di GPIO. In caso contrario, si procede con la trasmissione via radio. I dati contenuti nel buffer vengono suddivisi in gruppi da 60 Byte e inseriti nel payload dei pacchetti, fino al completo svuotamento del buffer. Ciascun pacchetto viene inviato alla radio sulla linea MOSI di SPI1, e da qui viene trasmesso secondo le regole del protocollo standard *IEEE* 802.15.4. Il lampeggiamento di un led indica la corretta trasmissione del pacchetto.

Riportiamo lo pseudo codice eseguito dalla scheda per riassumere quanto appena descritto, adottando la notazione semplificata introdotta nel paragrafo 3.6. Indichiamo con B la capienza del buffer di trasmissione (in Byte): il numero di pacchetti da preparare per svuotarlo è pari a $B/60$. Il tempo di attesa tra l'invio di un pacchetto e il successivo è indicato con t_{wait}, che specificheremo meglio nel paragrafo successivo.

Main LPC1768-H:

$inizializza\ sistema,\ SPI,\ radio$;

$MOSI \leftarrow modalità\ SPI0$;

$configura\ interfacce\ e\ prepara\ header$ 802.15.4;

while (1) **do**

$abilita\ interrupt\ su\ MISO$;

$pin\ MISO \rightarrow GPIO$;

// $uscita\ dalla\ subroutine$

buffer[ ] $\leftarrow dati$;

if ($buffer\ pieno$) **then**

for ($B/60$) **do**

$trasmetti\ pacchetto$;

$attendi\ t_{wait}$;

end for

end if

end while

Subroutine:

$disabilita\ interrupt\ su\ MISO$;

$pin\ GPIO \rightarrow MISO$;

$salva\ dati$;

4.4.2 Velocità di trasmissione

Attualmente si sfrutta la massima velocità di trasmissione (senza perdita di dati), con l'invio di un pacchetto (68 Byte di lunghezza, includendo l'header) circa ogni $t_{wait} = 15$ ms. Il limite massimo è stato determinato sperimentalmente, inserendo nel payload il valore di una variabile "contatore" con funzione di *Sequence Number* e verificando così la ricezione ordinata dei vari pacchetti. Il tempo di attesa t_{wait} è stato quindi progessivamente ridotto fino al raggiungimento del limite minimo, al di sotto del quale si sono riscontrate perdite di dati. La *bitrate* risultante è circa $R_{MAX} = 4.53$ kB/s.

Il calcolo della potenza attiva a partire dai campioni di tensione e corrente non viene

compiuto direttamente dalla scheda, ma periodicamente dal server, dopo la ricezione di un certo numero di pacchetti, mediante la funzione `power_calc`. Abbiamo visto che essa ricorre alla FFT per determinare lo sfasamento tra i due segnali. Attualmente il dominio della Trasformata è costituito da 1024 campioni, pertanto ne sono richiesti altrettanti dal segnale di partenza. In realtà vengono considerati 40 periodi delle sinusoidi, che corrispondono a 840 campioni totali, per quanto visto al paragrafo 4.3.2, mentre i campioni rimanenti sono posti a zero. Lo "*zero padding*" consente tra l'altro di aumentare la risoluzione della Trasformata [10] e di individuare quindi più facilmente le due componenti principali.

Dato che ogni pacchetto contiene 15 campioni, per eseguire il calcolo della potenza attiva sono sufficienti $840/15 = 56$ pacchetti. Sfruttando la massima bitrate raggiungibile, il server ha la possibilità di effettuarlo ogni 840 ms circa. Si tratta naturalmente di una frequenza eccessivamente elevata, considerato soprattutto che non ci aspettiamo variazioni così rapide di potenza nel tempo. Supponendo invece di voler disporre di un campione di potenza attiva ogni 30 secondi, ad esempio, è sufficiente una bitrate pari a

$$R = \frac{4530 \cdot 840}{30000} \approx 127\ B/s.$$

4.5 Sensore TelosB

Il nodo sensore TelosB (Figura 4.5) comunica con il server, e viene al tempo stesso alimentato, mediante la porta USB. Abbiamo utilizzato una delle applicazioni incluse nella directory di installazione del sistema operativo Tiny OS, chiamata "Base Station". Essa consente al nodo di mettersi in ascolto del canale radio specificato e ricevere eventuali pacchetti trasmessi, inserendoli temporaneamente in un buffer. Il lampeggiamento di un led indica la corretta ricezione del pacchetto.

L'applicazione Java "Listen" permette inoltre di visualizzare sullo standard output il conenuto di ogni pacchetto, codificato in formato esadecimale. Nel nostro caso l'output è stato reindirizzato verso il file volt_curr, a cui accede la funzione `power_calc`.

La struttura dei pacchetti è riportata in Figura 4.6. Il primo byte dell'header (formato da bit tutti nulli) indica che il pacchetto è di tipo "*Active Message*" (*AM*). Il sistema AM, adottato nella maggior parte delle reti di sensori wireless, definisce un meccanismo di comunicazione molto rapido tra due nodi [18]. Nell'header del pacchetto viene specificato il cosiddetto "*handler*",

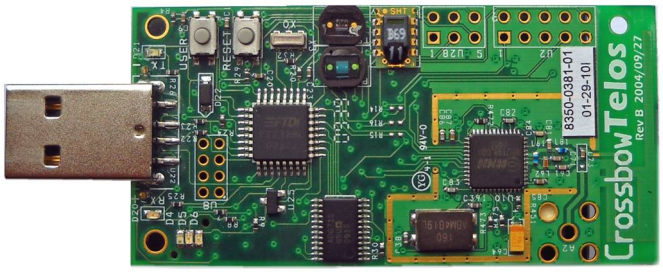


Figura 4.5: Nodo sensore TelosB.

un processo di livello Utente (nello stack protocollare standard TCP/IP), presente nel nodo ricevitore, a cui viene affidata la gestione dei dati trasportati all'interno. Questi ultimi vengono direttamente passati come argomento all'handler, riducendo notevolmente l'overhead introdotto dai livelli intermedi dello stack. Nell'implementazione di Tiny OS, il numero che identifica univocamente l'handler è inserito nell'ultimo byte dell'header (indicato in Figura con *hid, handler ID*).

Il secondo e il terzo byte contengono l'indirizzo del nodo di destinazione (*dst*), mentre il quarto e il quinto quello del nodo sorgente (*src*). Seguono il byte *len*, indicante la dimensione del payload (nel nostro caso 60), e il byte *gid* (*group ID*), utilizzabile per definire un gruppo di nodi che devono condividere le stesse informazioni sul canale (paragonabile all'approccio di tipo "Multicast" per il protocollo IP).

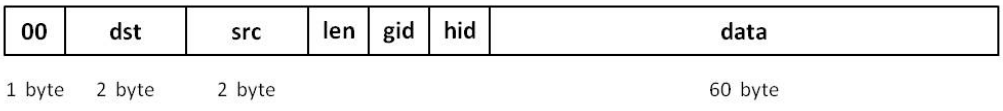


Figura 4.6: Struttura dei pacchetti ricevuti dal sensore TelosB.

4.6 Smart Meter

4.6.1 Caratteristiche del dispositivo

I componenti fondamentali (misuratore MCP3909, scheda LPC1768-H, radio AT86RF231) sono stati posti in un contenitore di dimensioni 20 x 14.5 x 6 cm. È stato aggiunto un trasformatore di corrente (TA) per scalare opportunamente il segnale prelevato dalla rete elettrica e renderlo misurabile in ingresso al canale *CH0* del dispositivo MCP3909. Analogamente è stato aggiunto un partitore resistivo per scalare la tensione e renderla così misurabile in ingresso al canale *CH1*.

Sui lati esterni sono stati resi accessibili: i due pin di alimentazione del misuratore MCP3909 (5 Volt), utilizzati contemporaneamente per alimentare la scheda LPC1768; l'antenna adibita alle comunicazioni via radio (nel nostro caso, alla sola trasmissione dei dati); i due cavi per il collegamento alla rete elettrica. In alternativa, i componenti possono essere alimentati tramite la porta USB della scheda (raggiungibile dal foro praticato sul lato corrispondente del contenitore). Il prototipo di Radio Smart Meter così assemblato è riportato in Figura 4.7.

Per realizzare lo scenario descritto al paragrafo 1.3 è necessario collegare lo SM (Smart Meter) al quadro elettrico principale dell'edificio. Il prototipo attuale tuttavia prevede una corrente massima inferiore a 10 A, ed è stato quindi predisposto alla misurazione di potenza assorbita da carichi meno elevati rispetto a quello massimo sopportabile da una rete domestica. In particolare, la resistenza equivalente del TA è pari a 0.3 Ω; il suo rapporto di trasformazione è 20:5, raddoppiato a 40:5 mediante un doppio avvolgimento del filo elettrico attorno alla bobina. Tenendo conto della tensione massima che possiamo avere in ingresso al canale *CH1* senza entrare in saturazione (660 mV), se indichiamo con i_M la massima corrente secondaria del TA si deve avere:

$$0.3\, {i_M}^{(s)} = 0.66 \;\Rightarrow\; i_M = 2.2\ A.$$

La massima corrente primaria del TA risulta quindi ${i_M}^{(p)} = 8.8$ A; il corrispondente valore efficace è pari a circa 6.22 A, che moltiplicato per il valore efficace nominale di tensione (230 V) determina una massima potenza in ingresso di 1.430 kW.

Per effettuare le misurazioni, uno dei cavi accessibili dall'esterno dello SM viene collegato alla linea elettrica mediante una spina di tipo L da 10 A, mentre il secondo cavo termina con una presa dello stesso tipo, a cui è possibile collegare il carico desiderato.

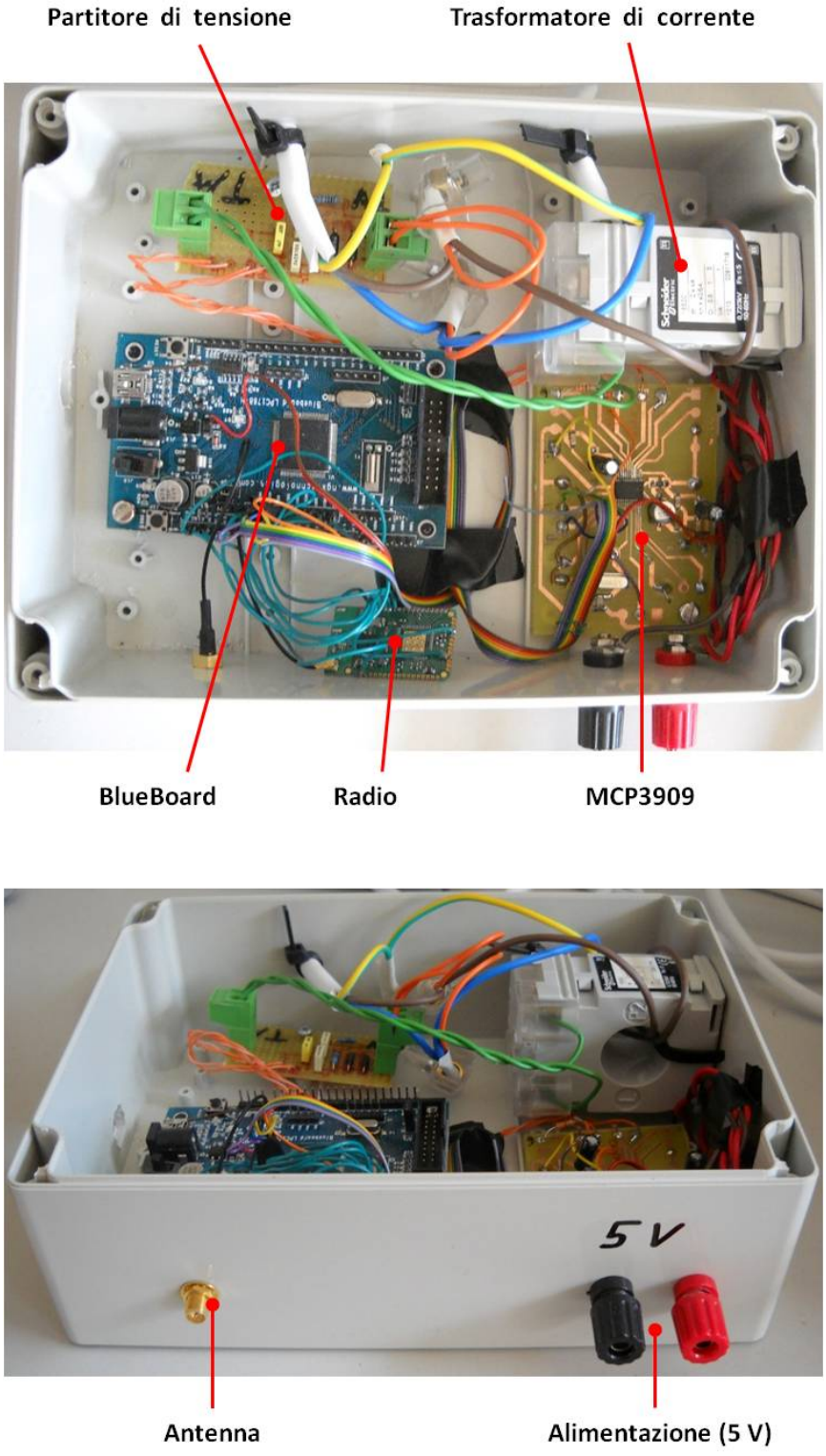


Figura 4.7: Prototipo di Radio Smart Meter elaborato, visuale dall'alto (foto sopra) e di lato (foto sotto).

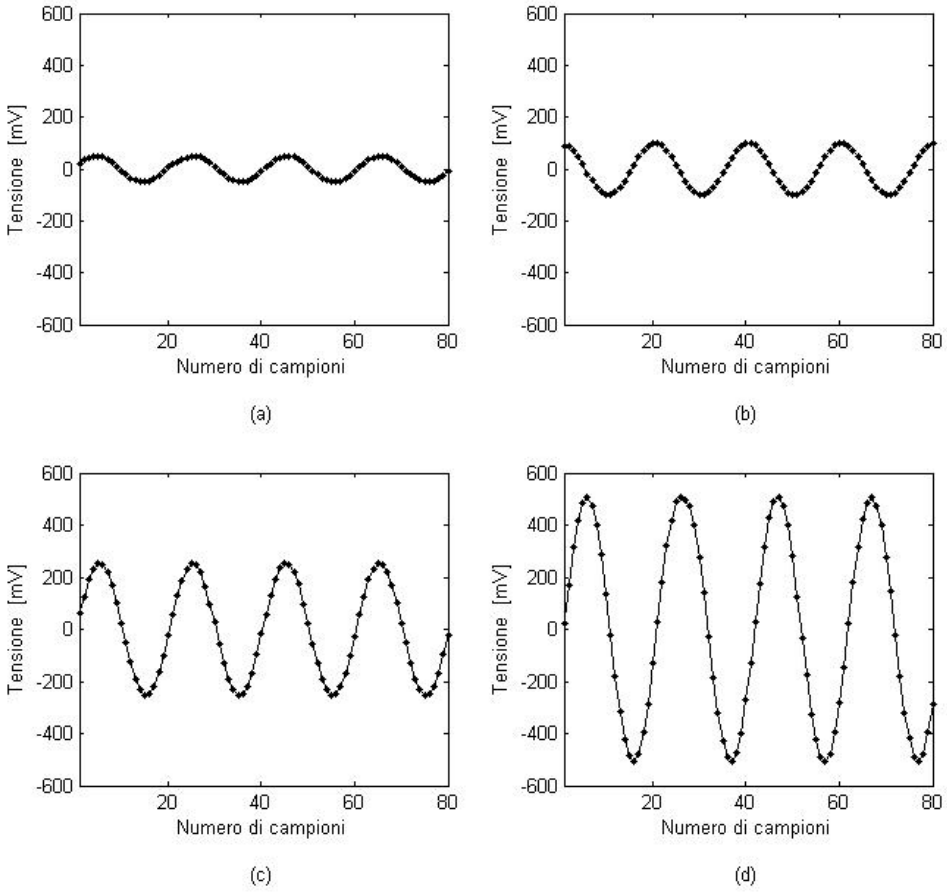


Figura 4.8: Primi 80 campioni ricevuti dal generatore d'onda, con tensione sinusoidale di ampiezza 50 mV (a), 100 mV (b), 250 mV (c) e 500 mV (d).

4.6.2 Risultati delle misurazioni

Sono stati effettuati innanzitutto alcuni test prima dell'assemblaggio finale, utilizzando esclusivamente il misuratore MCP3909 e la scheda programmabile, in modo da verificare la corretta raccolta e trasmissione dei dati. In sostituzione della linea elettrica e del carico, è stato impiegato un generatore d'onda, per fornire in ingresso al canale *CH1* segnali sinusoidali di ampiezze differenti. La Figura 4.8 riporta i primi 80 campioni ricevuti dal server dopo la generazione di un segnale di tensione con ampiezza rispettivamente di 50, 100, 250 e 500 mV, e con frequenza pari a 50 Hz. Si può osservare che la forma sinusoidale viene effettivamente mantenuta, a conferma del corretto funzionamento dei moduli software adibiti alla gestione dei dati e delle interfacce di comunicazione tra i componenti.

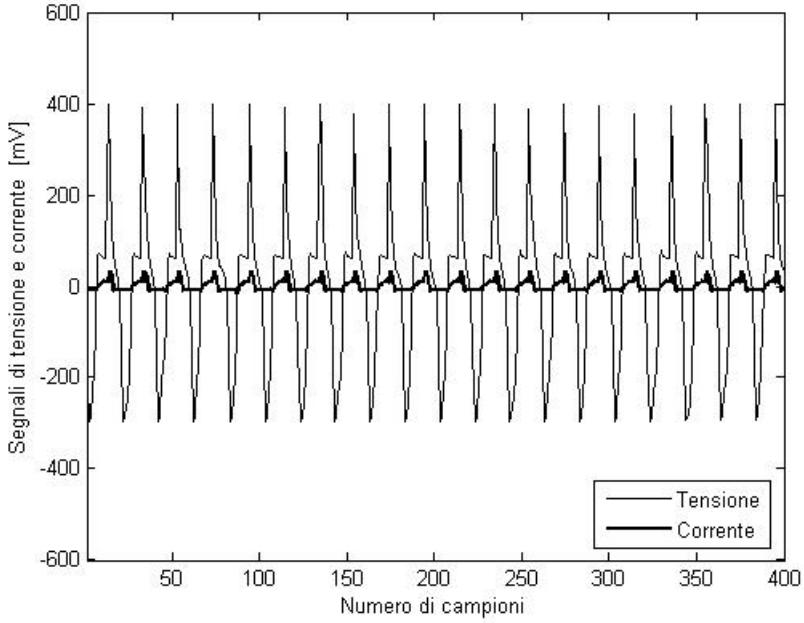
Dopo l'assemblaggio finale, e quindi l'aggiunta dei circuiti di trasformazione di tensione e corrente, è stata eseguita una prima caratterizzazione dello SM attraverso quattro test, in cui il carico è rappresentato da alcune lampadine collegate tra loro in parallelo. Esse sono state collegate alla rete elettrica per mezzo di un trasformatore di isolamento, con conseguente riduzione della tensione assorbita dal carico, dal valore nominale di 220-230 V al valore di 110 V. Tra il carico e il nostro dispositivo inoltre è stato interposto in serie un wattmetro digitale, in modo da confrontarne i valori indicati con quelli prodotti dallo SM. In ogni test sono stati prelevati 2000 campioni, che, per quanto visto al paragrafo 4.3.2, corrispondono a circa 95 periodi delle sinusoidi di tensione e corrente.

I risultati ottenuti non sono stati del tutto soddisfacenti. È possibile verificare l'andamento "pseudo-sinusoidale" dei segnali campionati e la loro differenza di fase (praticamente nulla), ma essi appaiono piuttosto distorti. A titolo di esempio, si riportano i primi 400 campioni di uno dei quattro test in Figura 4.9(a), e i primi 80 campioni dello stesso sottoinsieme in Figura 4.9(b), per maggior chiarezza. I valori corrispondono agli ingressi dei canali *CH0* e *CH1* del MCP3909, espressi in mV. A questo proposito si ricorda che anche il canale finalizzato all'acquisizione di corrente è di tipo differenziale, pertanto il segnale che la rappresenta non ha le dimensioni fisiche di una corrente (Ampere), ma è un segnale di tensione, espresso in mV.

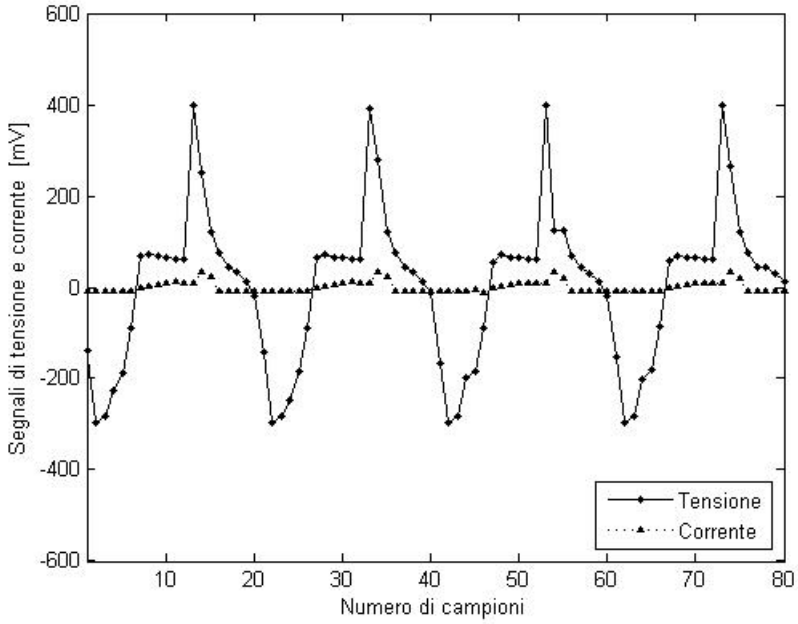
Possiamo notare che i segnali non presentano simmetria tra il valore minimo e quello massimo. È necessario quindi compiere una qualche interpolazione per cercare di ricostruire le ampiezze delle sinusoidi in ingresso e determinare poi i valori efficaci dei segnali originali. Indichiamo con I_M e V_M le ampiezze di corrente e tensione, mentre con $I_M{}'$ e $V_M{}'$ le ampiezze (riscalate) visibili agli ingressi di *CH0* e *CH1*. In questo caso, abbiamo osservato che si ottengono risultati accettabili (vicini a quelli rilevati dal wattmetro) stimando:

- l'ampiezza $V_M{}'$ come la media tra il valore massimo campionato $\tilde{V}_M$ e il modulo del valore minimo campionato $\tilde{V}_m$. Con riferimento alla Figura 4.9, ad esempio, si ottiene
$$V_M{}' = \frac{\tilde{V}_M + |\tilde{V}_m|}{2} = \frac{401.76 + 298.53}{2} = 350.15\ mV;$$
- l'ampiezza $I_M{}'$ come il valore massimo campionato. Con riferimento alla medesima Figura, $I_M{}' = 35.2$ mV.

Abbiamo successivamente utilizzato $V_M{}'$ e $I_M{}'$ per determinare un coefficiente di normalizzazione con cui riscalare i valori raccolti e risalire dunque ai segnali di partenza.



(a)



(b)

Figura 4.9: Primi 400 campioni dei segnali di tensione e corrente prelevati in uno dei quattro test eseguiti (a) e ingrandimento dei primi 80 campioni (b).

Ciò è necessario anche al fine di calcolare la potenza attiva assorbita dal carico. Descriviamo il procedimento adottato riferendoci inizialmente ai campioni di corrente. La Tabella seguente (4.1) riporta i valori efficaci di corrente indicati dal wattmetro (I_{eff}) nei quattro test, i rispettivi valori massimi e le ampiezze stimate $I_M{}'$.

Test	I_{eff}	I_M	$I_M{}'$
1	0.19 A	0.27 A	11.6 mV
2	0.32 A	0.45 A	18.46 mV
3	0.5 A	0.71 A	29.61 mV
4	0.63 A	0.89 A	35.2 mV

Tabella 4.1: Valori di corrente indicati dal wattmetro (A) e in ingresso al canale *CH0* (mV).

Ricordando che il canale *CH0* può ricevere una tensione massima di 470 mV senza entrare in saturazione, la corrispondente corrente massima si può ricavare dalle misure di ciascun test come:

$$i_{max} = \frac{I_M \cdot 470}{I_M{}'},$$

che dovrebbe risultare circa 8.8 A, cioè il valore ricavato al paragrafo precedente. In realtà, se consideriamo ad esempio il primo test, otteniamo $i_{max} \approx 10.94$ A. Ciò è dovuto naturalmente all'errore presente nei dati. Ripetendo il calcolo per tutti e quattro i test e valutando la media delle correnti massime, si ottiene $\bar{i}_{max} \approx 11.38$ A. Il coefficiente di normalizzazione cercato è quindi

$$K_I = \frac{11.38}{470}.$$

Si può procedere in modo analogo con i segnali di tensione. Il valore efficace V_{eff} indicato dal wattmetro è pari a 125 V, da cui ricaviamo $V_M = 176.78$. La Tabella 4.2 riporta anche le ampiezze stimate $V_M{}'$.

Il canale *CH1* può ricevere una tensione massima di 660 mV, perciò la tensione massima corrispondente si può determinare come:

$$v_{max} = \frac{V_M \cdot 470}{V_M{}'}.$$

Mediando ancora una volta rispetto ai valori dei quattro test otteniamo $\bar{v}_{max} \approx 329.91$ V, abbastanza vicino al valore di picco della rete elettrica, pari a 325.27 V (partendo da 230 V

Test	V_{eff}	V_M	$V_M{}'$
1	125 V	176.78 V	357.68 mV
2	125 V	176.78 V	354.07 mV
3	125 V	176.78 V	352.78 mV
4	125 V	176.78 V	350.15 mV

Tabella 4.2: Valori di tensione indicati dal wattmetro (V) e in ingresso al canale *CH1* (mV).

efficaci). Il coefficiente di normalizzazione delle tensioni è quindi

$$K_V = \frac{329.91}{660}.$$

I coefficienti K_I e K_V possono essere utilizzati per riscalare opportunamente tutti i campioni ricevuti dal server.

In questo caso risulta particolarmente semplice calcolare la potenza attiva assorbita dal nostro carico, dato che lo sfasamento fra tensione e corrente è pressoché nullo. La potenza attiva infatti si riduce al prodotto dei valori efficaci. Per ciascun test, essi sono stati determinati a partire dai dati ricevuti:

$$I = \frac{I_M{}' \cdot K_I}{\sqrt{2}};$$

$$V = \frac{V_M{}' \cdot K_V}{\sqrt{2}}.$$

La Tabella seguente (4.3) confronta i valori di potenza attiva indicati dal wattmetro (P) nei quattro test con quelli calcolati $(\tilde{P})$.

Test	P	$\tilde{P}$
1	24.5 W	25.11 W
2	40 W	39.55 W
3	64 W	63.21 W
4	79 W	74.59 W

Tabella 4.3: Valori di potenza attiva indicati dal wattmetro e calcolati dai dati ricevuti.

Capitolo 5

Conclusioni

Nella Tesi è stato proposto un possibile schema per il monitoraggio e il controllo dei consumi elettrici di un edificio, con particolare riferimento a un'ambiente di tipo domestico. Lo schema deve essere esaminato secondo l'ottica del futuro sviluppo della rete elettrica, destinata in pochi anni a diventare un sistema distribuito "intelligente" di gestione e trasporto delle risorse energetiche, definito attualmente in letteratura con il termine "Smart Grid". Tale sistema può essere suddiviso in più sotto-reti dette "Smart Microgrid".

Una parte del nostro lavoro è stata dedicata alla definizione di un modello teorico per descrivere il problema e formulare al tempo stesso una metodologia di risoluzione, basata sulla Programmazione Dinamica. Ciò ha permesso quindi l'implementazione di un algoritmo per lo scheduling delle appliance controllabili, che incidono maggiormente sui consumi energetici a livello locale. L'obiettivo è quello di organizzare l'esecuzione di tutte le attività, impostate dall'utente in un determinato istante, entro le scadenze stabilite, minimizzando però il costo complessivo dovuto all'utilizzo delle risorse energetiche. Per raggiungere tale risultato, l'algoritmo cerca di allocare più energia possibile nei periodi di tempo in cui si rileva una maggior quantità di energia rinnovabile, prodotta dagli appositi impianti installati sull'edificio stesso o su nodi adiacenti, appartenenti alla medesima Smart Microgrid. Questo consente di ridurre non solo il costo monetario del consumatore, ma anche gli sprechi energetici, in modo da contribuire positivamente all'efficienza dell'intera Rete. Quando non è possibile ricorrere a fonti rinnovabili, l'algoritmo cerca di allocare l'energia necessaria dei periodi in cui essa presenta un costo inferiore.

Dalle simulazioni compiute sono stati ricavati ottimi risultati sia dal punto di vista del-

l'efficacia (pianificazione delle attività secondo i criteri riportati sopra) sia dal punto di vista dell'efficienza (tempo massimo di esecuzione all'inizio di ogni intervallo dell'ordine di un paio di secondi). Considerando quest'ultimo risultato, si può pensare di aumentare la precisione dell'algoritmo riducendo la durata degli intervalli in cui viene discretizzato il tempo (slot) e aumentando il numero degli stati che rappresentano i possibili livelli di potenza elettrica, ottenendo comunque tempi di esecuzione soddisfacenti.

La seconda parte del nostro lavoro è stata riservata alla realizzazione in laboratorio di un prototipo di Radio Smart Meter, per la misurazione in tempo reale di tensione, corrente e potenza attiva assorbita dalla rete elettrica. Lo scopo principale del dispositivo è quello di fornire i dati necessari all'esecuzione dell'algoritmo di scheduling descritto sopra. Sono stati realizzati i collegamenti e le interfacce software per la comunicazione tra i diversi componenti (microcontrollore, circuito integrato per la misurazione energetica e radio), adottando come sistema principale l'interfaccia SPI. Tutto è stato poi assemblato in un unico contenitore, assieme agli opportuni circuiti di trasformazione e alimentazione. I dati raccolti possono essere trasmessi via radio tramite il protocollo standard IEEE 802.15.4. In ricezione è stato utilizzato il nodo sensore TelosB.

Nel prototipo attuale, come sì è visto, la potenza attiva non viene calcolata direttamente dallo Smart Meter, ma dal server, dopo la ricezione di un certo numero di campioni di tensione e corrente. In uno stadio successivo si vuole correggere questo aspetto, facendo in modo che il server riceva unicamente i valori di potenza attiva, o in alternativa, i valori efficaci di tensione e corrente assieme a quelli di potenza. Questo prevede la configurazione, all'interno della scheda programmabile, dell'apposito modulo software adibito all'emulazione dell'Unità Aritmetica per operazioni di tipo Floating Point (non è infatti presente la relativa Unità hardware).

I risultati ottenuti dai test non sono stati del tutto soddisfacenti: si riscontrano alcuni problemi molto probabilmente legati ai circuiti di alimentazione e trasformazione o a collegamenti fisici tra i diversi componenti realizzati in modo non del tutto corretto, che introducono una considerevole distorsione nei segnali campionati dal dispositivo. Il prototipo deve pertanto essere revisionato.

Una volta ottenuto un prototipo perfettamente funzionante, sarà possibile aumentare il livello di integrazione dei relativi componenti, affiancando al microcontrollore principale (LPC1768) un'unico circuito di dimensioni notevolmente ridotte.

Tra gli sviluppi futuri del nostro lavoro, inoltre, va sicuramente dato spazio a una sistematica

raccolta di misure di potenza assorbita da varie appliance reali, in modo da tracciarne dei profili energetici dettagliati. Ciò richiederà l'utilizzo di un trasformatore di corrente con un rapporto di trasformazione lievemente superiore a quello attuale, che è in grado di sopportare un carico massimo di 1.4 kW. Allo stesso modo, è utile rilevare la potenza assorbita da diversi gruppi di appliance definite come "non controllabili", e la potenza generata dai tipici moduli fotovoltaici installabili sugli edifici. Queste misure consentono di tracciare alcuni possibili profili energetici delle due grandezze e quindi offrono informazioni importanti per le simulazioni dell'algoritmo di scheduling.

Disponendo di un vasto ambiente in cui sono presenti numerosi apparecchi elettrici, è possibile inoltre organizzare il monitoraggio di più aree, anche lontane tra loro, dotandole ciascuna del Radio Smart Meter presentato. Tramite una rete costituita da nodi sensori TelosB o di tipo analogo, i dispositivi possono trasmettere i dati raccolti ad un unico server centrale, per la successiva elaborazione di statistiche o altro.

Un altro possibile sviluppo del nostro lavoro può essere l'utilizzo combinato dello Smart Meter con opportuni algoritmi di *Signal Processing*, allo scopo di individuare dei *pattern* specifici nei segnali di tensione, corrente o potenza associati a diverse appliance. Questo consente il riconoscimento di ciascuna di esse e, soprattutto, la distinzione da altre appliance, anche qualora si trovino ad essere attive simultaneamente, come avviene nella maggior parte dei casi. Un sistema di questo tipo permetterebbe un monitoraggio estremamente dettagliato, senza la necessità di disporre di uno Smart Meter per ogni appliance analizzata.

Bibliografia

[1] S.Papathanassiou, N.Hatziargyriou, K.Strunz, "A benchmark low voltage microgrid network". *CIGRE Symposium*, Atene, 2005.

[2] Enel.it, "Smart Grids", http://www.enel.it/it-IT/reti/enel_distribuzione/qualita/progetti_smart_grids (ultima consultazione Maggio 2011).

[3] DOE - Nation Energy Technology Laboratory, "The NETL Smart Grid Implementation Strategy", http://www.netl.doe.gov/smartgrid/ (ultima consultazione Maggio 2011).

[4] A.Molderink, V.Bakker, G.C.Bosman, J.L.Hurink, G.J.M.Smit, "Management and Control of Domestic Smart Grid Technology". *IEEE Transactions on Smart Grid*, vol.1, 2010.

[5] A.Molderink, V.Bakker, G.C.Bosman, J.L.Hurink, G.J.M.Smit, "The microCHP scheduling problem". *2nd Global Conference on Power Control Optimization*, Bali, Indonesia, 2009.

[6] A.-H.Mohsenian-Rad, A. Leon-Garcia, "Optimal Residential Load Control With Price Prediction in Real-Time Electricity Pricing Environments". *IEEE Transactions on Smart Grid*, vol.1, 2010.

[7] Y.Agarwal, T.Weng, R.K.Gupta, "Mycro-Systems Driving Energy Metering in Smart Grids". *47th Design Automation Conference*, Anaheim, California, 2010.

[8] G.Cariolaro, G.Pierobon, *Processi aleatori* (4a Edizione), CLEUP, Padova, Italia, 2008.

[9] D.P.Bertsekas, *Dynamic Programming and Optimal Control* (3rd Edition), Athena Scientific, Belmont, Massachusetts, 2005.

[10] S.K:Mitra, *Digital Signal Processing: A Computer-Based Approach* (3rd Edition), McGraw Hill, New York, 2006.

[11] Enea.it, “L’etichetta energetica”, http://old.enea.it/produzione_scientifica/pdf_op_svil_sost/Op24.pdf, Dicembre 2003 (ultima consultazione Maggio 2011).

[12] Microchip.com, MCP3909 Data Sheet, http://ww1.microchip.com/downloads/en/DeviceDoc/22025b.pdf, rel.22025B (aggiornato Aprile 2009).

[13] NXP.com, LPC17xx User Manual, http://www.nxp.com/documents/user_manual/UM10360.pdf, rev.2 (aggiornato Agosto 2010).

[14] NGX Technologies Online Store, LPC1768-H v1.0 User Manual, http://shop.ngxtechnologies.com/download/header/LPC1768/NGX_BlueBoard_LPC1768_H.pdf (aggiornato Ottobre 2010).

[15] Atmel.com, AT86RF231-ZU/ZF, http://www.atmel.com/dyn/resources/prod_documents/doc8111.pdf (aggiornato Settembre 2009).

[16] Willow.co.uk, TelosB Data Sheet, http://www.willow.co.uk/TelosB_Datasheet.pdf, rev.A (ultima consultazione Maggio 2011).

[17] Microchip.com, “SPI - Overview and use of the PICmicro Serial Peripheral Interface”, http://ww1.microchip.com/downloads/en/devicedoc/spi.pdf (ultima consultazione Maggio 2011).

[18] T. von Eicken, D.E.Culler, S.C.Goldstein, K.E.Schauser, “Active Messages: A mechanism for integrated communication and computation”. *Proceedings of the 19th Annual International Symposium on Computer Architecture*, Queensland, Australia, 1992.