Agenzia nazionale per le nuove tecnologie,
l'energia e lo sviluppo economico sostenibile

MINISTERO DELLO SVILUPPO ECONOMICO

Ottimizzazione multi-obiettivo in scenari Demand-Response di un edificio terziario reale

A. Monteriù, L. Ciabattoni, F. Ferracuti, G. Comodi, S. Longhi

Report RdS/PAR2017/097

OTTIMIZZAZIONE MULTI-OBIETTIVO IN SCENARI DEMAND-RESPONSE DI UN EDIFICIO TERZIARIO REALE

Andrea Monteriù, Lucio Ciabattoni, Francesco Ferracuti, Gabriele Comodi, Sauro Longhi
(Università Politecnica delle Marche)

Settembre 2018

Report Ricerca di Sistema Elettrico
Accordo di Programma Ministero dello Sviluppo Economico - ENEA
Piano Annuale di Realizzazione 2017
Area: Efficienza energetica e risparmio di energia negli usi finali elettrici ed interazioni con altri vettori energetici
Progetto: D.1 "Tecnologie per costruire gli edifici del futuro"
Obiettivo: D "Gestione di edifici in contesto Smart District e scenari di Demand-Response"
Responsabile del Progetto: Ing. Giovanni Puglisi, ENEA

Il presente documento riporta una sintesi delle attività di ricerca svolte all'interno dell'Accordo di collaborazione "Ottimizzazione multi-obiettivo in scenari Demand-Response di un edificio terziario reale"

Responsabile scientifico ENEA: Ing. Stefano Pizzuti

Responsabile scientifico Università Politecnica delle Marche: Dott. Ing. Andrea Monteriù

Indice

Indice figure

SOMMARIO

Grazie alle diverse tecnologie sviluppatesi nell'ultimo decennio in ambito energetico, gli edifici sono divenuti oramai sistemi energicamente complessi da gestire. In effetti, l'attenzione non è più rivolta soltanto al risparmio energetico in senso stretto, ma anche ad altri importanti fattori quali, ad esempio, la gestione e la massimizzazione del comfort della persona e dell'edificio in generale, la riduzione della spesa energetica e della spesa economica, la massimizzazione dell'autoconsumo da fonti rinnovabili, ed altri ancora. L'ulteriore difficoltà nasce dal fatto che spesso tali fattori sono contrastanti fra loro e, di conseguenza, per poter risolvere il problema energetico, risulta fondamentale utilizzare strategie di ottimizzazione mono- e multi-obiettivo che permettano di determinare la combinazione dei parametri di regolazione del sistema.

In questo contesto, l'attività svolta da ENEA ed Università Politecnica delle Marche in questa annualità mira proprio a fornire uno strumento per l'ottimizzazione di obiettivi specifici di natura energetica/economica/comfort di un edificio del terziario mediante l'utilizzo di pompa di calore, pannelli fotovoltaici anche in scenari di dynamic pricing.

In particolare, in questa annualità l'attività di ricerca si è concentrata nella predisposizione del simulatore di carichi elettrici e termici con algoritmi di ottimizzazione e la loro implementazione in scenari demand-response dell'edificio terziario reale F40 del Centro Ricerche di ENEA Casaccia. L'edificio F40, attualmente riscaldato dalla rete di teleriscaldamento e raffrescato con chiller elettrici, è stato simulato in una versione completamente "elettrica", vale a dire: si è simulata la climatizzazione (estiva ed invernale) con pompa di calore; l'installazione di un impianto fotovoltaico di potenza nominale pari a 20 kWp e di uno storage elettrico da circa 40 kWh. Lo scenario prevede anche un prezzo orario dell'energia variabile (dynamic pricing).
Il software della precedente annualità è stato modificato cosicché algoritmi di ottimizzazione possano accedere alle variabili decisionali e il risultato dell'ottimizzazione possa essere inviato al simulatore stesso. In particolare, lo scambio delle suddette variabili avverrà mediante l'utilizzo di variabili condivise sul Workspace Matlab e file ".mat".
Successivamente sono stati definite delle funzioni di costo da ottimizzare con obiettivi specifici riguardanti ad esempio la minimizzazione della spesa energetica, la minimizzazione della spesa economica, la massimizzazione dell'autoconsumo da fonti rinnovabili, l'ottimizzazione del comfort, o una loro combinazione.
Inoltre, in ambiente Matlab/Simulink, l'algoritmo di ottimizzazione Particle Swarm Optimization (PSO) è stato integrato nel software e utilizzato per la gestione ottimale dell'edificio F40. L'algoritmo di ottimizzazione PSO è stato confrontato con l'algoritmo Artificial Bee Colony (ABC). Infine, l'ottimizzazione multi-obiettivo è sviluppata integrando l'algoritmo Multi-Objective PSO (MPSO).

Tutti gli scenari testati dimostrano come l'utilizzo di uno strumento di ottimizzazione dell'edificio possa contribuire a sfruttare al massimo l'autoconsumo di energia prodotta dall'impianto rinnovabile e/o a minimizzare la spesa economica e/o massimizzare il comfort dell'edificio.

1 Introduzione

I temi sviluppati nell'ambito del presente accordo di collaborazione tra ENEA e l'Università Politecnica delle Marche, riguardano il miglioramento di un simulatore di smart buildings per testare funzionalità di smart energy management in presenza di produzione da fonti rinnovabili, storage elettrico e in contesti di dynamic pricing.
Il Ministero dello Sviluppo Economico ed ENEA hanno stipulato un Accordo di Programma in base al quale è concesso il contributo finanziario per l'esecuzione delle linee di attività del Piano Triennale 2015-2017 della Ricerca e Sviluppo di Interesse Generale per il Sistema Elettrico Nazionale.
Il presente allegato tecnico si riferisce al Piano Annuale di Realizzazione 2017, per quanto attiene all'Area "Efficienza energetica e risparmio di energia negli usi finali elettrici ed interazioni con altri vettori energetici", tematica di ricerca "Edifici intelligenti"; nello specifico, si riferisce all'obiettivo D "Gestione di edifici in contesto Smart District e scenari di Demand-Response", sub-obiettivo d.1 "Ottimizzazione multi-obiettivo in scenari Demand-Response di un edificio terziario reale" per aggregati di edifici terziari", del progetto D.1 "Tecnologie per costruire gli edifici del futuro".
I temi sviluppati nell'ambito del presente accordo di collaborazione tra ENEA e il Dipartimento di Ingegneria dell'Informazione dell'Università Politecnica delle Marche riguardano l'integrazione, l'implementazione e l'analisi di algoritmi di ottimizzazione energetica/economica/comfort di un simulatore di edificio già oggetto di sviluppo nel precedente triennio in scenari di Demand-Response.

In particolare, le attività svolte in questo PAR sono state:

- Integrazione di algoritmi di ottimizzazione con il simulatore edificio F40:
  - accesso alle variabili decisionali tra ottimizzatore e simulatore edificio mediante l'utilizzo di variabili condivise sul Workspace Matlab e file ".mat";
  - parallelizzazione delle simulazioni dell'edificio per velocizzare il processo di ottimizzazione.
- Definizione di diversi problemi di ottimizzazione.
- Implementazione e analisi di algoritmi di ottimizzazione come l'algoritmo Particle Swarm Optimization.

2 Descrizione delle attività svolte e risultati

Nelle annualità precedenti, è stato sviluppato un simulatore di edifici applicato sia a singoli edifici che a cluster di edifici. Nel tempo sono state sviluppate diverse funzionalità. In questa annualità il simulatore è stato sfruttato per analizzare e testare diverse politiche di gestione ottimale dell'edificio F40.
In particolare, le linee di attività principali di questa annualità, sono state:

1. Integrazione di algoritmi di ottimizzazione con il simulatore edificio F40.
2. Definizione del problema di ottimizzazione.
3. Implementazione ed analisi risultati.

2.1 Simulatore dell'edificio F40 nello scenario di utilizzo di: impianto fotovoltaico, pompa di calore, storage elettrico e dynamic pricing

Nelle annualità precedenti il simulatore è stato migliorato aggiungendo nuove funzionalità quali: la predisposizione alla simulazione di scenari di dynamic pricing; integrazione di storage sia elettrici che termici; integrazione di impianti fotovoltaici. Nelle annualità precedenti [1, 2, 3, 4], le componenti e le funzionalità del simulatore, sono state integrate tutte assieme ai fini della simulazione di edifici dotati di generazione distribuita, storage e carichi elettrici controllabili.

In questa annualità l'attività di ricerca si è concentrata nell'analisi di diverse politiche di gestione ottimale dell'edificio F40 dotato di pompa di calore, nello scenario di installazione di un impianto fotovoltaico di potenza nominale pari a 20 kWp e di un prezzo orario dell'energia variabile (dynamic pricing). In particolare, sono stati simulati e ottimizzati diversi funzionali considerando due settimane lavorative: una rappresentativa della stagione estiva, ed una rappresentativa della stagione invernale.

La Figura 2-1 mostra la schermata iniziale del software di simulazione, integrato con tutti i suoi componenti e con tutte le sue funzionalità ampiamente descritte nei Report tecnici delle annualità precedenti.

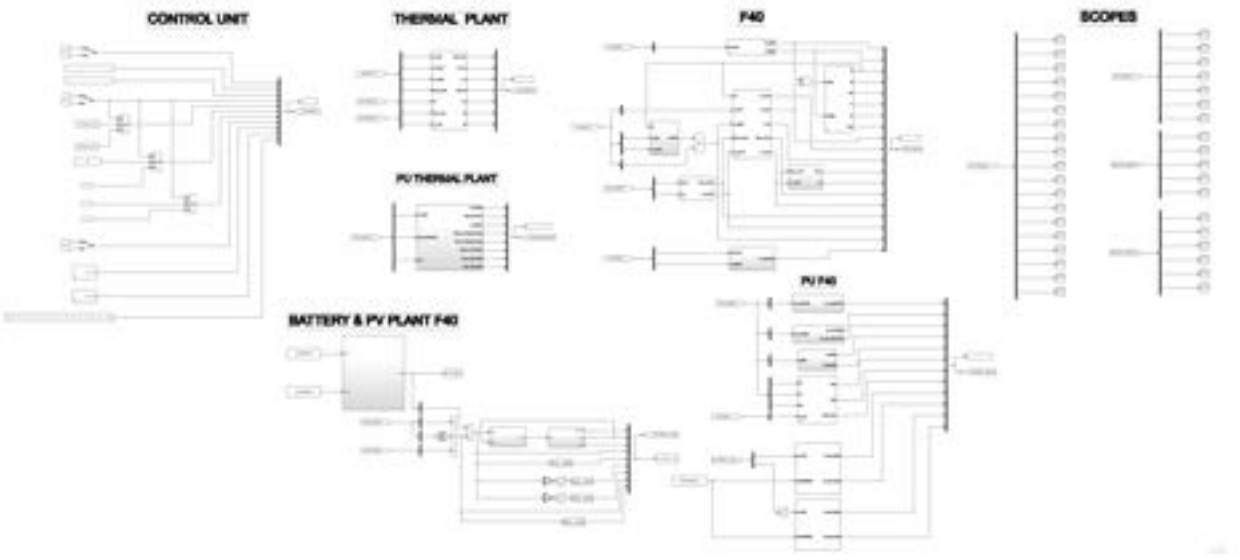

Figura 2-1 Schema a blocchi del Simulatore F40

2.1.1 Integrazione del sistema PV ed energy storage con l'algoritmo di ottimizzazione

Nell'annualità precedente sono stati integrati nel simulatore due modelli MATLAB/SIMULINK per la simulazione di impianti fotovoltaici e per la simulazione di accumuli di energia sia termica che elettrica. Al fine di integrare i due sistemi con l'algoritmo di ottimizzazione, alcuni Key Performance Indicators (KPI) sono stati estratti e salvati nel workspace del Matlab per calcolare le funzioni obiettivo da minimizzare e/o massimizzare.
La Figura 2-2, mostra l'integrazione e i cambiamenti effettuati in questa annualità, all'interno del simulatore, dei modelli fotovoltaico e batteria.

Al momento, il modello della batteria presenta un blocco BMS che attua logiche di gestione molto semplici:

- La batteria si carica soltanto quando c'è un eccesso di produzione dell'impianto fotovoltaico; non è stata simulata la possibilità di caricare la batteria con la rete per simulare scenari di cosiddetto "price arbitrage", vale a dire una gestione della batteria fatta solamente sulla differenza di prezzo tra energia caricata e quella scaricata;

- La batteria si scarica quando c'è domanda di energia elettrica ma non c'è produzione fotovoltaica o è insufficiente;

- La scarica della batteria non può scendere al di sotto del 30% dello stato di carica (State Of Charge, SOC).

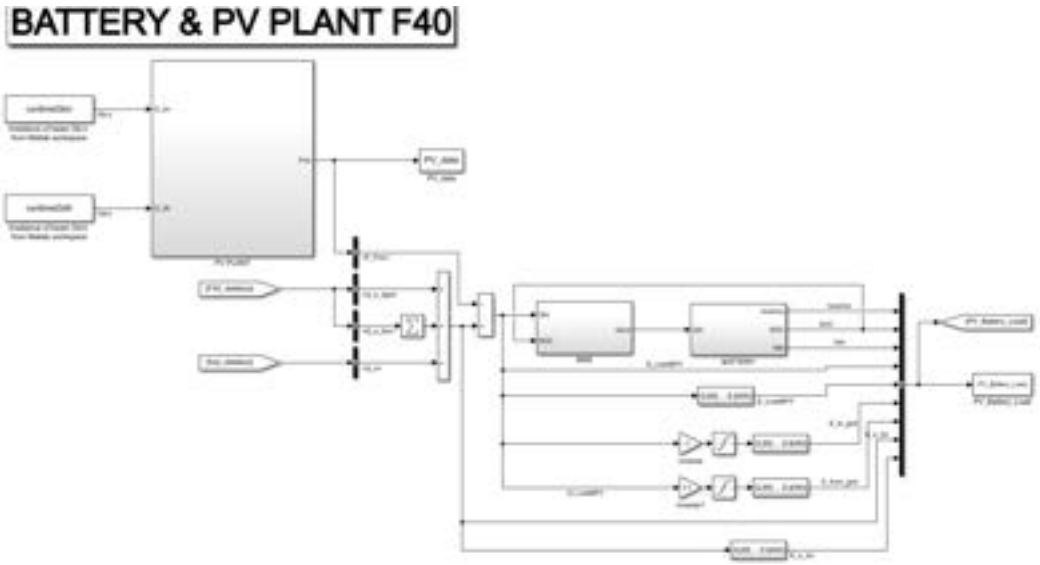

Figura 2-2 Schema a blocchi del sottosistema Batteria e Fotovoltaico

2.2 Integrazione di algoritmi di ottimizzazione con il simulatore edificio F40

Il software sviluppato nella precedente annualità è stato modificato in maniera tale che gli algoritmi di ottimizzazione possano accedere alle variabili decisionali e il risultato finale dell'ottimizzazione possa essere inviato al simulatore stesso. In particolare, lo scambio delle suddette variabili avverrà mediante l'utilizzo di variabili condivise sul Workspace Matlab e/o file ".mat".

Lo scambio di variabili tra il simulatore Simulink e il workspace di Matlab è stato possibile grazie all'utilizzo delle seguenti funzioni.

- *set_param*
- *setBlockParameter*

In particolare, *set_param* permette di impostare il parametro al valore specificato del modello o del blocco specificato mentre *setBlockParameter* permette di impostare il parametro solamente il/i valore/i del blocco specificato e non del modello anche durante una simulazione che utilizza il calcolo parallelo. Come è possibile vedere nella procedura Avvia Simulazione in Parallelo, la funzione

```
setBlockParameter(in(i),[model '/Winter Temperature Setpoints [°C]'],...
    'Value',strcat('[',num2str(paramsOpt(1:15)),']'));
```

permette di impostare i setpoints delle temperature invernali delle 15 stanze di cui è composto l'edificio.

Al fine di velocizzare il processo di ottimizzazione si è scelto di adottare tre opzionalità offerte dal Simulink che sono di aiuto quando si vuole simulare iterativamente un modello Simulink. Queste procedure si basano sulla parallelizzazione dei modelli Simulink da eseguire e sull'utilizzo di un workflow Simulink più snello, in particolare le tre opzionalità adottate sono:

- *parsim*;
- *fast restart*;
- *simulation mode: accelerator*.

Parsim permette di simulare in parallelo più modelli Simulink, il cui numero massimo dipende dal numero di core che un computer possiede. Nel codice Simulazione Parallela viene definita la procedura utilizzata per settare l'utilizzo di 4 core per l'esecuzione in parallelo di 4 modelli Simulink dell'edificio.

%%%%%%%%%%%%%%%%%% Simulazione Parallela %%%%%%%%%%%%%%%%%%

```
poolobj=gcp('nocreate');
if isempty(poolobj)
   parpool(4);
else
   delete(gcp('nocreate'));
end
```

Dopo il processo di inizializzazione, l'esecuzione del comando *parsim*, nella procedura Avvia Simulazione in Parallelo, permette di calcolare la funzione obiettivo di tutte le particelle dell'algoritmo di ottimizzazione in maniera parallela.

%%%%%%%%%%%%%%%%%% Inizializzazione %%%%%%%%%%%%%%%%%%%%%%

```
if ndays==1
   set_param(model,'SimulationMode','accelerator','ReturnWorkspaceOutputs','on',...
'FastRestart','off','LoadInitialState','off','SaveFinalState','on','FinalStateName','finalstate',...
      'SaveCompleteFinalSimState','on','StartTime',num2str(0),'StopTime',num2str(day-1));
else
   set_param(model,'SimulationMode','accelerator','ReturnWorkspaceOutputs','on',...
'FastRestart','off','LoadInitialState','on','InitialState','finalstate','SaveFinalState','on','FinalStateName','finalstate',...
      'SaveCompleteFinalSimState','on','StartTime',num2str(day*(ndays-1)),'StopTime',num2str(ndays*day-1));
end
save_system(model);
```

La funzionalità *fast restart* può essere utilizzata quando è necessario simulare un modello in modo iterativo evitando di compilare il modello ad ogni iterazione. Il modello comunque utilizza i nuovi valori dei parametri e degli ingressi forniti dalle precedenti simulazioni. La funzionalità *fast restart* viene impostata nella procedura Inizializzazione.

Al fine di accelerare la simulazione del modello Simulink è stata settata la funzionalità *simulation mode* da *normal* a *accelerator*, come mostrato nella procedura Inizializzazione. Di default, la modalità acceleratore utilizza l'accelerazione Just-in-Time (JIT) per generare un motore in memoria invece di generare codice C o file MEX, velocizzando la simulazione rispetto alla modalità di simulazione *normal*.

%%%%%%%%%%%%%%% Avvia Simulazione in Parallelo %%%%%%%%%%%%%%%%%%%

```
for i=1:options.SwarmSize
   paramsOpt=state.Positions(i,:);
   in(i)=Simulink.SimulationInput(model);
   in(i)=setBlockParameter(in(i),[model '/Winter Temperature Setpoints [°C]'],'Value',strcat('[',num2str(paramsOpt(1:15)),']'));
   in(i)=setBlockParameter(in(i),[model '/Winter Disc. Temp. Setpoint [°C]'],'Value',num2str(paramsOpt(16)));
end

out=parsim(in,'ShowProgress','on','TransferBaseWorkspaceVariables','on','UseFastRestart','on');

for i=1:options.SwarmSize                                    f1=c1*(mean(out(i).PV_Battery_Load.E_from_grid.Data));
f2=c2*(mean(out(i).F40meter_data.E_cost_flat_PV.Data)); f3=c3*(mean(out(i).F40meter_data.E_cost_dyn_PV.Data));
f4=c4*(mean(out(i).PV_Battery_Load.E_to_grid.Data)); f5=c5*(max(out(i).F40meter_data.PMV_mean.Data)); v=out(i).PV_Battery_Load.E_to_grid.Data-
out(i).PV_data.E_PVo.Data;                                   f6=c6*mean(v)/(mean(out(i).PV_data.E_PVo.Data));
   state.Fvals(:,i)=f1;
end
```

Infine la procedura Salva i risultati ottimi permette di salvare i parametri decisionali e la funzione obiettivo ottimi e di salvare in un file ".mat" anche i dati della simulazione del modello settato con i parametri ottimi trovati dall'algoritmo di ottimizzazione.

%%%%%%%%%%%%% Salva i risultati ottimi %%%%%%%%%%%%%%%%%%%%%%%%%

```
if ndays==1
   set_param(model,'SimulationMode','accelerator','ReturnWorkspaceOutputs','on',...
'FastRestart','off','LoadInitialState','off','SaveFinalState','on','FinalStateName','finalstate',...
      'SaveCompleteFinalSimState','on','StartTime',num2str(0),'StopTime',num2str(day-1));
else
   set_param(model,'SimulationMode','accelerator','ReturnWorkspaceOutputs','on',...
'FastRestart','off','LoadInitialState','on','InitialState','finalstate','SaveFinalState','on','FinalStateName','finalstate',...
      'SaveCompleteFinalSimState','on','StartTime',num2str(day*(ndays-1)),'StopTime',num2str(ndays*day-1));
end
save_system(model);
set_param([model '/Winter Temperature Setpoints [°C]'],'Value',strcat('[',num2str(paramsOpt(1:15)),']'));
set_param([model '/Winter Disc. Temp. Setpoint [°C]'],'Value',num2str(paramsOpt(16)));
sim(model);

save(strcat('PSO_',num2str(ndays)),'BestChart','BESTparam');
```

2.3 Definizione del problema di ottimizzazione

In questa annualità, il lavoro è stato incentrato nella definizione delle funzioni di costo, o funzioni obiettivo, da ottimizzare con obiettivi specifici riguardanti ad esempio la minimizzazione della spesa energetica, la minimizzazione della spesa economica, la massimizzazione dell'autoconsumo da fonti rinnovabili, l'ottimizzazione del comfort, o una loro combinazione.
Alcune funzioni obiettivo che sono state studiate sono:

- Minimizzare la spesa energetica in un dato periodo N.

$$J(V_I) = \sum_{i=1}^{N} Ea_i(V_I) = \sum_{i=i}^{N} sat(Ec_i(V_I) - Ep_i(V_I))$$

dove Ea è l'energia assorbita dalla rete, Ec l'energia consumata ed Ep l'energia prodotta nel periodo i-esimo. Si noti l'impiego della funzione saturazione in modo da considerare solo i valori positivi di energia (solo assorbimento da rete) senza considerare invece l'immissione di energia in rete per eccesso di produzione.

- Minimizzare la spesa economica.

$$J(V_I) = \sum_{i=1}^{N} c_i \times Ea_i(V_I) = \sum_{i=i}^{N} c_i \times sat(Ec_i(V_I) - Ep_i(V_I))$$

dove c è il costo dell'energia, Ea l'energia assorbita, Ec l'energia consumata ed Ep l'energia prodotta nel periodo i-esimo. Si noti l'impiego della funzione saturazione in modo da considerare solo i valori positivi di energia (solo assorbimento da rete) senza considerare invece l'immissione di energia in rete per eccesso di produzione. In questa particolare situazione si potrà altresì considerare il costo di investimento per l'installazione e l'acquisto delle risorse, spalmato in un orizzonte temporale pluriennale, utilizzando le funzioni VAN (valore attuale netto) e TAN (tasso attualizzazione netto).

- Massimizzare la percentuale di autoconsumo da fonti rinnovabili.

$$J(V_I) = \frac{\sum_{i=1}^{N} Ep_i(V_I) - sat(Ep_i(V_I) - Ec_i(V_I))}{\sum_{i=1}^{N} Ep_i(V_I)}$$

dove Ec l'energia consumata ed Ep l'energia prodotta nel periodo i-esimo. Si noti l'impiego della funzione saturazione in modo da considerare solo i valori positivi di energia che consente di considerare solo il consumo effettivo della risorsa rinnovabile senza considerare invece l'assorbimento di energia dalla rete per eccesso di consumi.

- Massimizzare l'autonomia dalla rete.

Questo problema di massimizzazione sarà tramutato in un problema di minimizzazione dove si considera il minimo dell'assorbimento (come nel primo caso):

$$J(V_I) = \sum_{i=1}^{N} Ea_i(V_I) = \sum_{i=i}^{N} sat(Ec_i(V_I) - Ep_i(V_I))$$

- Massimizzare il comfort.

In questa circostanza il funzionale utilizzato sarà direttamente il Predicted Mean Vote (PMV).

- Combinazione di più obiettivi.

Utilizzando ottimizzatori multi-obiettivo sarà possibile combinare due o più funzionali definiti in precedenza per ottenere dei valori delle variabili decisionali che siano atti all'ottimizzazione di tali funzioni di costo (ovvero, otterranno il miglior compromesso tra i diversi obiettivi).

Le variabili decisionali considerate, di cui è stato determinato il valore ottimo, sono i 15 setpoints di temperatura delle stanze e il setpoint di mandata dell'acqua, per un totale di 16 variabili decisionali.

2.4 Definizione del Problema

Nell'affrontare un problema di ottimizzazione, possono essere utilizzati vari approcci, a seconda sia della difficoltà specifica e delle dimensioni del problema in esame, sia degli obiettivi reali che si vogliono ottenere. In quei casi in cui sia necessario pervenire alla soluzione ottima di un certo problema, si utilizzerà un approccio di enumerazione implicita, sia esso basato su una formulazione di programmazione intera, o sullo sfruttamento delle proprietà combinatorie del problema. Se è invece sufficiente avere una garanzia sul massimo errore commesso, in termini relativi, si potrà far ricorso ad un algoritmo r-approssimato, ammesso che ve ne siano per il particolare problema in esame. Infatti, mentre certi metodi esatti (come i piani di Gomory) possono applicarsi, almeno in linea di principio, a qualsiasi problema di programmazione lineare a numeri interi, per quanto riguarda gli algoritmi approssimati non esiste, per ora, un approccio sistematico allo stesso livello di generalità.

Una strada diversa, che può diventare l'unica percorribile se le dimensioni e/o la complessità del problema sono elevate, se non è possibile pervenire ad un problema che ammetta dipendenza esplicita delle variabili decisionali con il funzionale da ottimizzare, quando un funzionale ha un andamento particolarmente "oscillatorio" (con un numero elevato di massimi o minimi locali) è quella che consiste nel cercare soluzioni di tipo euristico, ottenute applicando un algoritmo in grado di produrre una soluzione buone o sub-ottime in tempo relativamente breve.

Mentre chiaramente è possibile progettare euristiche specifiche per qualunque problema di ottimizzazione combinatoria, negli ultimi anni hanno acquisito importanza via via maggiore alcuni approcci euristici di tipo generale, detti metaeuristiche. La struttura e l'idea di fondo di ciascuna metaeuristica sono sostanzialmente fissate, ma la realizzazione delle varie componenti dell'algoritmo dipende dai singoli problemi. Spesso peraltro queste metaeuristiche traggono ispirazione (se non legittimazione) da alcune analogie con la natura fisica. Ad esempio, nel simulated annealing si paragona il processo di soluzione di un problema di ottimizzazione combinatoria, in cui si passa iterativamente da una soluzione ammissibile a un'altra, via via migliorando la funzione obiettivo, al processo con cui un solido, raffreddandosi, si porta in configurazioni con via via minore contenuto di energia. Negli algoritmi genetici, un insieme di soluzioni ammissibili viene visto come un insieme di individui di una popolazione che, accoppiandosi e combinandosi tra loro, danno vita a nuove soluzioni che, se generate in base a criteri di miglioramento della specie, possono risultare migliori di quelle da cui sono state generate. Gli approcci metaeuristici possono vedersi in realtà in modo omogeneo, come generalizzazioni di un unico approccio fondamentale, che è quello della ricerca locale. La ricerca locale si basa su quello che è, per certi versi, l'approccio più semplice e istintivo all'ottimizzazione: andare per tentativi. In effetti, l'idea di funzionamento della ricerca locale (come del resto quella delle varie metaeuristiche) è talmente elementare che è sorprendente constatare la loro relativa efficacia. Va tuttavia detto fin da ora che a fronte di questa relativa semplicità, l'applicazione di una qualunque metaeuristica a un problema di ottimizzazione combinatoria richiede comunque una messa a punto accurata e talvolta laboriosa.
Consideriamo un problema di minimizzazione, e una sua soluzione ammissibile x, con associato il valore della funzione obiettivo $f(x)$. La ricerca locale consiste nel definire un intorno di x (detto, nella terminologia della ricerca locale, vicinato), e nell'esplorarlo in cerca di soluzioni migliori, se ve ne sono. Se, in questo vicinato di x, si scopre una soluzione y per cui $f(y) < f(x)$, allora ci si sposta da x a y e si riparte da y con l'esplorazione del suo intorno. Se invece nel vicinato di x non si scopre nessuna soluzione migliore, allora vuol dire che x è un

minimo locale. Nella ricerca locale classica, arrivati in un minimo locale, l'algoritmo si ferma e restituisce questo minimo come valore di output. Ovviamente, non si ha nessuna garanzia in generale che tale valore costituisca una soluzione ottima del problema; anzi, tipicamente esso può essere anche molto distante dall'ottimo globale. Le metaeuristiche, in effetti, sono state concepite proprio per cercare di ovviare al problema di rimanere intrappolati in un minimo locale.
In base a quanto detto sopra, possiamo riassumere schematicamente l'algoritmo generale di ricerca locale come segue. Con x indichiamo una generica soluzione ammissibile del problema, $N(x)$ il suo vicinato, e $f(x)$ indica la funzione obiettivo.

Algoritmo di ricerca locale

1. Scegli una soluzione iniziale x;
2. Genera le soluzioni nel vicinato $N(x)$;
3. Se in $N(x)$ c'è una soluzione y tale che $f(y) < f(x)$, allora poni $x := y$ e vai a 2, altrimenti STOP.

Nella ricerca locale, come si vede, si ha un miglioramento della funzione obiettivo in corrispondenza di una mossa.

Per applicare l'approccio della ricerca locale a un particolare problema, bisogna fare alcune scelte di fondo.

- Occorre avere a disposizione una soluzione iniziale ammissibile. Questa può essere generata da un'euristica ad hoc per il particolare problema, o addirittura può essere generata casualmente. Risulta possibile anche eseguire l'algoritmo a partire da diverse soluzioni iniziali, ottenendo così diverse soluzioni euristiche, e scegliere poi la migliore.
- Bisogna definire in modo preciso e opportuno il vicinato di una soluzione.
- Definito il vicinato, occorre avere a disposizione un algoritmo efficiente per esplorarlo. Non è utile definire il vicinato in un modo teoricamente potente ma non essere in grado di esplorarlo in un tempo di calcolo ragionevole.
- Per evitare di incorrere in massimi/minimi locali è necessaria una procedura che possa permettere l'esplorazione di diverse parti dello spazio delle soluzioni.

A seconda di come vengono fatte queste scelte, l'algoritmo risultante può essere più o meno efficiente, e la soluzione proposta dalla ricerca locale risultare più o meno buona. Nel seguito presenteremo gli approcci seguiti per risolvere i problemi di ottimizzazione presentati.

2.5 Particle Swarm Optimization

Per risolvere il problema di ottimizzazione dei funzionali introdotti nel precedente capitolo, sono stati testati differenti approcci iterativi di ispirazione biologica di tipo population-based che appartengono al filone della Swarm Intelligence. La Swarm Intelligence (SI) è una famiglia di tecniche dell'intelligenza artificiale basata sullo studio di comportamenti collettivi in sistemi decentralizzati e auto organizzati. Essi sono costituiti da una popolazione di agenti i quali interagiscono tra di loro e con il loro ambiente e sono denominati particelle. Il motivo della denominazione particelle risiede nel fatto che la loro velocità e la loro accelerazione, quantità che misurano come le particelle si muovono nello spazio delle soluzioni, sono più appropriate ad esse [5]. Ciascuna particella rappresenta una possibile candidata soluzione ed ha associato un valore, chiamato fitness value, che indica la sua qualità. Ogni particella inoltre è caratterizzata da una posizione. La soluzione di un generico problema di ottimizzazione mediante questo approccio deriva dall'interazione sociale di queste particelle, in quanto riusciranno ad affrontare il problema solo come gruppo.

Il primo di tali algoritmi è la Particle Swarm Optimization (PSO), un approccio che si rifà al comportamento sociale degli stormi di uccelli o dei banchi di pesce. Rappresenta, nella sua versione base, una tecnica per la ricerca degli ottimi non vincolati. Inizialmente, la PSO viene creata da J. Kennedy e R. Eberhart nel 1995 [5] per simulare graficamente la coreografia degli stormi di uccelli e per scoprire le regole sottostanti al loro modo di volare in sincronia, cambiare direzione improvvisamente, disperdersi per poi riunirsi. Solo in un secondo momento è stata utilizzata come strumento per risolvere problemi di ottimizzazione. Le radici della PSO risiedono quindi nei processi naturali di gruppi di animali, in questo caso stormi di uccelli o banchi di pesce, e

ciò conduce l'algoritmo all'interno della Swarm Intelligence. In tale contesto ogni uccello facente parte di uno stormo deve perseguire in maniera ottimale la ricerca di cibo, la fuga dai predatori e così via. Per soddisfare questi obiettivi utilizza informazioni relative alla propria velocità e posizione e alle direzioni e velocità prese dalla popolazione. Inoltre, il volatile stabilisce il suo comportamento in base a quello degli altri, deve quindi seguire i propri vicini, evitare collisioni e restare nello stormo. Una singola particella, quando scorge una fonte di cibo (ovvero, una zona dello spazio in cui la soluzione può essere migliore) può agire secondo due differenti modalità. La prima è dettata dall'individualismo, cioè tenderà ad allontanarsi dal gruppo per raggiungerla; la seconda è dovuta alla socialità, ovvero tenderà a rimanere comunque all'interno del gruppo. Si può quindi distinguere una strategia di ricerca basata sull'esplorazione, legata all'individualismo del singolo, ed una basata sullo sfruttamento, connessa alla socialità e quindi all'utilizzo dei successi di altri individui. Nella PSO le due componenti convivono ed hanno due diversi pesi nello spostamento della singola particella.
Qualitativamente parlando, un numero prefissato di particelle sono inizialmente posizionate in modo casuale all'interno dello spazio delle soluzioni ammissibili; quest'ultima rappresenta un'area di ricerca della soluzione ottima.
Nella versione della PSO implementata lo spazio di ricerca non è composto da tutto R^n (dove n è il numero di variabili del problema) ma da un suo sottospazio, avendo incorporato i vincoli del problema sotto forma di bound entro i quali le variabili sono "forzate" a restare.
Ogni particella della popolazione esplora l'area di ricerca e determina il proprio successivo movimento considerando la propria posizione corrente, la propria migliore posizione passata, chiamata personal best fitness, e la migliore posizione passata dello sciame, detta global best fitness, il tutto scambiando queste informazioni con le altre particelle vicine del gruppo. Come detto precedentemente, la qualità di una posizione, possibile soluzione, viene definita dalla funzione di fitness. In questo modo si avrà che l'intero gruppo tenderà a dirigersi verso la migliore posizione individuata globalmente grazie all'interazione tra le particelle.

Bisogna tenere conto che il gruppo è caratterizzato da tre elementi:

- Robustness (robustezza): qualora una particella si allontana dallo stormo, il resto del gruppo esegue lo stesso il compito;
- Flexibility (flessibilità): si possono risolvere diversi problemi applicando le stesse regole d'interazione tra gli individui;
- Self-organization (auto-organizzazione): il comportamento articolato del gruppo è sorretto da regole semplici.

Da un punto di vista quantitativo, la PSO è caratterizzata da M particelle dove ognuna rappresenta una possibile soluzione, da N variabili del problema di ottimizzazione e dalla fitness function f: $S \in R^n \rightarrow R$ che individua la bontà della posizione dell'i- esima particella.

Ciascuna particella j è a sua volta caratterizzata all'iterazione t da:

- $\vec{x}(t)$: posizione attuale della j-esima particella al tempo t;
- $\vec{v}(t)$: velocità attuale della j-esima particella al tempo t;
- $\vec{p}(t)$: migliore posizione passata della j-esima particella;
- $\vec{g}(t)$: migliore posizione passata tra tutte le particelle dello sciame;
- $f(\vec{x}(t))$: valore di bontà della posizione attuale;

L'algoritmo PSO [5] è inizializzato con particelle posizionate casualmente e uniformemente in determinati bound delle variabili decisionali. Vengono assegnate anche le velocità iniziali e causalmente nel range $[v_{min}\ v_{max}]$. Successivamente la funzione obiettivo è valutata per ogni particella e la miglior funzione obiettivo e le relative variabili decisionali sono salvate. La velocità viene aggiornata considerando la velocità corrente ($\vec{v}(t)$) la posizione della migliore particella ($\vec{g}(t)$) e le migliori posizioni ($\vec{p}(t)$) relative ad ogni particella:

$$\vec{v}(t+1) = \omega\vec{v}(t) + \phi_1 rand(0,1)\left(\vec{p}(t) - \vec{x}(t)\right) + \phi_2 rand(0,1)(\vec{g}(t) - \vec{x}(t))$$

dove ω è il *inertial weight* e ϕ_1, ϕ_2 sono costanti che pesano $\vec{p}(t)$ e $\vec{g}(t)$.
Poi, iterativamente, ogni particella viene spostata in una nuova posizione calcolata in base alla velocità aggiornata ad ogni passo di tempo t. La posizione di ogni particella viene aggiornata:

$$\vec{x}(t+1) = \vec{x}(t) + \vec{v}(t+1)$$

Le iterazioni procedono fino a quando l'algoritmo raggiunge un criterio di arresto. Di seguito i passi salienti dell'algoritmo PSO.

Algoritmo PSO (pseudo-code)

1:Inizializzare particelle
2:**ripetere**
3: Calcolare la funzione obiettivo per ogni particella.
4: Modificare le migliori particelle.
5: Scegliere la particella migliore.
6: Calcolare le velocità delle particelle.
7: Aggiornare posizioni delle particelle.
8:fino a quando i criteri di arresto non sono soddisfatti.

2.6 Multi-Objective Particle Swarm Optimization

Multi-objective PSO è un'estensione multi-obiettivo dell'algoritmo PSO. In particolare, MPSO incorpora, nell'algoritmo standard PSO, il fronte di Pareto al fine di gestire problemi con diverse funzioni obiettivo. L'idea di questo algoritmo nell'utilizzare un repository di particelle che viene poi utilizzato da altre particelle per guidare il proprio volo. Di seguito i passi salienti dell'algoritmo MPSO [6], [7].

Algoritmo MPSO (pseudo-code)

1: Inizializzare posizioni e velocità particelle
2: **ripetere**
3: Calcolare la funzione obiettivo per ogni particella.
4: Salvare le posizioni delle particelle che rappresentano i vettori non dominanti nel repository.
5: Inizializzare la memoria di ogni particella e salvare le migliori posizioni di ogni particella.
6: Calcolare le velocità delle particelle.
7: Aggiornare posizioni delle particelle.
8: Calcolare la funzione obiettivo per ogni particella.
9: Aggiornare il repository.
10: Salvare le migliori posizioni di ogni particella e aggiornare la memoria.
11:fino a quando i criteri di arresto non sono soddisfatti.

2.7 Artificial Bee Colony

L'Artificial Bee Colony è un algoritmo meta-euristico basato sugli sciami, presentato da [8] e ispirato al modello di approvvigionamento delle colonie di api [10].
È stato presentato come un algoritmo per l'ottimizzazione numerica, ma può essere utilizzato anche per problemi di natura combinatoria, vincolata e non. Il modello delle colonie di api consiste di tre componenti fondamentali: food sources, employed bees e unemployed bees. Nel sistema colonia, la qualità di una risorsa

alimentare dipende da diversi fattori, come la distanza dall'alveare, la quantità di cibo presente e la facilità di estrazione; a ciascuna risorsa è assegnata un'ape il cui compito è quello di memorizzare le informazioni relative alla risorsa (distanza, direzione, sfruttabilità). Raccolte queste informazioni, le employed bees condividono le informazioni raccolte con le unemployed bees, con una probabilità che è proporzionale alla bontà della risorsa stessa. La comunicazione delle informazioni avviene tramite la waggle dance e con questa danza, l'ape definisce distanza (durata della danza) e direzione della risorsa (calcolando l'angolo che questa forma con la direzione del sole).
Maggiore è la quantità di informazioni relative a una risorsa che vengono condivise nell'alveare, maggiore è la possibilità che le onlooker bees siano reclutate per il suo sfruttamento. Il sistema prevede inoltre che quando una risorsa alimentare si estingue, essa potrà essere abbandonata. In questo caso l'employed bee ad essa associata si trasformerà in scout bee e si dedicherà alla ricerca di una risorsa nelle vicinanze dell'alveare.

Definiamo lo spazio delle soluzioni come lo spazio entro cui si muovono le api artificiali: una soluzione sarà associata ad una fonte di cibo, la cui bontà sarà quantificabile dalla funzione di fitness del problema da ottimizzare. La colonia contiene tre tipi di api: employed bees, unemployed bees e scout bees.

In una fase iniziale, le scout bees vengono inviate randomicamente alla ricerca di nuove risorse. Successivamente, le employed bees e le onlooker bees proseguono l'opera di sfruttamento, provando a migliorare le soluzioni note cercando tra quelle vicine. Si opera quindi una greedy selection tra la soluzione nota e la nuova soluzione proposta. Se la nuova soluzione è accettata, essa sostituirà la precedente, altrimenti si invecchierà la soluzione esistente. Di seguito i passi salienti dell'algoritmo ABC [11].

Algoritmo ABC (pseudo-code)

1:Inizializzare popolazione
2:**ripetere**
3: Employed Bees Phase: ogni ape cerca di migliorare la soluzione ad essa associata, effettuando una ricerca tra le soluzioni confinanti.
4: Onlooker Bees Phase: le onlooker bees selezionano le risorse da sfruttare proporzionalmente ai loro valori di fitness, tentando a loro volta di migliorarle.
5: Scout Bees Phase: le api associate alle soluzioni che corrispondono a risorse esauste (ad es. dopo diversi tentativi di miglioramento falliti) diventano scout bees generando nuove soluzioni.
6: Memorizzare le migliori risorse trovate.
7:fino a quando i criteri di arresto non sono soddisfatti.

2.8 Risultati delle simulazioni

In questa annualità, gli algoritmi di ottimizzazione sono stati analizzati e testati sull'edificio F40 nell'ipotesi che questo sia dotato di pompa di calore, impianto PV da 20 kWp e storage elettrico con capacità pari a 40 kWh. Le specifiche della potenza dell'impianto e della capacità dello storage sono state definite insieme con ENEA. In particolare, la taglia del fotovoltaico è stata definita dopo aver misurato la superficie realmente disponibile per un'installazione reale.

Di seguito vengono riportati i dati utilizzati nelle simulazioni.

Dati utilizzati climatici e di set-point di edificio al fine dell'ottimizzazione:
Set-point invernale nei locali riscaldati: lower bound 18°C, upper bound 22°C.
Temperatura di mandata invernale: lower bound 55°C, upper bound 70°C.
Set-point estivo nei locali raffrescati: lower bound 24°C, upper bound 28°C.
Temperatura di mandata estiva: lower bound 10°C, upper bound 14°C.

Dati climatici: dati climatici dell'anno medio della centralina METEONORM di Ciampino. Pur avendo a disposizione un anno reale per il C.R. Casaccia (2013), si è preferito usare l'anno medio della centralina Meteonorm di Roma Ciampino, in quanto rappresentativo di un periodo più lungo (10 anni) e quindi meno soggetto alle singolarità dei singoli anni.

L'edificio F40 è stato simulato considerando gli apporti gratuiti delle apparecchiature elettriche e delle presenze nei seguenti periodi:

Periodo invernale:
14-Gen-2013 – 27-Gen-2013

Periodo estivo:
14-Jul-2013 – 27-Jul-2013

Dati impianto fotovoltaico:
Potenza di picco:20 kWp

Dati storage elettrico:
Capacità: 40 kWh
Profondità di scarica: 70%
Efficienza carica e scarica: 95%

Parametri algoritmi di ottimizzazione:
24 particles
50 iterazioni

Test effettuati
Di seguito vengono riportati i risultati delle simulazioni. In particolare, vengono presentati i risultati del PSO considerando nell'ordine i seguenti funzionali:

- Massimizzare l'autonomia dalla rete/minimizzazione assorbimento.
- Minimizzare la spesa economica utilizzando il flat pricing.
- Minimizzare la spesa economica utilizzando il dynamic pricing.
- Massimizzare il comfort utilizzando l'indice PMV
- Massimizzare la percentuale di autoconsumo da fonti rinnovabili
- Minimizzazione della spesa economica utilizzando il dynamic pricing e ottimizzare il comfort attraverso il PMV

Per ciascuno di questi scenari verranno presentati i risultati sia in estate che in inverno di due settimane.
Il MPSO è stato utilizzato per ottimizzare due funzionali contrastanti che sono il comfort in termini di PMV e la spesa energetica. I risultati sono stati confrontati con il caso senza ottimizzazione considerando una giornata invernale e una giornata estiva.
Infine il PSO è stato confrontato con l'algoritmo ABC nella settimana invernale considerando come funzionale la spesa energetica.

2.8.1 Massimizzare l'autonomia dalla rete

Inverno

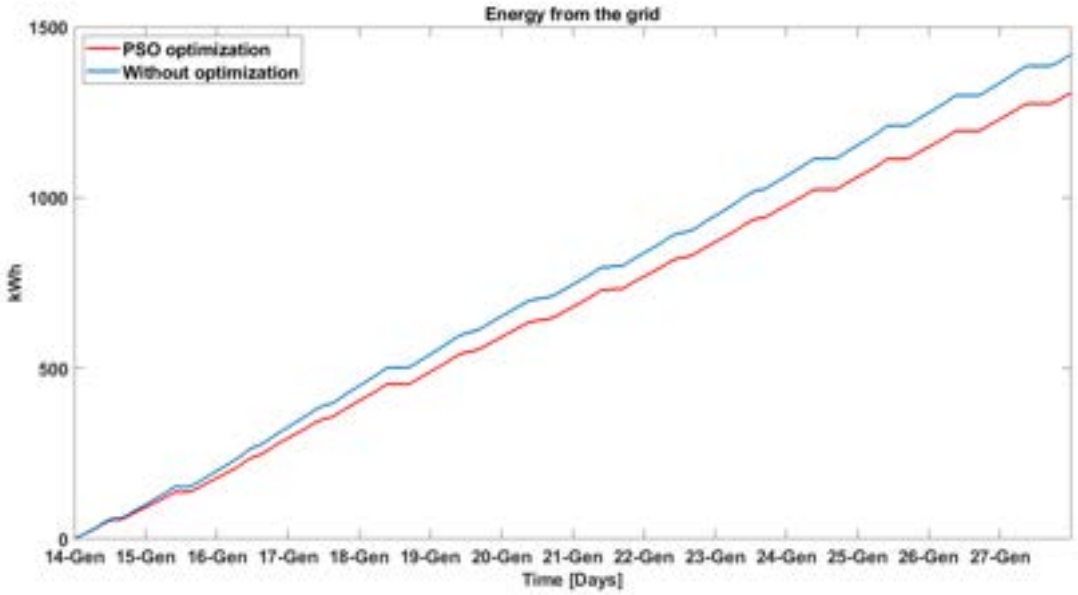

Figura 2-3 Energia prelevata dalla rete (energia elettrica consumata meno energia PV) nel periodo 14/01-27/01

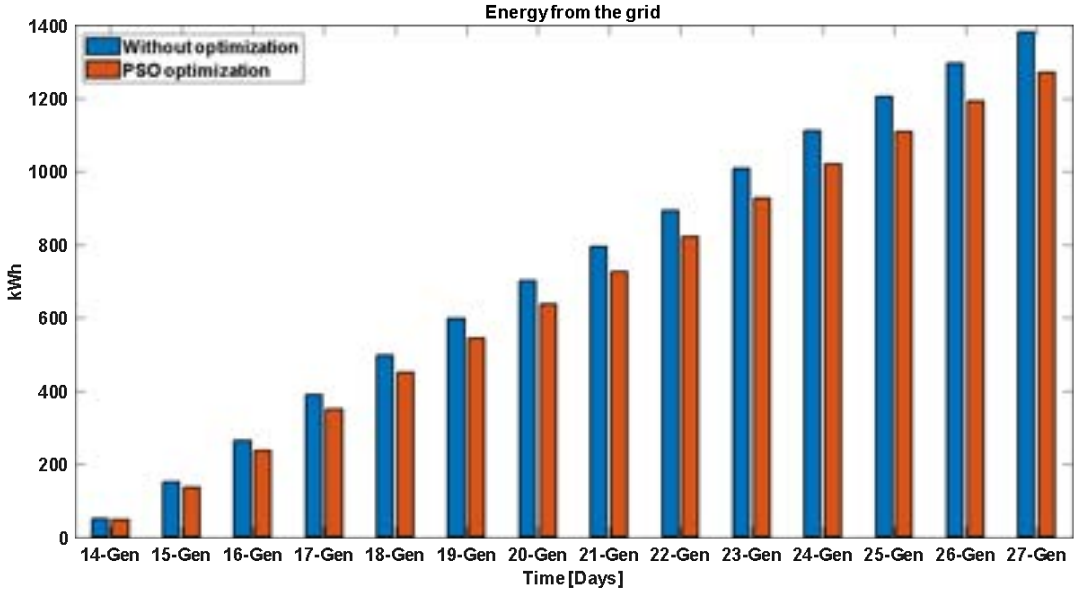

Figura 2-4 Media giornaliera dell'energia prelevata dalla rete (energia elettrica consumata meno energia PV) nel periodo 14/01-27/01

La Figura 2-3 mostra l'energia prelevata dalla rete nel periodo invernale, mentre la Figura 2-4 mostra la media giornaliera dell'energia prelevata dalla rete. L'algoritmo di ottimizzazione permette di risparmiare 113 kWh nelle due settimane di ottimizzazione.

Estate

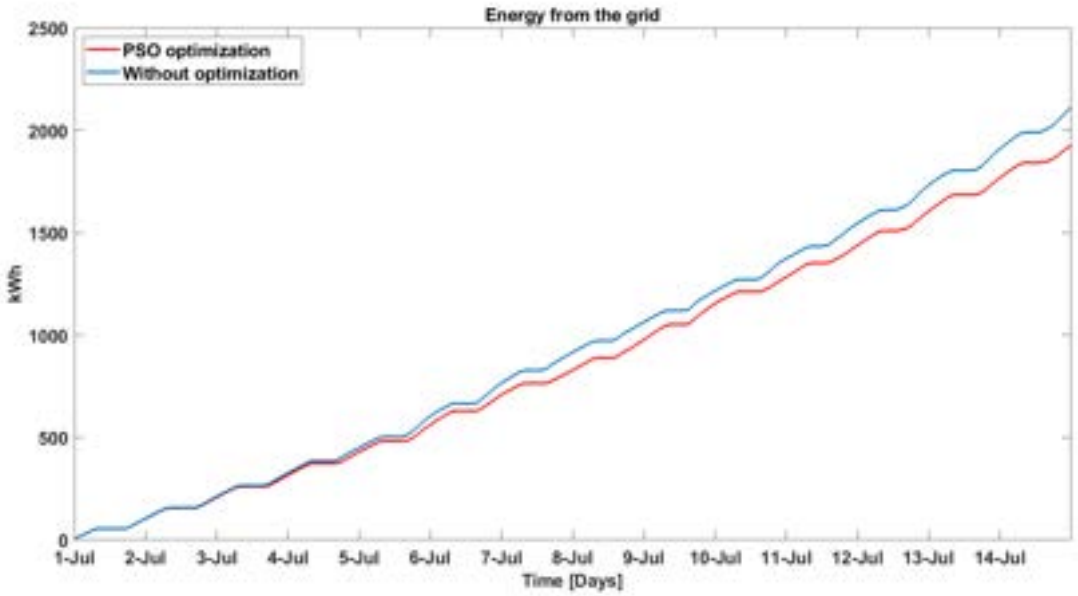

Figura 2-5 Energia prelevata dalla rete (energia elettrica consumata meno energia PV) nel periodo 01/07-14/07

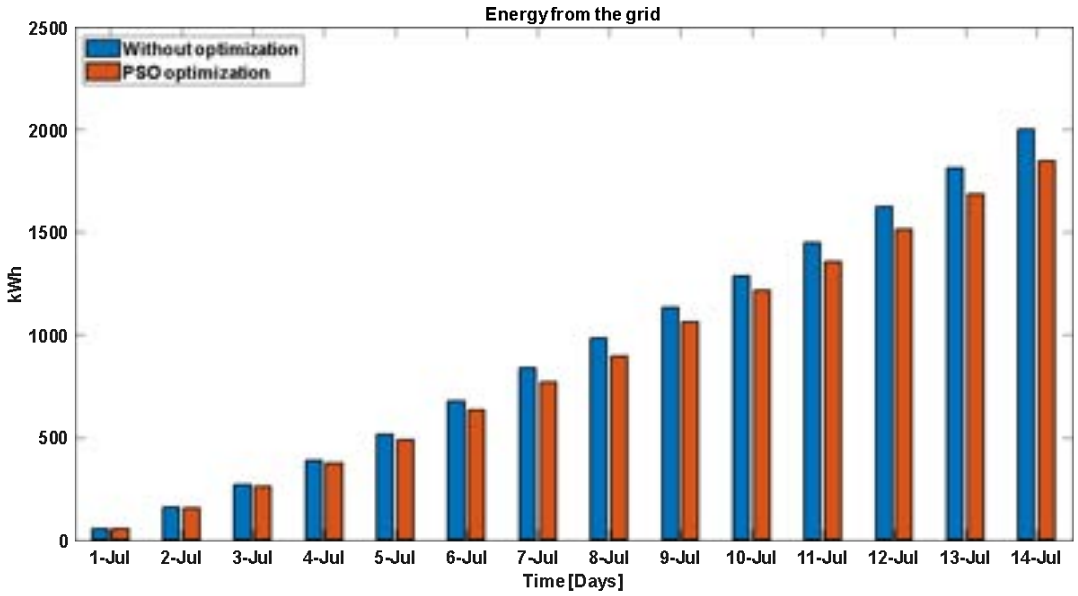

Figura 2-6 Media giornaliera dell'energia prelevata dalla rete (energia elettrica consumata meno energia PV) nel periodo 01/07-14/07

La Figura 2-5 mostra l'energia prelevata dalla rete nel periodo estivo, mentre la Figura 2-6 mostra la media giornaliera dell'energia prelevata dalla rete. L'algoritmo di ottimizzazione permette di risparmiare 84 kWh nelle due settimane di ottimizzazione.

2.8.2 Minimizzare la spesa economica: flat pricing

Inverno

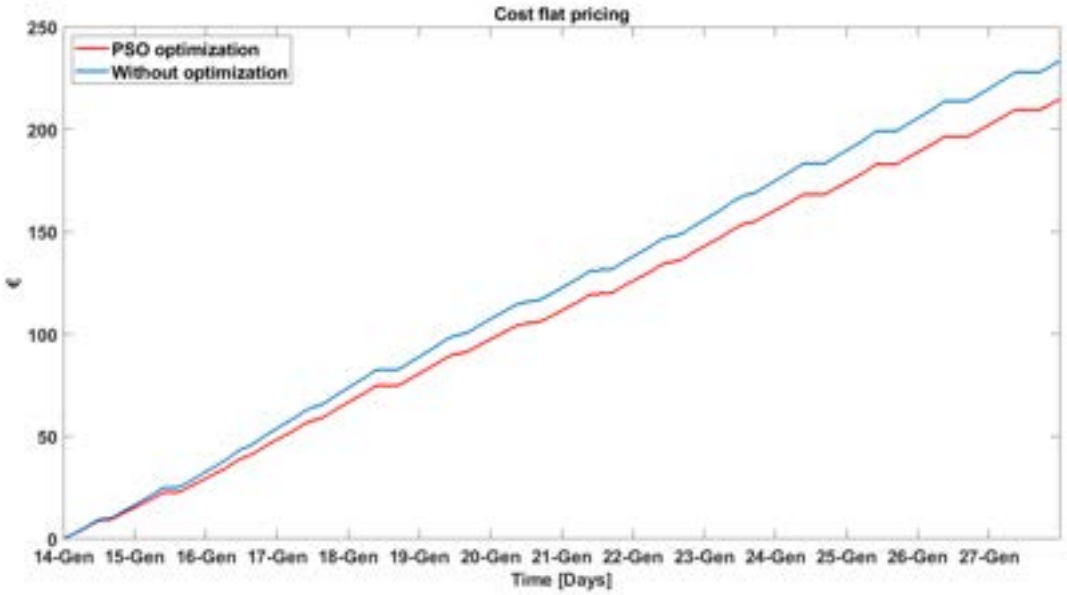

Figura 2-7 Flat pricing con PV nel periodo 14/01-27/01

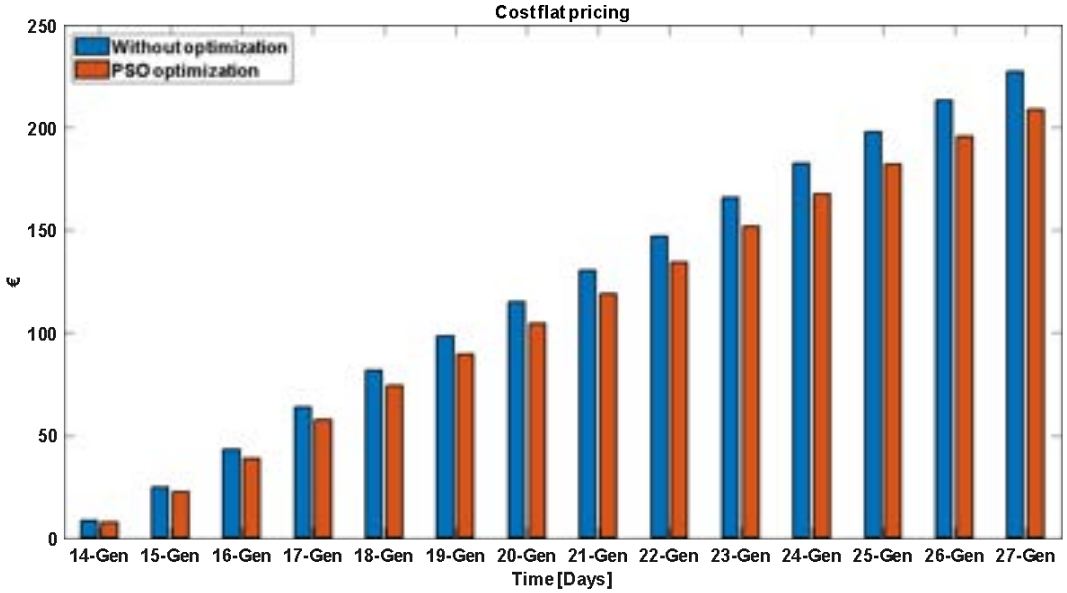

Figura 2-8 Media giornaliera del flat pricing con PV nel periodo 14/01-27/01

La Figura 2-7 mostra la spesa energetica considerando il flat pricing nel periodo invernale, mentre la Figura 2-8 mostra la media giornaliera della spesa energetica considerando il flat pricing. L'algoritmo di ottimizzazione permette di risparmiare 19 € nelle due settimane di ottimizzazione.

Estate

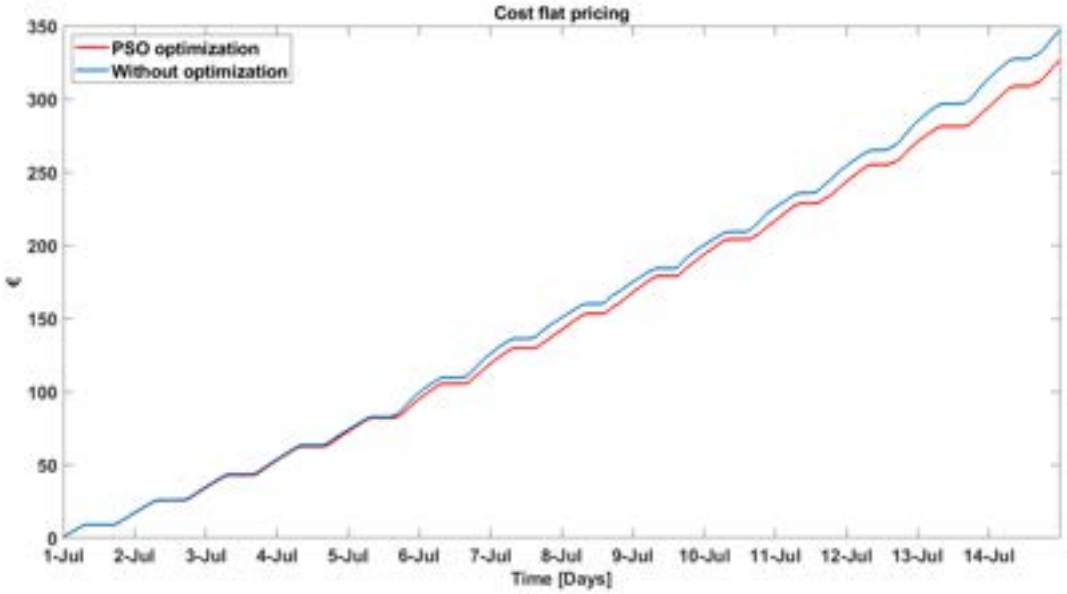

Figura 2-9 Flat pricing con PV nel periodo 01/07-14/07

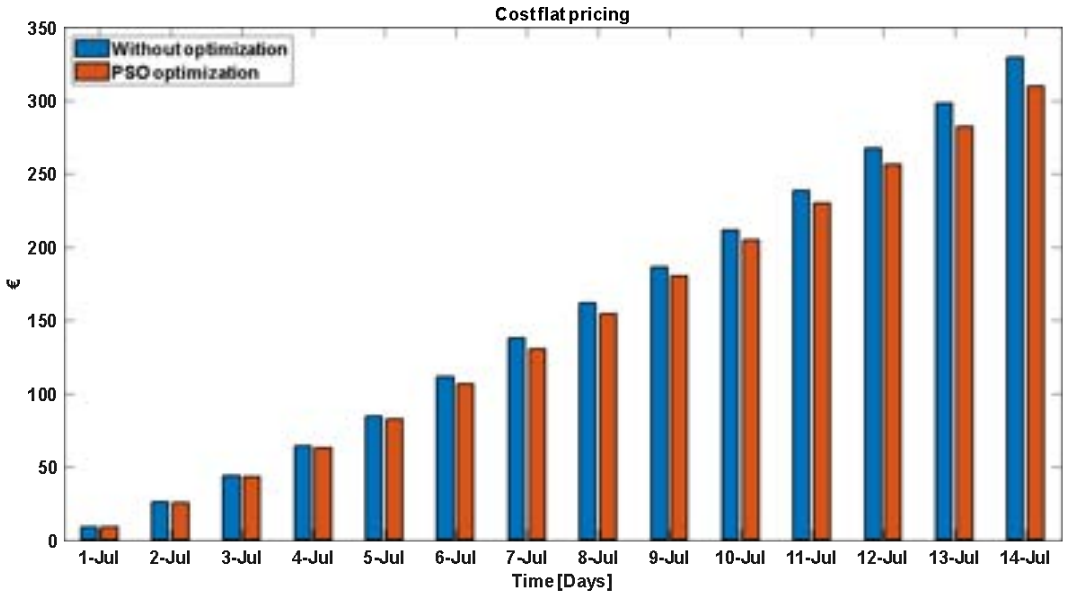

Figura 2-10 Media giornaliera del flat pricing con PV nel periodo 01/07-14/07

La Figura 2-9 mostra la spesa energetica considerando il flat pricing nel periodo estivo, mentre la Figura 2-10 mostra la media giornaliera della spesa energetica considerando il flat pricing. L'algoritmo di ottimizzazione permette di risparmiare 21 € nelle due settimane di ottimizzazione.

2.8.3 Minimizzare la spesa economica: dynamic pricing

Inverno

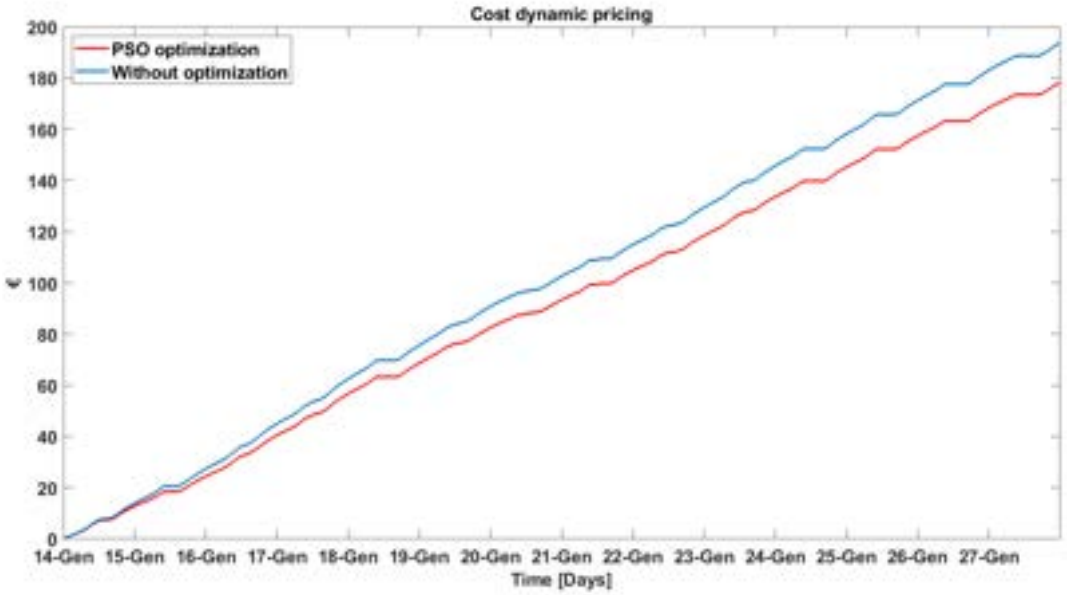

Figura 2-11 Dynamic pricing con PV nel periodo 14/01-27/01

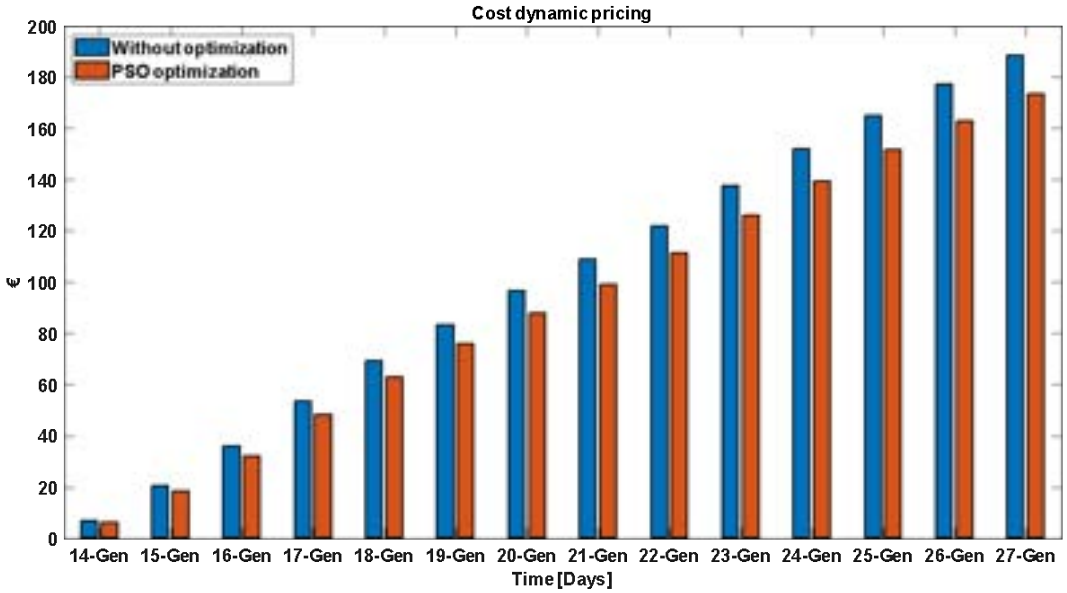

Figura 2-12 Media giornaliera del dynamic pricing con PV nel periodo 14/01-27/01

La Figura 2-11 mostra la spesa energetica considerando il dynamic pricing nel periodo invernale, mentre la Figura 2-12 mostra la media giornaliera della spesa energetica considerando il dynamic pricing. L'algoritmo di ottimizzazione permette di risparmiare 16 € nelle due settimane di ottimizzazione.

Estate

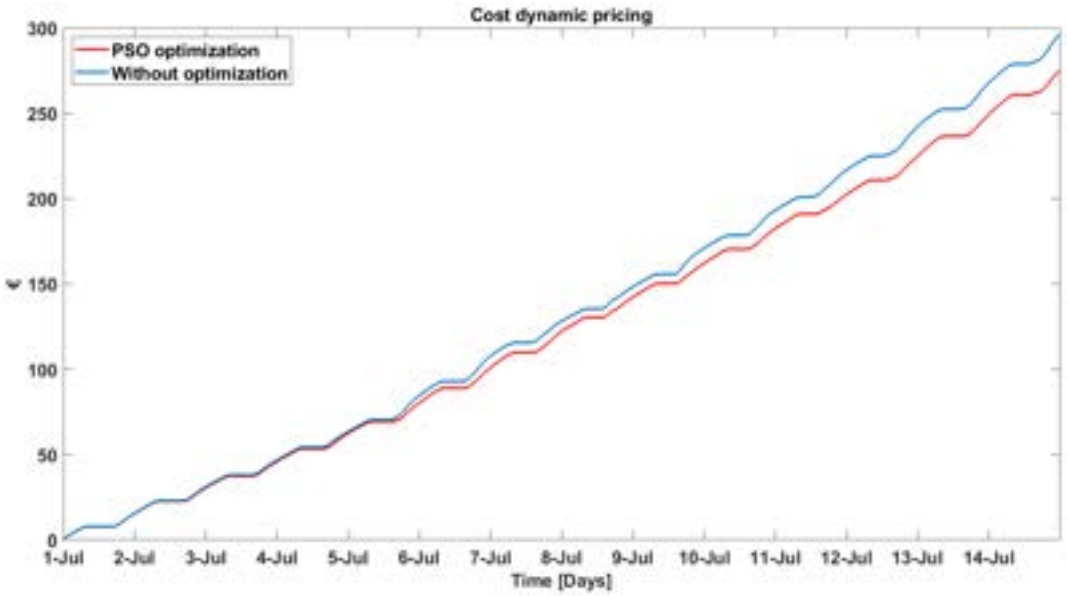

Figura 2-13 Dynamic pricing con PV nel periodo 01/07-14/07

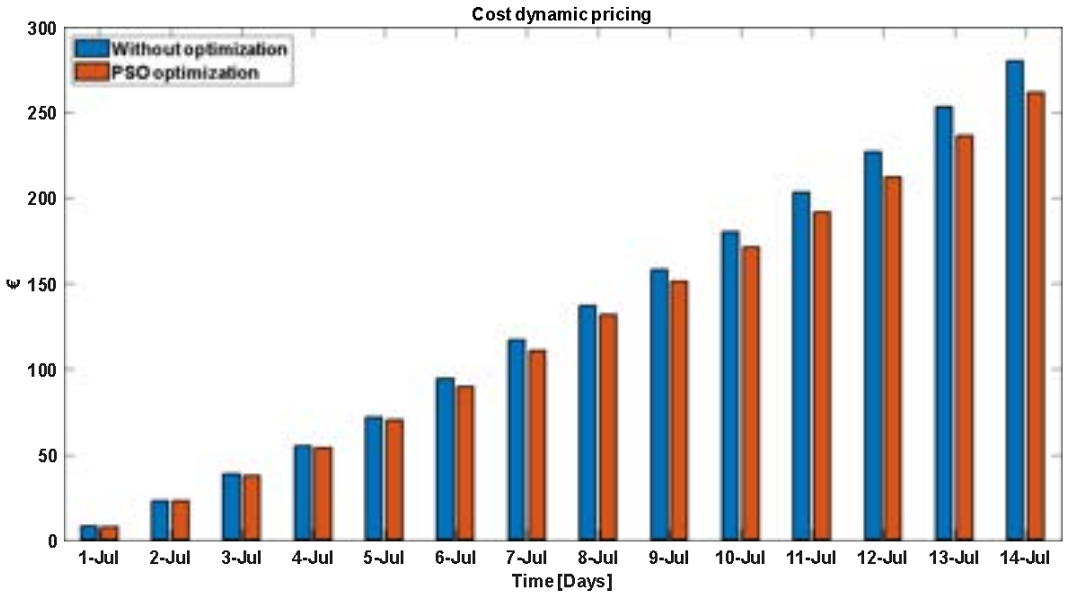

Figura 2-14 Media giornaliera del dynamic pricing con PV nel periodo 01/07-14/07

La Figura 2-13 mostra la spesa energetica considerando il flat pricing nel periodo estivo, mentre la Figura 2-14 mostra la media giornaliera della spesa energetica considerando il flat pricing. L'algoritmo di ottimizzazione permette di risparmiare 22 € nelle due settimane di ottimizzazione.

2.8.4 Minimizzare l'energia in eccesso reimmessa in rete

Inverno

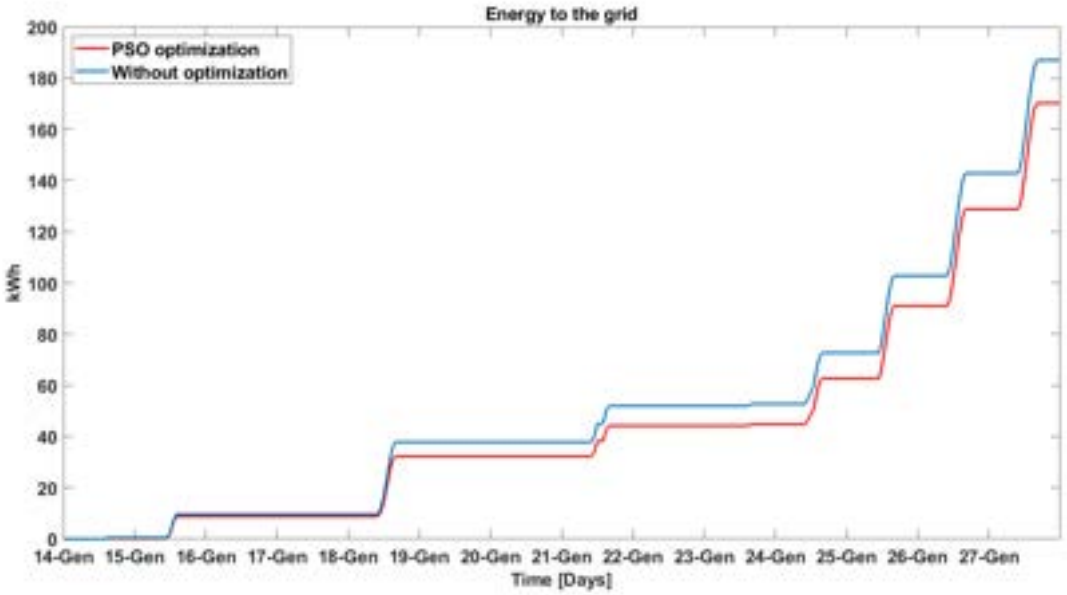

Figura 2-15 Energia in eccesso reimmessa in rete considerando il PV nel periodo 14/01-27/01

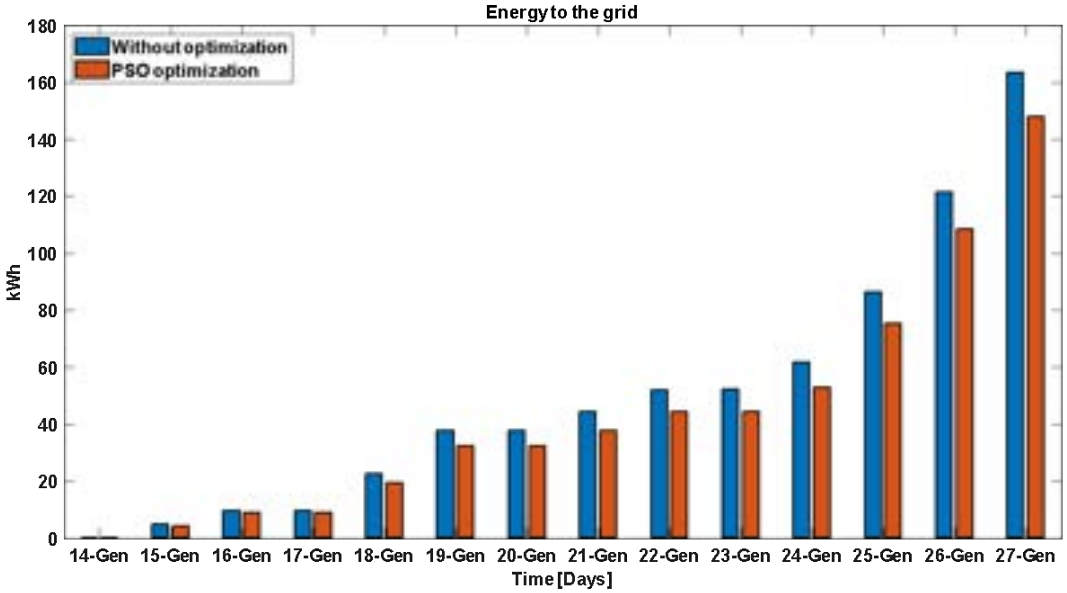

Figura 2-16 Media giornaliera dell'energia in eccesso reimmessa in rete considerando il PV nel periodo 14/01-27/01

La Figura 2-15 mostra l'energia in eccesso reimmessa in rete considerando il PV nel periodo invernale, mentre la Figura 2-16 mostra la media giornaliera dell'energia in eccesso reimmessa in rete considerando il PV. L'algoritmo di ottimizzazione permette di risparmiare minimizzare l'energia reimmessa in rete di 26.8 kWh nelle due settimane di ottimizzazione.

Estate

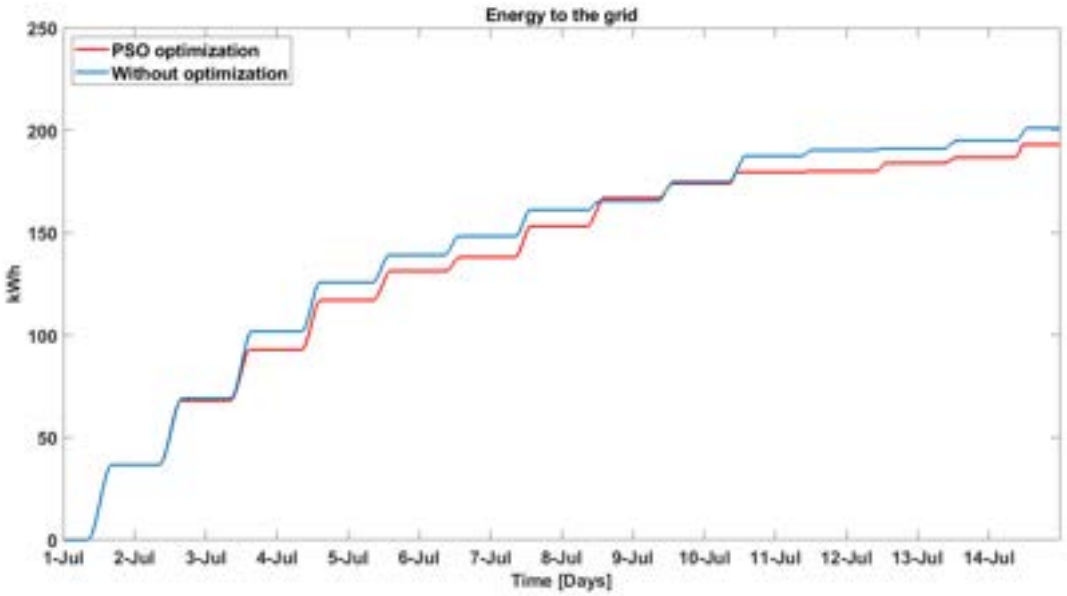

Figura 2-17 Energia in eccesso reimmessa in rete considerando il PV nel periodo 01/07-14/07

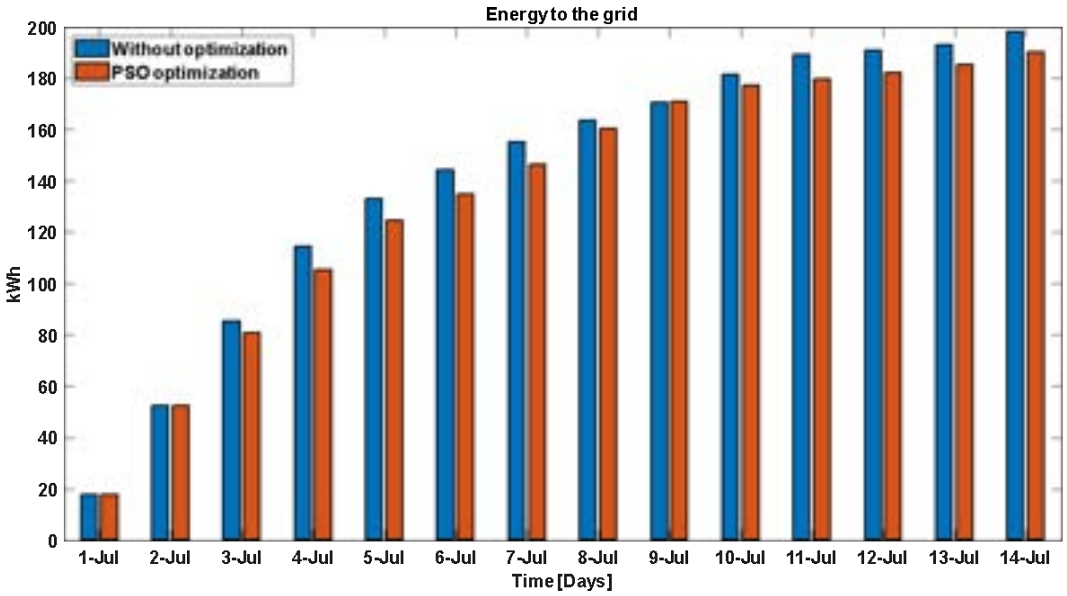

Figura 2-18 Media giornaliera dell'energia in eccesso reimmessa in rete considerando il PV nel periodo 01/07-14/01

La Figura 2-17 mostra l'energia in eccesso reimmessa in rete considerando il PV nel periodo estivo, mentre la Figura 2-18 mostra la media giornaliera dell'energia in eccesso reimmessa in rete considerando il PV. L'algoritmo di ottimizzazione permette di risparmiare minimizzare l'energia reimmessa in rete di 8 kWh nelle due settimane di ottimizzazione.

2.8.5 Massimizzare il comfort: PMV

Inverno

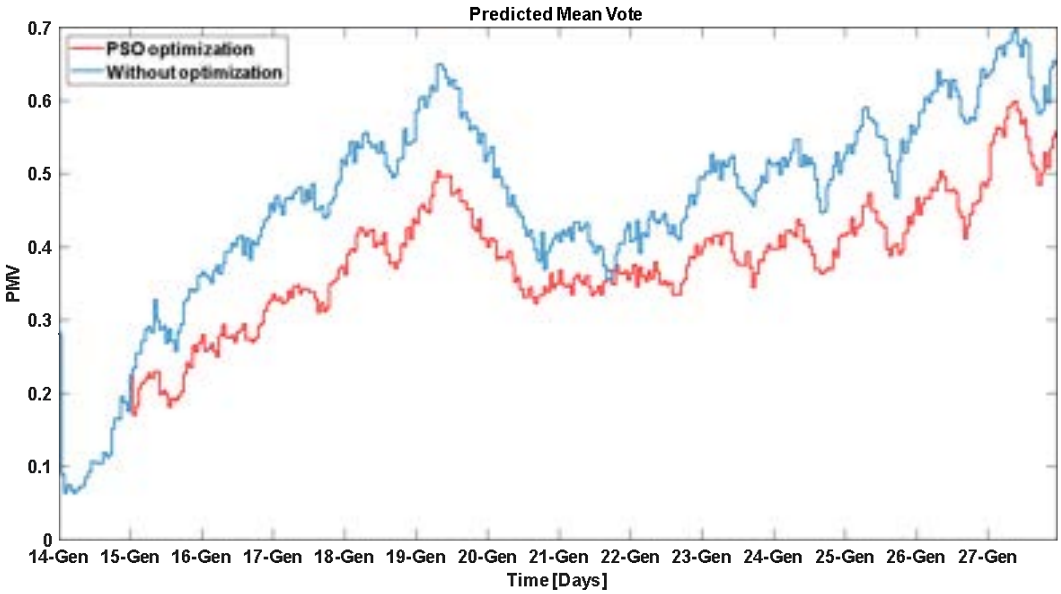

Figura 2-19 Predicted Mean Vote (PMV) nel periodo 14/01-27/01

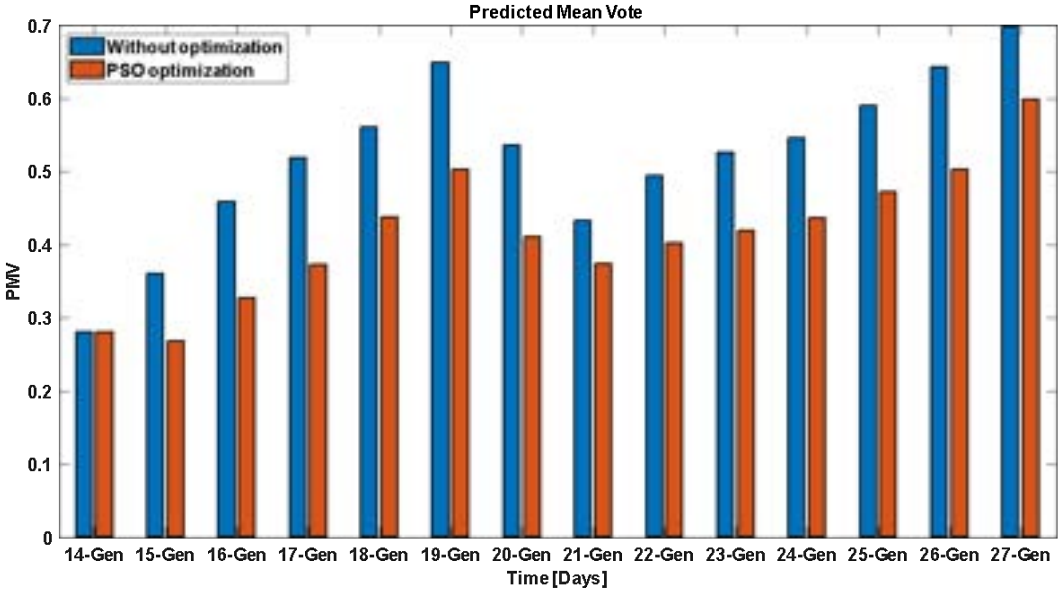

Figura 2-20 Media giornaliera del Predicted Mean Vote (PMV) nel periodo 14/01-27/01

La Figura 2-19 mostra il PMV nelle due settimane invernali, mentre la Figura 2-20 mostra la media giornaliera dello stesso indice. Mediamente il PSO permette di minimizzare mediamente il PMV, e quindi massimizzare il comfort, del 21% rispetto al caso senza ottimizzazione nelle due settimane considerate.

Estate

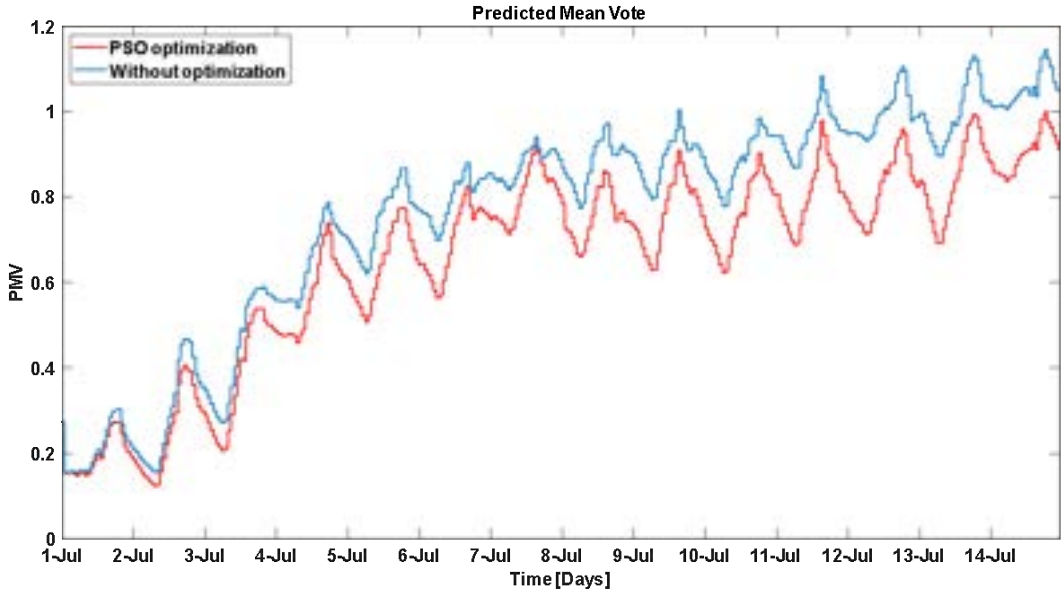

Figura 2-21 Predicted Mean Vote (PMV) nel periodo 01/07-01/07

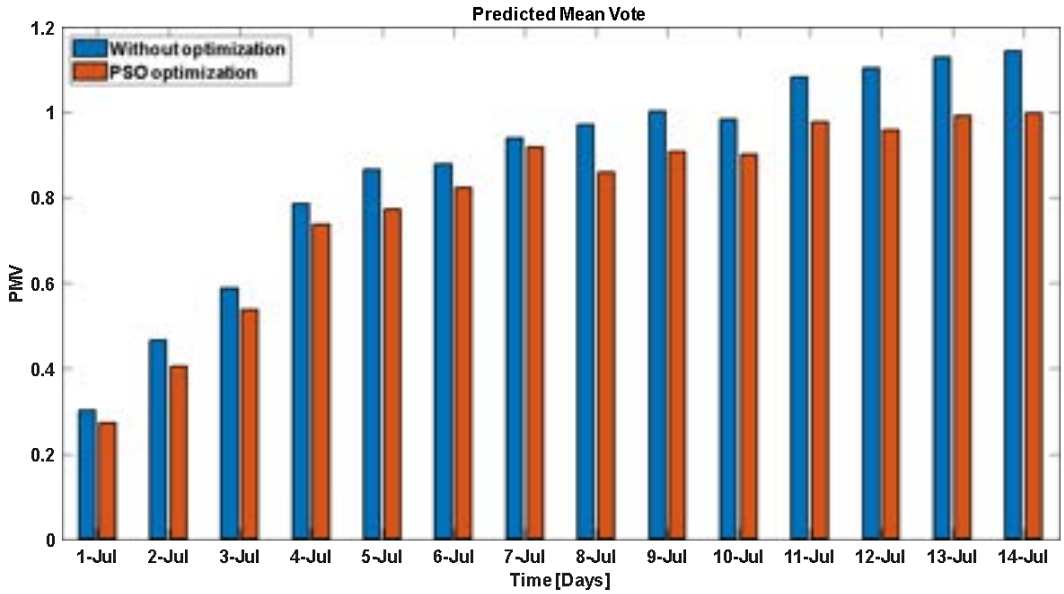

Figura 2-22 Media giornaliera del Predicted Mean Vote (PMV) nel periodo 01/07-14/07

La Figura 2-21 mostra il PMV nelle due settimane estive, mentre la Figura 2-22 mostra la media giornaliera dello stesso indice. Mediamente il PSO permette di minimizzare mediamente il PMV, e quindi massimizzare il comfort, del 6% rispetto al caso senza ottimizzazione nelle due settimane considerate.

2.8.6 Massimizzare la percentuale di autoconsumo da fonti rinnovabili

Inverno

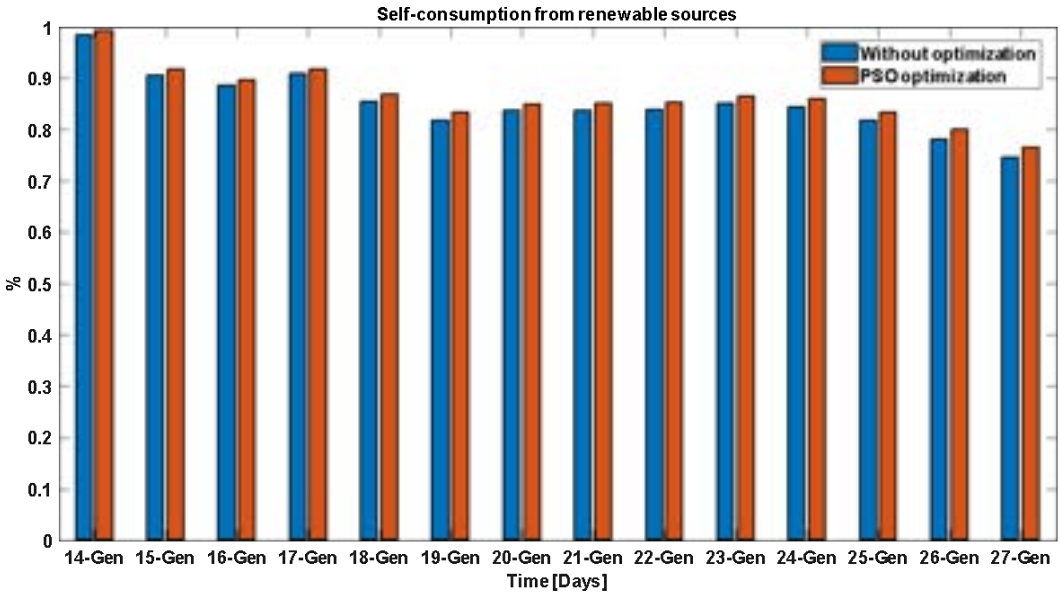

Figura 2-23 Percentuale di autoconsumo da fonti rinnovabili (PV) nel periodo 14/01-27/01

Estate

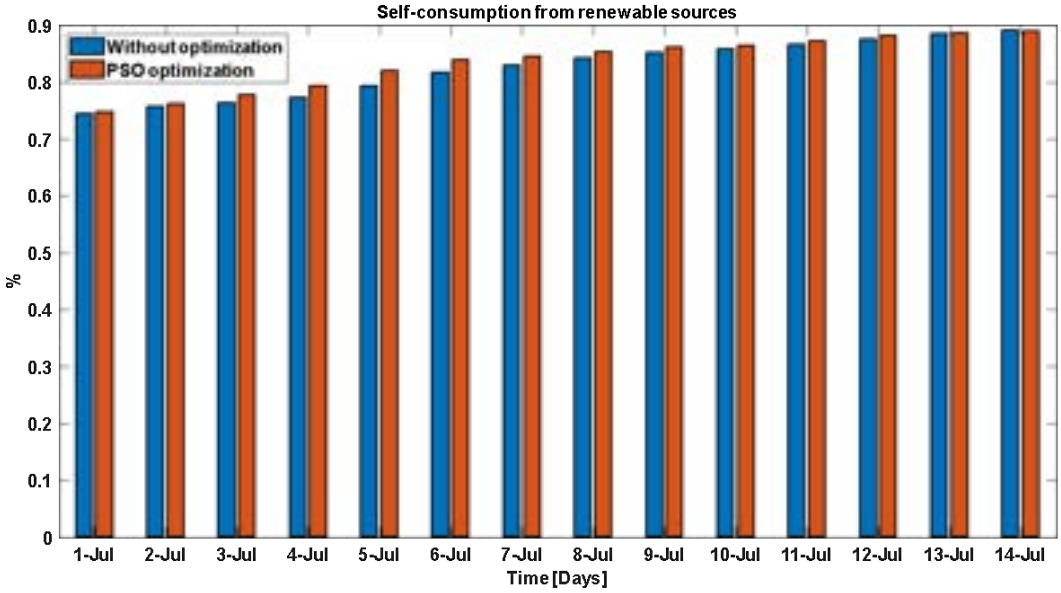

Figura 2-24 Percentuale di autoconsumo da fonti rinnovabili (PV) nel periodo 01/07-14/07

La Figura 2-23 e la Figura 2-24 mostrano la percentuale di autoconsumo da fonti rinnovabili, in questo caso il PV, nel periodo invernale ed estivo, rispettivamente. Nel caso invernale l'ottimizzazione permette di incrementare nelle due settimane l'autoconsumo dell'1,5% mentre nel caso estivo dell'1,2%.

2.8.7 Ottimizzare il comfort e la spesa energetica (MPSO)

Inverno

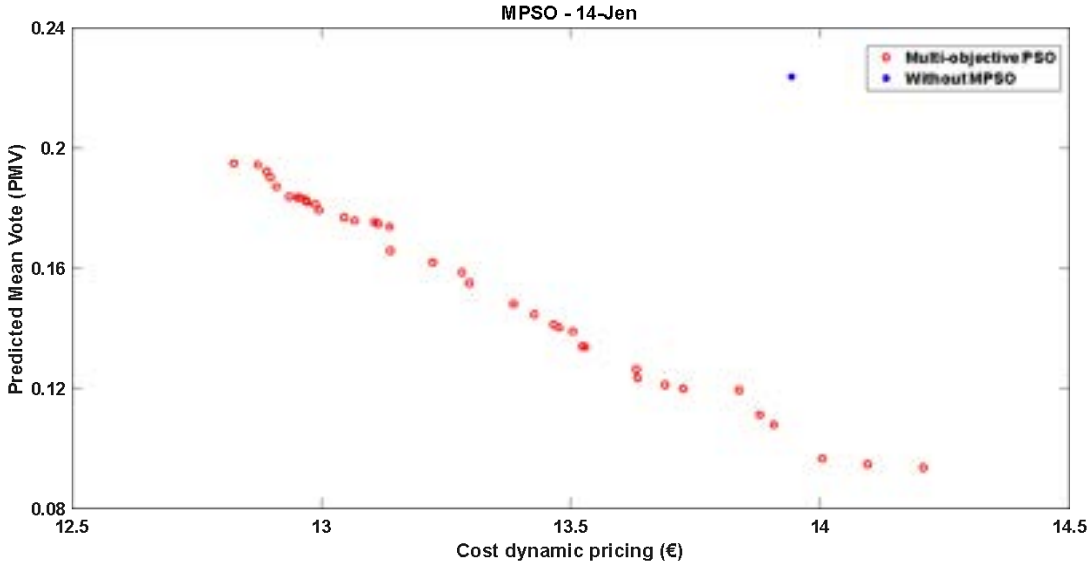

Figura 2-25 Punti non dominanti (in rosso) dell'algoritmo MPSO considerando il PMV e la spesa energetica in termini di dynamic pricing, nella giornata 14/01

Estate

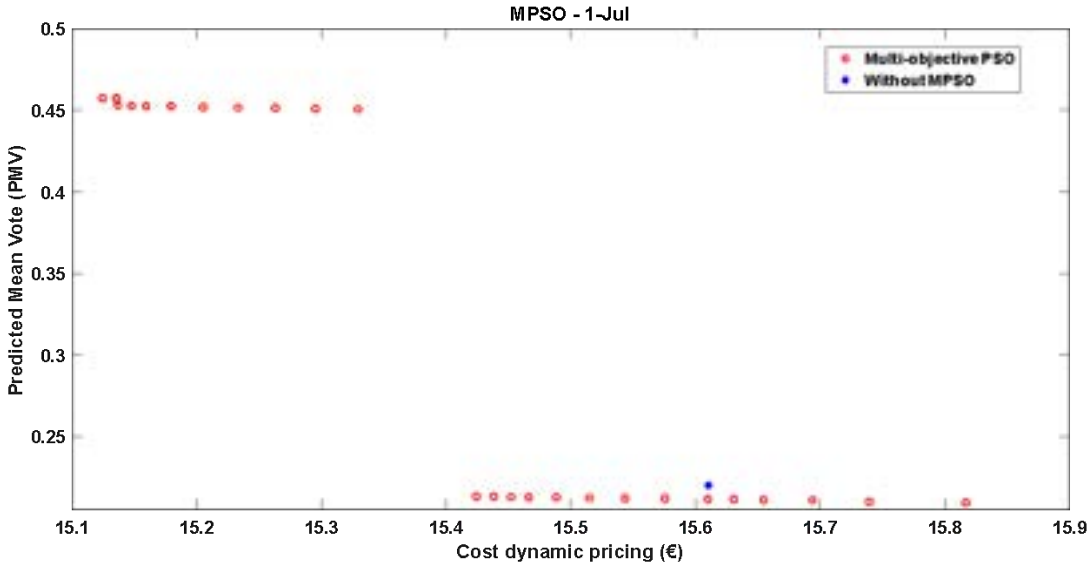

Figura 2-26 Punti non dominanti (in rosso) dell'algoritmo MPSO considerando il PMV e la spesa energetica in termini di dynamic pricing, nella giornata 01/07

La Figura 2-25 e la Figura 2-26 mostrano i risultati dell'algoritmo MPSO considerando come funzioni obiettivo il PMV (asse y) e la spesa energetica con il dynamic pricing (asse x) per i giorni 14/01 e 01/07 rispettivamente. In rosso sono mostrati i punti non dominati del fronte di Pareto mentre in blu il risultato della simulazione senza ottimizzazione. Si può notare come in entrambe le giornate l'algoritmo MPSO riesce a trovare punti ottimi migliori del caso senza ottimizzazione. Ciò è più evidente nel caso invernale dove a parità di spesa energetica è possibile minimizzare il PMV e quindi migliorare il comfort come evidenziato anche nella Figura 2-19.

2.8.8 Confronto con Artificial Bee Colony

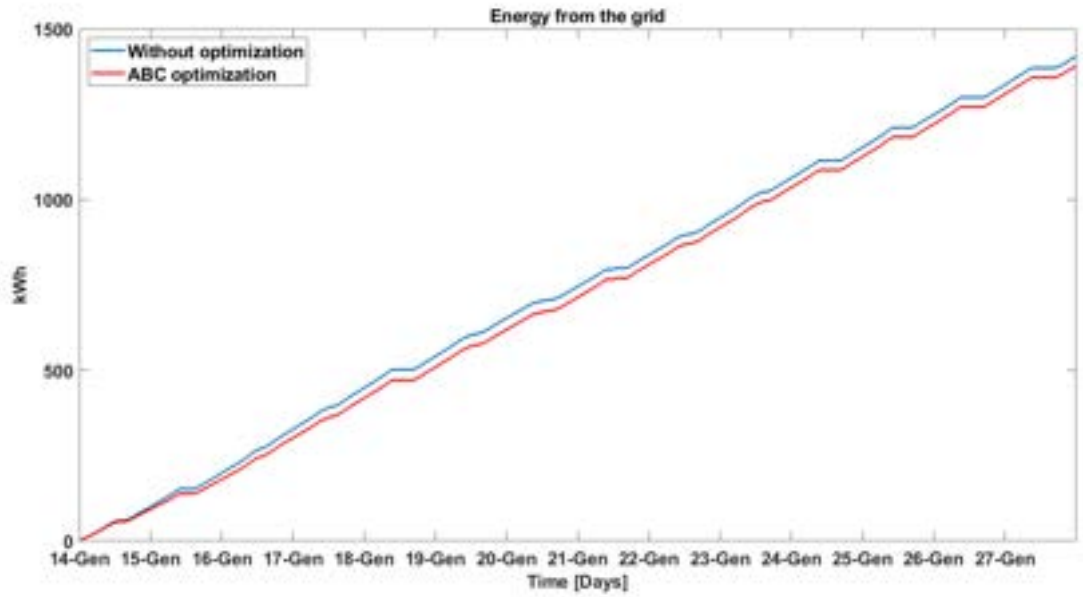

Figura 2-27 Ottimizzazione dell'energia prelevata dalla rete (energia elettrica consumata meno energia PV) nel periodo 14/01-27/01 utilizzando l'algoritmo ABC

La Figura 2-27 mostra l'ottimizzazione dell'energia prelevata dalla rete (energia elettrica consumata meno energia PV) nel periodo invernale utilizzando l'algoritmo ABC. L'ottimizzazione con l'algoritmo ABC permette un risparmio energetico di 20 kWh nelle due settimane invernali. Confrontando questo risultato con quello mostrati in Figura 2-3, l'algoritmo ABC presenta prestazioni peggiori in questo caso rispetto l'algoritmo PSO che nelle stesse condizioni permette di risparmiare 113 kWh.

I risultati ottenuti mostrano dimostrano come l'utilizzo di uno strumento di ottimizzazione dell'edificio possa contribuire a sfruttare al massimo l'autoconsumo di energia prodotta dall'impianto rinnovabile e/o a minimizzare la spesa economica e/o massimizzare il comfort dell'edificio. Effettivamente, in tutti gli scenari esaminati, l'algoritmo di ottimizzazione scelto, ovvero il Particle Swarm Optimization, ha mostrato un netto miglioramento delle prestazioni rispetto al caso in cui non venga utilizzato. Le prestazione dell'algoritmo di ottimizzazione sono risultate efficace in tutti i problemi considerati ovvero per quanto riguarda il problema di massimizzare l'autonomia dalla rete/minimizzazione assorbimento,il problema di minimizzare la spesa economica utilizzando il flat pricing, oppure di minimizzare la spesa economica utilizzando il dynamic pricing, così come nel problema di massimizzare il comfort utilizzando l'indice PMV, oppure massimizzare la percentuale di autoconsumo da fonti rinnovabili, e anche nel caso di minimizzazione della spesa economica utilizzando il dynamic pricing e ottimizzare il comfort attraverso il PMV. La scelta dell'algoritmo PSO per questa tipologia di problemi, è stata confermata dal confronto con l'algoritmo ABC.
L'efficacia dell'applicazione di algoritmi di ottimizzazione nella gestione energetica dell'edificio è stata anche dimostrata nel problema di ottimizzazione di due funzionali contrastanti: il comfort in termini di PMV e la spesa energetica. In questo caso, l'algoritmo selezionato, il Multi-Objective Particle Swarm Optimization, è risultato essere più performante al caso in cui non venga utilizzato un algoritmo di ottimizzazione.
Tali risultati delineano perciò un approccio metodologico di ottimizzazione della gestione energetica di un edificio. In effetti, attraverso l'utilizzo di algoritmi di ottimizzazione, risulta sempre possibile ottenere prestazioni migliori al caso in cui tali algoritmi non vengano utilizzati. Inoltre, la metodologia risulta generale, in quanto le funzioni obiettivo possono essere modificate e combinate in modo agevole, per poi essere implementate nei sistemi reali.

3 Conclusioni

In questa annualità l'attività svolta da ENEA ed Università Politecnica delle Marche mirava a fornire uno strumento per l'ottimizzazione di obiettivi specifici di natura energetica/economica/comfort di un edificio del terziario mediante l'utilizzo di pompa di calore, pannelli fotovoltaici e anche in scenari di dynamic pricing. Quindi l'attività di ricerca si è concentrata nella predisposizione del simulatore di carichi elettrici e termici con algoritmi di ottimizzazione e la loro implementazione in scenari demand-response. Gli algoritmi sono stati testati con il simulatore, sviluppato nelle annualità precedenti, dell'edificio terziario reale F40 del Centro Ricerche ENEA di Casaccia, attualmente riscaldato dalla rete di teleriscaldamento e raffrescato con chiller elettrici. L'edificio è stato simulato in una versione completamente "elettrica", vale a dire: si è simulata la climatizzazione (estiva ed invernale) con pompa di calore; l'installazione di un impianto fotovoltaico di potenza nominale pari a 20 kWp e di uno storage elettrico da circa 40 kWh. Lo scenario prevede anche un prezzo orario dell'energia variabile (dynamic pricing).
Il software della precedente annualità è stato modificato cosicché gli algoritmi di ottimizzazione possano accedere alle variabili decisionali e il risultato dell'ottimizzazione possa essere inviato al simulatore stesso. In particolare, lo scambio delle suddette variabili avviene mediante l'utilizzo di variabili condivise sul Workspace Matlab e file ".mat".
Successivamente sono stati definite delle funzioni di costo da ottimizzare con obiettivi specifici riguardanti ad esempio la minimizzazione della spesa energetica, la minimizzazione della spesa economica, la massimizzazione dell'autoconsumo da fonti rinnovabili, l'ottimizzazione del comfort, o una loro combinazione.
Infine, in ambiente Matlab/Simulink, l'algoritmo di ottimizzazione diversi algoritmi di ottimizzazione sono stati integrati nel software e utilizzati per la gestione ottimale dell'edificio F40.

Tutti gli scenari testati dimostrano come l'utilizzo di uno strumento di ottimizzazione dell'edificio possa contribuire a sfruttare al massimo l'autoconsumo di energia prodotta dall'impianto rinnovabile e/o a minimizzare la spesa economica e/o massimizzare il comfort dell'edificio.

Gli sviluppi futuri di questa attività consistono nell'ampliamento dei test del caso multiobiettivo in altri scenari e per periodi di tempo più lunghi, nell'intriduzione nel simulatore di una logica di gestione della batteria che tenga conto della possibilità di caricare la batteria con la rete per simulare scenari "price arbitrage" ed, infine, nell'implementazione dell'ottimizzazione proposta nel caso reale.

4 Riferimenti bibliografici

[1] Gabriele Comodi, Alessandro Fonti. Sviluppo di funzionalità per un simulatore di micro-distretto orientato alla gestione attiva della domanda. Report RdS/PAR2015

[2] G. Comodi, A. Fonti, A. Giantomassi, F. Polonara, S. Longhi. Sviluppo di un simulatore di edifici orientato alla gestione attiva della domanda. Report RdS/PAR2013/063.

[3] G. Comodi, A. Fonti, F. Polonara, S. Longhi. Miglioramento delle funzionalità di un simulatore di edificio e sua evoluzione verso la simulazione di reti di edifici in scenari di demand response. Report RdS/PAR2014/025

[4] Andrea Monteriù, Lucio Ciabattoni, Francesco Ferracuti, Gabriele Comodi, Sauro Longhi. Studi di scenari simulati di Demand-Response di un edificio terziario reale. Report RdS/PAR2016

[5] J. Kennedy and R. Eberhart, "Particle swarm optimization," in Proceedings., IEEE International Conference on Neural Networks, 1995., vol. 4, 1995, pp. 1942–1948 vol.4.

[6] Coello, C. A. C., Pulido, G. T., & Lechuga, M. S. (2004). Handling multiple objectives with particle swarm optimization. IEEE Transactions on evolutionary computation, 8(3), 256-279.

[7] Sierra, M. R., & Coello, C. A. C. (2005, March). Improving PSO-based multi-objective optimization using crowding, mutation and dominance. In International Conference on Evolutionary Multi-Criterion Optimization (pp. 505-519). Springer Berlin Heidelberg.

[8] Karaboga, D, (2005). An idea based on honey bee swarm for numerical optimization. Technical Report TR06, Erciyes University, Engineering Faculty, Computer Engineering Department.

[9] Tereshko, V., Loengarov, A. (2005), Collective decision-making in honey bee foraging dynamics, Computing and Information Systems, 9 (3): 1-7, University of the West of Scotland, UK.

[10] D. Karaboga and B. Akay, "A comparative study of artificial bee colony algorithm," Applied Mathematics and Computation, vol. 214, no. 1, pp. 108 – 132, 2009.

5 Curriculum Vitae

5.1 Andrea Monteriù

Andrea Monteriù è Ricercatore di Automatica presso il Dipartimento di Ingegneria dell'Informazione dell'Università Politecnica delle Marche dove attualmente è anche docente del Corso di Fondamenti di Automatica. Ha conseguito la Laurea Magistrale (V.O.) con lode in Ingegneria Elettronica nel 2003 presso l'Università Politecnica delle Marche, con una tesi sviluppata presso la Technical University of Denmark, Lyngby, Danimarca, sotto la supervisione del Prof. Mogens Blanke. Nel 2002 ha conseguito il Master of Science Degree in Electrical Engineering presso la stessa università danese. Ha conseguito il titolo di Dottore di Ricerca in Sistemi Artificiali Intelligenti nel 2006 presso l'Università Politecnica delle Marche, sotto la supervisione del Prof. Sauro Longhi. Nel 2005, ha lavorato come ricercatore visitatore presso il Center for Robot Assisted Search & Rescue della University of South Florida, Tampa, collaborando col Prof. Kimon Valavanis. È autore di oltre 140 pubblicazioni su riviste e congressi internazionali ed è autore di due brevetti. I suoi principali interessi di ricerca includono le metodologie di controllo per sistemi dinamici, la diagnosi guasti e le tecniche di controllo tollerante ai guasti, tecniche di controllo e guida di sistemi autonomi impiegati in diversi campi di applicazione, la robotica mobile e di servizio, e le tecnologie assistive.

5.2 Francesco Ferracuti

Francesco Ferracuti ha ricevuto nel 2010 ha conseguito la laurea specialistica in Automazione Industriale e nel 2014 il Dottorato di ricerca in Ingegneria Informatica, Gestionale e dell'Automazione dall'Università Politecnica delle Marche. Abilitato all'esercizio della professione di ingegnere nel Marzo 2013 nel settore Industriale. Da Settembre 2012 a Ottobre 2013 è stato visiting scholar presso The University of Manchester (UK). I suoi principali interessi di ricerca comprendono, la diagnosi guasti, l'elaborazione numerica dei segnali, l'apprendimento automatico, il riconoscimento di pattern e la loro applicazione in ambito industriale. Autore di oltre 50 articoli scientifici peer review, attualmente è il tesoriere della sezione italiana della IEEE Consumer Electronics Society e vicepresidente e co-fondatore dal 2017 dello spin-off Revolt srl dell'Università Politecnica delle Marche.

5.3 Lucio Ciabattoni

Lucio Ciabattoni, nato a San Benedetto del Tronto (AP) il 12 Luglio 1986, ha ricevuto nel 2008 la laurea triennale in Ingegneria Informatica e dell'Automazione (cum laude) dall' Università Politecnica delle Marche. Nel 2010 ha conseguito la laurea specialistica in Automazione Industriale (cum laude) e nel 2014 il Dottorato di ricerca in Ingegneria Informatica, Gestionale e dell'Automazione dall'Università Politecnica delle Marche. Dal Novembre 2012 a Giugno 2013 è stato visiting scholar presso la University of Arizona (USA), department of physics. Autore di oltre 60 articoli scientifici peer review è attualmente il presidente della sezione italiana della IEEE Consumer Electronics Society e associate editor delle IEEE Transactions on Consumer Electronics.
Ingegnere abilitato nel settore industriale, impegnato da oltre 5 anni nel settore della home automation, delle energie rinnovabili e dell'intelligenza artificiale, collabora regolarmente con diverse ditte del settore. Co-fondatore nel 2014 dello spin-off META srl dell'Università Politecnica delle Marche di cui è stato CEO fino al Febbraio 2016. Presidente e Co-Fondatore dal 2017 dello spin-off Revolt srl dell'Università Politecnica delle Marche.

5.4 Gabriele Comodi

Gabriele Comodi ha conseguito la laurea in ingegneria meccanica nel 2001 con la votazione di 110 e lode presso l'Università degli studi di Ancona (ora Politecnica delle Marche). Nel 2004 acquisisce il titolo di dottore di ricerca in "Energetica" presso il Dipartimento di Energetica (ora DIISM – Dipartimento di ingegneria industriale e scienze matematiche) dell'Università Politecnica delle Marche. In seguito, è stato titolare di 3 assegni di ricerca annuali e nel 2007 è diventato ricercatore presso il Dipartimento di Energetica dell'Università Politecnica delle Marche. Attualmente è ricercatore confermato presso il DIISM nel settore scientifico disciplinare "Sistemi per l'Energia e l'ambiente" (ING-IND/09).

I principali temi di ricerca sono: i) integrazione di sistemi di generazione distribuita (microturbine a gas, motori Stirling, motori a combustione interna, fuel cells, PV e CPV) in reti energetiche urbane; ii) efficienza energetica negli usi finali dell'energia; iii) demand side management di reti di edifici civili; iv) energy policy e programmazione energetica locale. E' autore di oltre 70 pubblicazioni a livello nazionale ed internazionale. E' titolare di un brevetto industriale. E' stato relatore/correlatore di oltre 100 tesi di laurea triennale e specialistica. E' attualmente supervisor di 4 candidati al titolo di dottore di ricerca.

E' membro del gruppo di lavoro "Urban Energy Network" del Joint Program EERA-"Smart Cities" ed è membro della Task Force "Simulation Tools" dello stesso Joint Program.

Dal 2015 è visiting research fellow presso l'Energy Research Institute della Nanyang Technological University di Singapore.

5.5 Sauro Longhi

Sauro Longhi ha conseguito la laurea con Lode in Ingegneria Elettronica presso l'Università degli Studi di Ancona nel 1979. Dal 1983 presta servizio presso la Facoltà di Ingegneria dell'Università Politecnica delle Marche e dal 2001 è professore ordinario di Automatica. Dal 2001 al 2013 è stato coordinatore del Dottorato di Ricerca in "Sistemi Artificiali Intelligenti", poi curriculum in Ingegneria Informatica, Gestionale e dell'Automazione della Scuola di Dottorato in Scienze dell'Ingegneria dell'Università Politecnica delle Marche. Dal 2011 al 2013 è stato Direttore del Dipartimento di Ingegneria dell'Informazione dello stesso Ateneo. Dal 2005 al 2012 è stato Presidente del Corso di Laurea in Ingegneria Informatica e dell'Automazione (CUCS) dell'Università Politecnica delle Marche. Dal 2012 al 2013 è stato Componente del Senato Accademico. Dal 2012 è componente del consiglio scientifico del Centro per l'Innovazione e l'Imprenditorialità. Dal Novembre 2013 è Rettore dell'Università Politecnica delle Marche. Scadenza del mandato di rettore: 31 ottobre 2019. Da Aprile 2014 a Novembre 2014 è stato Presidente nazionale del Cluster Tecnologie per gli Ambienti di Vita. Dal Maggio 2014 è Presidente del Consortium GARR (Gruppo per l'Armonizzazione delle Reti di Ricerca). Dal Dicembre 2014 è componente dell'Organo di Gestione e Controllo del Cluster Nazionale "Fabbrica Intelligente". E' componente dell' IFAC Technical Committee (TC) on Marine Systems. I suoi interessi di ricerca comprendono la modellazione, l'identificazione e il controllo di sistemi lineari e nonlineari, il controllo di robot mobili, veicoli sottomarini, navi e veicoli autonomi, controllo cooperativo di agenti autonomi, robot di servizio per applicazioni assistive a supporto della mobilità, automazione domestica e degli edifici, controllo decentralizzato su reti, reti di sensori, gestione dell'alimentazione in auto ibride, controllo motore elettrico, sistema di controllo integrato, gestione e controllo delle risorse energetiche rinnovabili, gestione efficiente dei sistemi energetici, rilevamento automatico dei guasti e isolamento. Ha pubblicato più di 400 articoli su riviste e congressi internazionali ed è autore di due brevetti.