POLITECNICO DI MILANO
Scuola di Ingegneria Industriale dell'Informazione
Dipartimento di Elettronica, Informazione e Bioingegneria

VALUTAZIONE SPERIMENTALE DI FORMULAZIONI DCOP PER PROBLEMI DI GESTIONE DELL'ENERGIA

Relatore: Prof. Francesco AMIGONI

Tesi di Laurea Magistrale di:
Roberto CARETTONI
Matr. 770763

Anno Accademico 2012-2013

Sommario

Negli ultimi anni il tema della sostenibilità dell'energia sta catturando grande interesse a livello mondiale, e le tecnologie che ne supportano la realizzazione stanno acquisendo una rilevanza scientifica notevole.
Questa tesi si colloca nell'ambito dell'utilizzo dei sistemi multiagente (MAS) per il problema della gestione della domanda di energia elettrica.
In particolare, lo scopo di questa tesi è di valutare in maniera sperimentale l'applicabilità della formulazione DCOP (Distributed Constraint Optimization Problem) a problemi di gestione dell'energia elettrica in ambito domestico, dove gli elettrodomestici sono rappresentati da agenti intelligenti che comunicano fra loro al fine di ottimizzare l'utilizzo della potenza disponibile, minimizzando i costi in bolletta per l'utente finale, e garantendo di conseguenza benefici per l'intera rete elettrica.
In questa tesi sono mostrate diverse formulazioni DCOP sviluppate per affrontare il problema applicativo menzionato, valutando per ciascuna di esse i vantaggi in termini di espressività del modello, i problemi dovuti alla sua complessità, e i risultati sperimentali ottenuti utilizzando diversi algoritmi di risoluzione noti.

Indice

Sommario ..... **III**

Indice ..... **V**

Elenco delle figure ..... **VII**

1 Introduzione **1**

2 Stato dell'arte **5**
2.1 Agenti autonomi e sistemi multiagente 5
2.1.1 Applicazioni MAS per la gestione dell'energia elettrica 5
2.2 Distributed Constraint Optimization Problem 8
2.2.1 Applicazioni DCOP per la gestione dell'energia elettrica 9
2.3 Algoritmi di risoluzione DCOP 10
2.3.1 ADOPT 10
2.3.2 SynchBB 12
2.3.3 DPOP 14
2.3.4 MGM 15
2.3.5 MGM-2 15

3 Impostazione del problema applicativo **17**
3.1 Formalizzazione del problema applicativo 18

4 Formulazione del problema applicativo attraverso DCOP **23**
4.1 FRODO 23
4.1.1 Architettura di FRODO 24
4.1.2 Formato di input di FRODO 25
4.2 Modelli sviluppati 30
4.2.1 Formulazione vettoriale 31
4.2.2 Formulazione ad attività 36
4.2.3 Formulazione ad attività semplificata 43
4.2.4 Formulazione Dynamic DCOP 46

5 Risultati sperimentali e valutazioni **51**
5.1 Generatore di problemi 51
5.2 Ottimizzazione con n=2 elettrodomestici 54
5.3 Ottimizzazione con n>2 elettrodomestici 57
5.4 Ottimizzazione al variare del passo di campionamento 59
5.5 Ottimizzazione con vincoli utente 61

6 Conclusioni e sviluppi futuri **65**

Bibliografia **67**

A Esempi di istanze DCOP **73**
A.1 Formulazione vettoriale 73
A.2 Formulazione ad attività 77
A.3 Formulazione ad attività semplificata 82

Elenco delle figure

4.1 Esempio di file XCSP per FRODO (XCSPparser) [14] 26
4.2 Esempio di file XCSP per FRODO (JaCoPxcspParser) [14] 27
4.3 Esempio di file di configurazione di FRODO corrispondente all'algoritmo DPOP [14] 29
4.4 Suddivisione ad attività di un profilo di consumo 37
5.1 Esempio di file scenario 51
5.2 Esempio di file consumi 52
5.3 Esempio di file tariffe 53
5.4 Confronto dei tempi medi di esecuzione con *n*=2 elettrodomestici 55
5.5 Confronto dei costi medi con *n*=2 elettrodomestici 56
5.6 Costi medi percentuali rispetto ai test conclusi entro il timeout 56
5.7 Confronto dei tempi medi di esecuzione con *n*=3 elettrodomestici 57
5.8 Confronto dei costi medi con *n*=3 elettrodomestici 58
5.9 Confronto dei tempi medi di esecuzione al variare del passo di campionamento 59
5.10 Confronto dei costi al variare del passo di campionamento 60
5.11 Spostamento dell'avvio di un elettrodomestico in seguito all'ottimizzazione con istante di tempo massimo fissato 62
5.12 Spostamento dell'avvio di un elettrodomestico in seguito all'ottimizzazione con delay fissato 63
A.1 Esempio di istanza DCOP formulata secondo il modello vettoriale 74
A.2 Esempio di istanza DCOP formulata secondo il modello ad attività 78
A.3 Esempio di istanza DCOP formulata secondo il modello ad attività semplificato 83

Capitolo 1

Introduzione

Negli ultimi anni il tema della sostenibilità dell'energia sta catturando grande interesse a livello mondiale. A oggi, più dell'85% del fabbisogno energetico mondiale è coperto dai combustibili fossili [37], ma la limitata disponibilità di tali fonti di energia, insieme alla consapevolezza dei danni ambientali causati dal loro abuso, sta rendendo imperativo un progressivo aumento dell'impiego di energie rinnovabili e il raggiungimento di una maggiore efficienza energetica.

Per rispondere a tale esigenza sono state ideate le *smart grid*: si tratta dell'insieme di una rete di informazione e di una rete di distribuzione di energia elettrica, in grado di gestire in maniera intelligente ed efficiente la distribuzione di energia ottimizzandone il trasporto e la diffusione agli utenti finali.

Un aspetto importante all'interno di questo contesto riguarda la necessità di rendere la domanda di energia adattabile all'offerta, in modo da appiattire i picchi di richiesta, permettendo così di ridurre il numero e l'utilizzo di costosi (e inquinanti) impianti di generazione di energia elettrica [26]. Per fare questo, le smart grid sfruttano il *Demand Side Management* (DSM), che consiste in metodi per modificare la domanda di energia da parte del consumatore, generalmente incoraggiandolo a utilizzare meno energia durante le ore di picco, o spostandone la richiesta all'infuori dei periodi di punta.

Queste problematiche possono essere affrontate in due modi differenti: con un approccio centralizzato o con un approccio distribuito. Sebbene gli approcci centralizzati siano più semplici da implementare, e siano stati estesamente

studiati, l'utilizzo di approcci distribuiti garantisce numerosi potenziali vantaggi, tra cui maggiore flessibilità, affidabilità e tolleranza ai guasti.

Tra gli approcci distribuiti, il paradigma dei sistemi multiagente (MAS) [31] sta guadagnando un interesse sempre crescente negli ultimi anni.

Lo scopo di questa tesi è di valutare in maniera sperimentale l'impiego della formulazione DCOP (Distributed Constraint Optimization Problem) per modellare, in ambito MAS, il problema della gestione dell'energia elettrica dal lato del consumatore. Il problema applicativo affrontato in questo lavoro è inquadrato nell'ambito domestico: gli agenti del sistema rappresentano gli elettrodomestici collegati alla rete, e l'obiettivo dell'ottimizzazione è la minimizzazione dei costi in bolletta dovuti al loro utilizzo, assicurando che in ogni istante di tempo considerato non si ecceda mai la soglia di potenza massima fornita dal gestore dell'energia, e garantendo all'utente il massimo comfort possibile.

Abbiamo quindi sviluppato diverse formulazioni DCOP per il problema applicativo studiato, valutando per ciascuna di esse i vantaggi in termini di espressività del modello, e i problemi dovuti alla sua complessità (è noto infatti che i problemi di ottimizzazione vincolata, anche distribuita, siano NP-difficili).

Uno dei vantaggi di adottare la formulazione DCOP è il disaccoppiamento tra i modelli impiegati e gli algoritmi di risoluzione: abbiamo perciò valutato i modelli presentati utilizzando diversi algoritmi, e servendoci di FRODO, un framework open-source ideato per la risoluzione di DCOP, che simula un sistema multiagente distribuito in una macchina virtuale Java.

Sebbene il lavoro svolto in questa tesi sia collocato all'interno dell'ambito domestico, il problema della gestione dell'energia risulta di grande importanza in numerose applicazioni reali, e può quindi essere trasportato in molti altri campi (industriale e militare, per citare due esempi).

La tesi è strutturata nel modo seguente.

Nel Capitolo 2 sono introdotti i sistemi multiagente e i problemi di ottimizzazione vincolata distribuita, e vengono presi in rassegna alcuni dei lavori precedenti in materia di gestione dell'energia elettrica; infine, vengono presentati gli algoritmi di risoluzione DCOP utilizzati per la verifica sperimentale.

Nel Capitolo 3 viene descritto e formalizzato il problema applicativo della gestione della domanda di energia all'interno di una rete elettrica a generazione di potenza limitata.

Nella prima parte del Capitolo 4 viene illustrata la struttura del framework FRODO; nella seconda parte sono invece descritti e formalizzati i diversi modelli DCOP sviluppati, ed è motivata la scelta di uno di questi come miglior compromesso fra espressività del modello e complessità di risoluzione dei problemi rappresentati.

Nel Capitolo 5 vengono mostrati i risultati sperimentali ottenuti con il modello scelto utilizzando gli algoritmi presentati.

Nel Capitolo 6 sono riassunti gli obiettivi di questa tesi, le valutazioni del lavoro svolto e le prospettive future.

In Appendice A sono mostrati degli esempi di istanze DCOP formulate secondo i modelli presentati.

Capitolo 2

Stato dell'arte

2.1 Agenti autonomi e sistemi multiagente

Nell'ambito dell'Intelligenza Artificiale, con il termine *agente* si intende una qualsiasi entità (un robot o un programma software, ad esempio) in grado di interagire con l'ambiente circostante tramite percezioni (gli input) e azioni (gli output).

Un *sistema multiagente* (MAS) [31] è un sistema composto da diversi agenti intelligenti che operano all'interno di uno stesso ambiente interagendo anche fra di loro. Scopo principale della ricerca in quest'ambito è di trovare metodi che permettano di costruire sistemi complessi composti da agenti autonomi i quali, operando soltanto sulla base di conoscenze locali e possedendo abilità limitate, sono comunque in grado di mettere in atto il comportamento globale desiderato.

Negli ultimi anni il paradigma MAS è stato applicato con successo in diversi ambiti, tra cui ad esempio la sicurezza [23], la gestione del traffico [2], e le simulazioni di risposta ai disastri [30].

Di particolare interesse per questo lavoro di tesi sono stati gli sviluppi dei sistemi multiagente applicati alla gestione dell'energia elettrica.

2.1.1 Applicazioni MAS per la gestione dell'energia elettrica

Fornire energia in maniera sostenibile è un problema attuale di interesse mondiale, e va gestito sotto diversi aspetti. Un modo per affrontare questa

sfida è sfruttare le energie rinnovabili. Un altro approccio possibile, se non doveroso, è risparmiare energia durante le fasi di distribuzione e consumo.

Per raggiungere tale scopo, i sistemi di distribuzione dell'energia elettrica (come le smart grid) devono essere in grado di predire il consumo dell'utenza e di reagire in maniera intelligente e tempestiva. La natura distribuita e il comportamento autonomo di tali sistemi li rendono particolarmente adatti alla metodologia multiagente.

In [33] è stato sviluppato un framework basato sulla teoria dei giochi per modellare dispositivi di stoccaggio di energia elettrica in sistemi a larga scala, dove ogni dispositivo è controllato da un agente egoista che punta a massimizzare il proprio profitto monetario. Sotto determinate assunzioni, la strategia proposta permette a tutti i dispositivi connessi di convergere a un comportamento profittevole ed efficiente.

In [24] è stato sviluppato un modello che considera, oltre che una gestione più complessa della rete, anche la gestione del comfort all'interno delle case attraverso l'uso di *smart meter*, dei dispositivi in grado di comunicare con la smart grid. Tali dispositivi, attraverso l'utilizzo di tecniche di apprendimento automatico, predicono l'andamento del mercato energetico del giorno successivo, permettendo così di gestire l'ottimizzazione dei carichi elettrici utilizzando una politica adattiva, piuttosto che reattiva.

In [32] è stato presentato un meccanismo di aste fra agenti, rappresentanti i compratori e i venditori di energia, che garantisce che gli acquirenti paghino un prezzo equo per eventuali picchi di domanda, e che i fornitori ricevano un compenso adeguato per rispondere a tali aumenti improvvisi. In tale modo, il mercato garantisce le condizioni migliori per tutti gli agenti coinvolti nelle trattative, anche nel caso in cui le previsioni di domanda e offerta siano imprecise; in [10] è stato presentato un sistema di coordinamento per bilanciare la domanda e l'offerta in reti elettriche.

Oltre ad ottimizzare la fase di distribuzione attraverso tecniche come la gestione delle smart grid, per raggiungere lo scopo di ottenere energia sostenibile è importante anche minimizzarne il consumo. L'approccio multiagente risulta essere particolarmente adatto anche in questo caso: appare naturale, infatti, modellare carichi elettrici collegati ad una stessa rete come agenti autonomi che comunicano fra loro per ottenere la migliore allocazione possibile della risorsa condivisa (l'energia elettrica).

In [13] è stato proposto un sistema multiagente basato sul coordinamento distribuito in situazioni di incertezza attraverso i processi di decisione di Markov (MDP), in grado di ottimizzare sia la quantità di energia consumata, sia il livello di comfort degli utenti.

In [25] è stato mostrato un sistema di riscaldamento casalingo che minimizzasse costi ed emissioni di anidride carbonica all'interno di una smart grid, mantenendo il comfort degli utenti il più alto possibile. Il sistema è in grado di determinare le caratteristiche termiche della casa e, attraverso l'impiego di processi gaussiani, può predire i parametri ambientali delle successive ventiquattro ore, in modo da poter impostare adeguatamente il sistema di riscaldamento.

In [1] è stato proposto un modello in cui gli agenti sono suddivisi tra agenti che forniscono l'energia e agenti che la utilizzano. Essi cooperano e si coordinano per raggiungere una soluzione il più possibile vicino all'ottimo, cercando di soddisfare i vincoli di comfort dell'utente e rispettando nel frattempo i vincoli energetici. Il sistema utilizza un protocollo di negoziazione che sfrutta due meccanismi principali: di emergenza e di anticipazione. Il primo garantisce il rispetto dei vincoli, mentre il secondo calcola il miglior uso possibile dell'energia, in base a predizioni sulla produzione, sul consumo e criteri di comfort per gli utilizzatori.

2.2 Distributed Constraint Optimization Problem

Un gran numero di problemi di coordinamento in sistemi multiagente, tra cui scheduling di attività, pianificazione di itinerari, allocazione di risorse ed altri, può essere modellato come un Distributed Constraint Optimization Problem (DCOP).

Un DCOP [31] è una formulazione matematica di un problema adatta a modellare un sistema multiagente, nel quale gli agenti cooperano al fine di ottimizzare una funzione obiettivo globale. Ogni agente gestisce una variabile, che ha un proprio dominio finito e discreto, e il cui valore può essere vincolato al valore assegnato dagli altri agenti alle proprie variabili. Solo l'agente che controlla una variabile ne conosce il dominio e può deciderne il valore.

In un DCOP, a differenza di un DCSP [31] (Distributed Constraint Satisfaction Problem), i vincoli tra le variabili sono associati a valori numerici che rappresentano dei costi (o delle utilità, a seconda che si tratti di un problema di minimizzazione o massimizzazione), e la funzione obiettivo globale è espressa come aggregazione delle funzioni di costo (o utilità) tra le variabili.

Un DCOP si può quindi formalizzare con una tupla $< X, D, F >$, dove $X = \{x_1, x_2, \dots, x_n\}$ è un insieme di n variabili, ognuna associata a un agente, $D = \{D_1, D_2, \dots, D_n\}$ è l'insieme dei domini delle variabili e $F = \{f_1, f_2, \dots, f_k\}$ è un insieme di funzioni di costo (vincoli) della forma $f_k(x_{k1}, x_{k2}, \dots, x_{kj}) \rightarrow \mathbb{R} \cup \pm\infty$. L'obiettivo è trovare l'assegnamento X^* che minimizzi la funzione di costo aggregata (o massimizzi la funzione di utilità aggregata) espressa come $R(X) = \sum_k f_k$, quindi $X^* = argmin_X R(X)$.

I DCOP sono problemi NP-difficili: ciò significa che può essere molto complesso per un algoritmo completo risolvere un problema al crescere del

numero delle variabili presenti, a causa della crescita esponenziale del carico computazionale.

2.2.1 Applicazioni DCOP per la gestione dell'energia elettrica

In [27] l'approccio DCOP è stato utilizzato per modellare il controllo della distribuzione di potenza tra carichi appartenenti a uno stesso sistema elettrico, soddisfacendo sia i vincoli di funzionamento dei singoli dispositivi sia l'obiettivo di efficienza globale del sistema, identificato come il maggior numero di dispositivi accesi in relazione alla potenza totale disponibile nel sistema.
In [12] è stata proposta un'estensione di DCOP (Resource Constrained DCOP, o Multiply-Constrained DCOP) per trattare il caso delle reti di distribuzione energetica (in particolare del ripristino della rete in seguito a problemi sulla linea, dovuti ad esempio alla caduta di un nodo di generazione o di un arco di trasmissione), dove il flusso di energia dalle sorgenti ai pozzi è limitato dalla capacità di flusso. Per affrontare il problema sono state utilizzate delle nuove tipologie di vincoli sulle risorse che sfruttano l'aciclicità e la conservazione del flusso delle reti di distribuzione.
Gli approcci più comuni per il problema della gestione dell'energia elettrica in un ambiente con disponibilità di potenza limitata, tuttavia, sono basati principalmente su semplici soluzioni di distacco di carico (Load Shedding), le quali consistono nel disconnettere uno o più dispositivi dalla rete elettrica, quando non è disponibile abbastanza potenza da soddisfare la richiesta. Tali tecniche hanno il vantaggio di essere semplici da implementare, e permettono di trovare un assegnamento accettabile in un tempo relativamente breve; tuttavia, questo approccio non è adatto in determinati domini applicativi come quello domestico, in quanto difetta in flessibilità, permettendo soltanto un comportamento dei carichi di tipo acceso/spento (si pensi ad esempio ad una lavastoviglie disconnessa dalla rete mentre riscalda l'acqua per il lavaggio: ciò

renderebbe necessario, una volta disponibile la potenza richiesta, farle riprendere il ciclo dall'inizio, sprecando così energia e aumentando i costi).
L'approccio DCOP, oltre a permettere di ovviare a questa limitazione, fornisce un metodo generale per modellare svariati tipi di problemi in diverso modo, superando la necessità di creare soluzioni ad hoc per problemi specifici. Esistono inoltre numerose implementazioni di algoritmi di risoluzione di DCOP, perciò secondo le esigenze è possibile scegliere tra un'ampia gamma di soluzioni, ognuna con i propri punti di forza e di debolezza.

2.3 Algoritmi di risoluzione DCOP

Esistono due classi principali di algoritmi di risoluzione per problemi DCOP: gli algoritmi di inferenza e gli algoritmi di ricerca. Gli algoritmi di inferenza riducono sistematicamente il problema in un problema equivalente più semplice, mentre gli algoritmi di ricerca esplorano tutto lo spazio dei possibili assegnamenti delle variabili.
Esistono poi sia algoritmi completi (che trovano la soluzione ottima) sia incompleti (che trovano una soluzione non necessariamente ottima).
ADOPT, SynchBB, e DPOP sono esempi di algoritmi completi, mentre MGM e MGM-2 sono esempi di algoritmi incompleti.

2.3.1 ADOPT

ADOPT (Asynchronous Distributed OPTimization) [16] è un algoritmo corretto (ossia termina sempre producendo un risultato corretto) e completo, in grado di trovare una soluzione ottimale al problema o almeno una soluzione a una distanza massima dall'ottimo definita dall'utente, utilizzando comunicazioni locali asincrone e con complessità spaziale polinomiale per ogni agente.

Invece di agire sul grafo dei conflitti (un grafo costruito facendo corrispondere uno specifico nodo a ogni variabile, e tracciando un arco tra due nodi se esiste almeno un conflitto fra gli assegnamenti delle variabili corrispondenti), ADOPT costruisce un albero di ricerca in profondità (Depth First Search) che definisce un ordinamento tra gli agenti attraverso i rapporti di parentela tra i nodi dell'albero (che rappresentano appunto gli agenti).

La comunicazione è locale giacché un agente non scambia messaggi con tutti gli altri, ma solo con il nodo padre e i nodi figlio nell'albero DFS.

L'idea principale dietro ADOPT è di ottenere asincronismo permettendo a ogni agente di cambiare il valore assegnato alla propria variabile quando scorge la possibilità di trovare una soluzione migliore rispetto a quella attuale. Questa strategia di ricerca permette asincronismo perché un agente non necessita una conoscenza globale del problema per prendere una decisione riguardo al suo problema locale. Dato che questa strategia permette di abbandonare soluzioni prima che sia provata la loro ottimalità locale, è possibile che alcune di esse siano riutilizzate.

La seconda idea chiave in ADOPT è quindi quella di ricostruire in maniera efficiente soluzioni considerate precedentemente (con complessità spaziale polinomiale) attraverso l'uso di una soglia di backtrack, cioè una tolleranza sul costo della soluzione che previene l'utilizzo di backtracking indiscriminato.

La terza caratteristica principale di ADOPT consiste nel fornire un meccanismo di individuazione della terminazione all'interno dell'algoritmo stesso: gli agenti terminano infatti quando trovano una soluzione completa il cui costo è al di sotto della loro soglia di backtrack.

I messaggi scambiati fra gli agenti sono:

- messaggi VALUE: inviati da un nodo verso i propri figli, percorrendo quindi l'albero DFS dall'alto verso il basso. Contengono il valore assegnato dall'agente alla propria variabile;
- messaggi COST: inviati da un nodo verso il proprio padre, percorrendo quindi l'albero DFS dal basso verso l'alto. Contengono il valore che

rappresenta il costo della scelta effettuata dal nodo genitore sommato ai costi delle scelte effettuate da tutti i nodi figli. Servono inoltre per calcolare i limiti inferiori e superiori utilizzati dagli agenti per guidare il backtracking;

- messaggi THRESHOLD: inviati da un nodo verso i propri figli, percorrendo quindi l'albero DFS dall'alto verso il basso. Servono per ridurre ridondanze nella ricerca notificando la soglia di backtrack.

Ogni agente mantiene traccia degli assegnamenti dei vicini con priorità maggiore attraverso un *contesto*, ovvero una soluzione parziale contenente gli assegnamenti noti. Tale contesto viene aggiornato dall'agente ogni qualvolta viene ricevuto un messaggio VALUE dal nodo padre oppure quando l'agente decide di cambiare il valore della propria variabile in seguito alla ricezione di un messaggio COST da un nodo figlio.

L'algoritmo giunge al termine quando i limiti inferiore e superiore dell'agente al nodo radice sono uguali, il che significa che è stata trovata la soluzione ottima.

Lo svantaggio principale di ADOPT è la sua complessità in termini di messaggi scambiati, che cresce esponenzialmente rispetto al numero degli agenti (mentre lo spazio utilizzato dai messaggi e la memoria richiesta per ogni agente cresce linearmente). Per questa ragione l'algoritmo degrada significativamente in prestazioni al crescere del numero degli agenti.

2.3.2 SynchBB

SynchBB (Synchronous Branch and Bound) [8] è un algoritmo completo pensato inizialmente per risolvere problemi di soddisfacimento di vincoli distribuiti (DCSP) caratterizzati da un elevato numero di vincoli, adottando un approccio per problemi di soddisfacimento parziale di vincoli distribuiti (DPCSP).

Un DPCSP è formulato partendo da un DCSP per il quale non esistono soluzioni ammissibili. Introducendo una relazione di priorità fra i vincoli, nel caso non sia possibile trovare una soluzione ammissibile per il problema di partenza, si possono rilassare i vincoli di priorità minore, fino a trovare una soluzione che soddisfi i vincoli di priorità maggiore.

SynchBB simula il noto approccio Branch and Bound in ambiente distribuito. Gli agenti, che comunicano attraverso uno schema che riflette il grafo dei conflitti, costruiscono in maniera incrementale un *percorso*, cioè un assegnamento di valori per le variabili del problema. A ogni iterazione il percorso (che contiene anche il numero di vincoli violati) viene valutato ed eventualmente esteso o modificato attraverso ulteriori assegnamenti. Il percorso di lunghezza n (con n uguale al numero degli agenti) corrisponde alla soluzione del problema.

Dato che SynchBB simula l'approccio Branch and Bound in ambiente distribuito, appare evidente che si tratti di un algoritmo corretto. Inoltre il controllo sequenziale degli agenti e l'ordinamento delle variabili permettono di esplorare l'intero spazio delle soluzioni, garantendo così che l'algoritmo sia completo.

Il maggior limite di SynchBB risiede nello schema di comunicazione: è necessario, infatti, che i messaggi siano ricevuti nello stesso ordine nel quale sono stati inviati, così come anche le fasi computazionali devono essere eseguite sequenzialmente, perciò l'algoritmo non può trarre vantaggio da alcun tipo di parallelismo.

In [6] questo approccio è stato esteso per puntare non al solo soddisfacimento dei vincoli, ma anche alla soluzione ottima, rendendo SynchBB adatto a risolvere i DCOP.

2.3.3 DPOP

DPOP (Dynamic Programming Optimization Protocol) [21] è un algoritmo corretto e completo basato sulla programmazione dinamica. Si tratta di un metodo a propagazione di utilità, ispirato dagli algoritmi Sum-Product [11] e DTREE [20], i quali però sono corretti solo per reti di vincoli a forma d'albero, mentre DPOP ne estende l'utilizzo a topologie arbitrarie adoperando un arrangiamento a pseudo-albero possibile a partire da un qualsiasi grafo dei vincoli del problema.

L'algoritmo si divide in tre fasi: la costruzione dello pseudo-albero, la propagazione delle utilità e la propagazione dei valori.

Durante la fase di propagazione delle utilità, gli agenti inviano, partendo dalle foglie dello pseudo-albero, i messaggi UTIL ai propri genitori. Tali messaggi rappresentano un vettore delle utilità che possono essere ottenute dai sotto-alberi con radice nei corrispondenti agenti, compatibilmente con ogni valore assegnabile dal nodo genitore. Le utilità vengono propagate risalendo l'albero, fino a quando il nodo radice riceve tutti i messaggi UTIL in arrivo dai propri figli.

A questo punto, in accordo con le informazioni ricevute, il nodo radice assegna alla propria variabile il valore che massimizza l'utilità del proprio sotto-albero. La sua scelta è poi notificata attraverso l'invio di un messaggio VALUE ai propri figli, i quali a loro volta, ricevuto il messaggio VALUE dai propri genitori, scelgono il valore ottimo per la propria variabile e propagano ai figli la propria scelta, fino a raggiungere le foglie dell'albero.

Il punto di forza di DPOP riguarda la complessità in termini di messaggi scambiati, che cresce linearmente rispetto all'altezza dello pseudo-albero, generando quindi un basso overhead di comunicazione. Tuttavia le performance possono essere influenzate dalla dimensione dei messaggi UTIL, che cresce esponenzialmente rispetto alla larghezza dello pseudo-albero.

2.3.4 MGM

MGM (Maximum Gain Messages) [15] è un algoritmo incompleto, in quanto non garantisce di trovare la soluzione ottima, se esiste. Si tratta di una modifica di un altro algoritmo, Distributed Breakout Algorithm (DBA) [36], il quale però non può essere applicato a sistemi realmente distribuiti, poiché richiede una conoscenza globale della qualità della soluzione. MGM prende spunto dalla teoria dei giochi, in quanto i DCOP vengono modellati come giochi in cui i giocatori sono le variabili controllate dagli agenti, le azioni sono gli assegnamenti di valori, e le informazioni note sono i valori noti degli agenti vicini. In questo modo, una soluzione corrisponde a un equilibrio di Nash del problema rappresentato.

L'algoritmo agisce in diversi turni (*round*): a ogni round gli agenti calcolano la propria utilità locale, cioè il massimo valore di utilità ottenibile cambiando l'assegnamento attuale del proprio *contesto* (l'insieme delle variabili appartenenti agli agenti vicini di cui un agente conosce l'assegnamento). Poi, gli agenti mandano un messaggio a tutti i vicini, indicando se l'assegnamento sortirà un guadagno o no. Un assegnamento è eseguito solo dall'agente che ne trarrà il maggior guadagno. L'algoritmo termina se non esiste nessun ulteriore assegnamento che può generare un guadagno.

Un aspetto molto importante di MGM riguarda la monotonicità dell'utilità globale rispetto al numero di round: in qualunque momento l'algoritmo sia interrotto, la soluzione trovata sino a quel momento sicuramente non è peggiore di una soluzione trovata in un round precedente.

2.3.5 MGM-2

Come MGM, anche MGM-2 [15] è un algoritmo incompleto basato sulla teoria dei giochi. Mentre in MGM gli agenti puntano a massimizzare soltanto

la propria utilità, unilateralmente, in MGM-2 ogni agente è coordinato con un altro.

A ogni round, gli agenti vengono divisi casualmente tra *offerenti* e *riceventi*; ogni offerente poi, sceglie casualmente un vicino a cui mandare un'*offerta*, la quale contiene i valori consigliati per l'offerente e il ricevente che porterebbero ad un aumento dell'utilità dell'offerente. Se il messaggio è inviato a un agente ricevente, questo calcola il guadagno in utilità globale per ogni coppia di valori nell'offerta (se invece l'offerta è ricevuta da un altro offerente, sarà semplicemente scartata). Se il guadagno globale massimo è positivo, il ricevente accetta l'offerta inviando un messaggio *accept*, ed entrambi assegnano alle proprie variabili i nuovi valori; in caso contrario il ricevente rifiuta con un messaggio *reject*.

Anche MGM-2 rispetta l'importante caratteristica di monotonicità dell'utilità globale. Rispetto a MGM, il costo in messaggi scambiati con quest'algoritmo è molto maggiore, ma in media MGM-2 trova soluzioni qualitativamente migliori.

Capitolo 3

Impostazione del problema applicativo

Scopo del lavoro è affrontare il problema della gestione della domanda di energia all'interno di una rete elettrica, in cui la limitata potenza disponibile deve essere distribuita tra i carichi connessi alla rete secondo le necessità, soddisfacendo tutti i vincoli imposti dall'utente, dai carichi e dal sistema stesso, minimizzando nel frattempo una funzione obiettivo associata.

Nello specifico, questo problema è stato affrontato nell'ambito domestico: i carichi elettrici sono rappresentati dagli elettrodomestici, che richiedono una certa quantità di potenza per funzionare, determinata dal programma di esecuzione (ad esempio, *Eco 60°* rappresenta uno dei possibili programmi di una lavastoviglie, e ha una curva di assorbimento differente dal programma *Standard 60°*). La potenza disponibile però non è illimitata, bensì è determinata dal gestore in base al contratto stipulato con l'utente. In ogni momento perciò la potenza assorbita da tutti gli elettrodomestici attivi non deve superare la soglia di potenza massima concordata.

I carichi elettrici, inoltre, possono avere diverse priorità per l'utente: quelli con priorità maggiore hanno la precedenza su quelli con priorità minore nel caso la potenza disponibile non sia sufficiente per mantenerli tutti attivi, o nel caso sia necessario spostare l'istante di avvio di un elettrodomestico per diminuire i costi.

Gli utenti possono anche esprimere altri tipi di preferenze: ad esempio, possono imporre che un carico termini il proprio ciclo di funzionamento entro un certo orario, oppure che l'avvio di un elettrodomestico sia anticipato o posticipato entro una certa soglia temporale massima.

L'obiettivo che ci si propone è di minimizzare i costi in bolletta sostenuti quando associamo al profilo di consumo di un'abitazione un piano tariffario energetico. Le tariffe sono generalmente differenziate per fasce orarie, e ad ogni fascia è associato un costo espresso in $€/KWh$. Benché la tariffa maggiormente diffusa tra i clienti domestici sia la tariffa bioraria, in [17] è stato mostrato come con questo tipo di tariffa i risparmi prodotti da un algoritmo di ottimizzazione siano troppo esigui da giustificare l'utilizzo del prototipo. Per questo motivo abbiamo scelto di adottare una tariffa oraria differenziata estratta dal *Mercato del giorno prima* (MGP) [5], una sede di negoziazione riconducibile alla borsa elettrica che permette di vendere o acquistare energia elettrica per ciascuna ora del giorno successivo.
Abbiamo formalizzato il problema applicativo come un DCOP.

3.1 Formalizzazione del problema applicativo

Possiamo esprimere il problema descritto nel modo seguente:

- n: numero dei carichi connessi alla rete;
- $(a_1, a_2, \dots, a_n)$: insieme degli agenti del sistema. A ogni carico è associato esattamente un agente;
- P_a: valore costante che rappresenta la potenza massima disponibile all'interno della rete;
- $(x_1, x_2, \dots, x_n)$: insieme delle variabili controllate dagli agenti.
- T: insieme che rappresenta i time step in cui è suddiviso l'orizzonte temporale di ottimizzazione. Ad esempio, con un orizzonte temporale giornaliero (quindi di 24 ore), utilizzando un passo di campionamento di 1 minuto, si ha $T = \{1,2, \dots ,1440\}$;
- $\left(p_{11}(t), \dots, p_{1m_1}(t), p_{21}(t), \dots, p_{2m_2}(t), p_{n1}(t), \dots, p_{nm_m}(t)\right)$: insieme dei profili di consumo degli *n* carichi. Ogni carico può avere più profili

di consumo. A ogni quanto di tempo è associata la potenza istantanea assorbita nell'intervallo temporale dall'elettrodomestico; ogni profilo di consumo assume la forma: $p_{ik}(t) \in \mathbb{R}, i \in [1,n], k \in [1,m_i], t \in T$;

- $(D_{11}, \dots, D_{1m_1}, D_{21}, \dots, D_{2m_2}, \dots, D_{n1}, \dots, D_{nm_n})$: insieme dei domini dei consumi istantanei degli *n* carichi. Ogni dominio assume la forma: $D_{ik} = p_{ikt}, i \in [1,n], k \in [1,m_i], t \in T$;
- $X_{ik}, i \in [1,n], k \in [1,m_i]$: potenza massima che può assorbire l'i-esimo carico nel profilo di consumo k-esimo;
- $priority_i, i \in [1,n], priority \in [1, priority_{max}]$: valore di priorità che è possibile associare a ciascun agente. Maggiore è il suo valore, maggiore è la priorità associata all'agente corrispondente. Questa caratteristica non è stata implementata in alcuna delle formulazioni successivamente sviluppate, in quanto non è stata ritenuta essenziale ai fini del lavoro. È comunque interessante la possibilità di modellare priorità fra i carichi;
- $(c_1, c_2, \dots, c_l)$: insieme delle funzioni di costo associate ai carichi. Ogni funzione di costo assume la forma: $c_j = f_j(x_{j1}, x_{j2}, \dots, x_{ji}) \to \mathbb{R} \cup \pm\infty, j \in [1,l], i \in [1,n]$;

Assumiamo che gli agenti siano sempre connessi al sistema, anche quando i carichi associati non stiano consumando temporaneamente energia.

Come esempio, consideriamo la semplice situazione seguente:

- $n = 2$;
- $P_a = 10$;
- $T = \{1, \dots, 5\}$
- $p_{11}(t) = \{4,4,4,4,4\}, p_{12}(t) = \{9,9\}$;
- $p_{21}(t) = \{6,6,6,6\}$;

Utilizzando i dati appena specificati, abbiamo formulato uno scenario in cui sono presenti due elettrodomestici, controllati quindi da due agenti (a_1, a_2), la

potenza massima disponibile in ogni istante di tempo è di 10 W, e l'orizzonte temporale considerato è suddiviso in 5 campioni. L'elettrodomestico associato all'agente a_1 può assumere due profili di consumo: uno richiede un assorbimento di potenza di 4 W per 5 istanti di tempo, l'altro richiede un consumo di 9 W per 2 istanti di tempo; l'elettrodomestico associato all'agente a_2, invece, possiede un unico profilo di consumo che richiede un assorbimento di potenza di 6 W per 4 istanti di tempo.

Come mostrato nel capitolo successivo, partendo da una formalizzazione generale come quella appena descritta, è possibile cercare di raggiungere una soluzione al problema di ottimizzazione seguendo strade diverse: ad esempio, possiamo utilizzare le variabili di decisione (x_1, x_2) per rappresentare la potenza assorbita da un agente in un dato istante di tempo. In tal caso, i domini delle variabili risultano quindi essere: $D_{11} = \{4\}$, $D_{12} = \{0, 9\}$, $D_{21} = \{0, 6\}$; con il primo profilo di consumo l'agente a_1 può assegnare alla propria variabile un unico valore, mentre con il secondo profilo, siccome esso prevede un tempo di funzionamento di soli due campioni temporali, l'agente può assegnare anche il valore "0" (così come l'agente a_2), che corrisponde allo spegnimento dell'elettrodomestico. Volendo ad esempio trovare un assegnamento delle variabili che sfrutti il più possibile l'energia disponibile, la funzione obiettivo da minimizzare (considerando il problema suddiviso in istanze indipendenti corrispondenti ad ogni istante di tempo) si potrebbe così esprimere: $f(x_1, x_2) = max_{util} + availability$, dove la componente $max_{util} = P_a - (x_1 + x_2)$ incentiva l'utilizzo della potenza disponibile, e la componente $availability = \begin{cases} 0 \Leftrightarrow max_{util} \geq 0 \\ \infty \; altrimenti \end{cases}$ impedisce che la potenza allocata superi quella disponibile all'interno della rete.

Questa formalizzazione assume una forma più generale rispetto alla trattazione effettuata sullo stesso dominio applicativo in due lavori precedenti, da cui abbiamo tratto ispirazione per questo studio.

In [3], l'obiettivo di distribuire la potenza disponibile fra i carichi connessi alla rete è stato affrontato con una formulazione tale da rendere necessario generare per ogni istante di tempo considerato una nuova istanza DCOP. Ad esempio, per risolvere un problema di ottimizzazione giornaliera con questo modello, utilizzando un passo di campionamento di 1 minuto, bisogna generare 1440 istanze DCOP, ognuna indipendente dall'altra. Il limite di questa formulazione è quindi la mancanza di una rappresentazione interna del tempo. Ogni istanza DCOP è atomica e a sé stante, e non tiene conto dei risultati delle istanze precedenti. Con la formulazione che abbiamo presentato questo limite è stato superato, giacché vi è una rappresentazione esplicita del tempo nel modello.

In [17], invece, è stato affrontato il problema della minimizzazione dei costi in bolletta tramite la decisione degli istanti d'inizio dei cicli di funzionamento degli elettrodomestici, grazie all'utilizzo dei loro profili di consumo. In questa formulazione, quindi, il tempo è stato espresso esplicitamente. Tuttavia, questa rappresentazione non ha permesso alcuna libertà di azione sulla definizione dei profili di consumo. Con la formulazione che abbiamo presentato, invece, è data la possibilità di agire su di essi attivamente: ad esempio, potrebbe essere ammissibile allungare il ciclo di riscaldamento dell'acqua di una lavastoviglie, riducendo nel frattempo la potenza richiesta durante tale attività (nell'esempio precedente, scegliendo il profilo di consumo $p_{11}(t)$ anziché il profilo $p_{12}(t)$). Inoltre, essendo il nostro un approccio distribuito, è l'agente stesso a decidere la soluzione migliore per sé, valutando soltanto i propri possibili profili di consumo. In [17], invece, è stato utilizzato un approccio centralizzato: l'intera mole di dati e di calcoli è perciò assegnata a un unico agente onnisciente.

Capitolo 4

Formulazione del problema applicativo attraverso DCOP

Nel Capitolo 3 abbiamo introdotto e formalizzato il problema applicativo affrontato, anticipando l'intenzione di utilizzare la formulazione DCOP per la sua risoluzione.
Per fare ciò, abbiamo utilizzato una modellazione che fosse coerente con lo standard di FRODO, il framework poi utilizzato per risolvere i problemi.

4.1 FRODO

FRODO (FRamework for Open/Distributed constraint Optimization) [19] è un framework Java open-source per la soluzione di problemi di ottimizzazione vincolata distribuita (DCOP), sviluppato inizialmente all'Artificial Intelligence Laboratory (LIA) dell'Ecole Polytechnique Fédérale de Lausanne (EPFL), in Svizzera.
Per la realizzazione di questo lavoro di tesi abbiamo utilizzato FRODO 2.11, una versione completamente riprogettata e reimplementata rispetto a quella originale [14].
FRODO si occupa di simulare in una singola macchina virtuale Java un ambiente multiagente, in cui ogni agente è eseguito in maniera asincrona in un thread dedicato, e comunica con gli altri attraverso scambi di messaggi. Con la nuova revisione di FRODO è anche possibile creare un vero e proprio ambiente distribuito, in cui ogni agente è ospitato su un diverso elaboratore e comunica con gli altri attraverso la rete tramite il protocollo TCP.

Il framework è molto personalizzabile, e permette all'utente di implementare e sperimentare un qualsiasi algoritmo di ottimizzazione distribuita.

4.1.1 Architettura di FRODO

FRODO è strutturato con un'architettura multistrato modulare.
Lo strato di comunicazione, responsabile dello scambio di informazioni fra gli agenti, è implementato come una coda di messaggi Java. Tipicamente ogni agente possiede una coda, che utilizza per ricevere e inviare messaggi. Ogni coda possiede il proprio processo di esecuzione, perciò particolare attenzione è stata posta nell'evitare che i processi potessero bloccarsi per lungo tempo in attesa di messaggi, in modo da limitare le implicazioni di performance dovute all'avere un solo thread per agente, nel caso di problemi dove un grande numero di agenti risiede sulla stessa JVM.
Lo strato degli spazi delle soluzioni fornisce le classi per modellare i problemi di ottimizzazione combinatoria. Dato uno spazio di possibili assegnamenti ad alcune variabili, uno *spazio delle soluzioni* è una rappresentazione di assegnamenti tali che questi corrispondano a delle soluzioni del dato problema.
Nel contesto di problemi di ottimizzazione, gli spazi delle soluzioni sono associati alle utilità (o costi), in modo da poter valutare la qualità di ogni soluzione.
Tra le diverse implementazioni degli spazi delle soluzioni in FRODO, la più semplice è quella a *ipercubo*, ossia una rappresentazione estensionale di uno spazio in cui a ogni combinazione di assegnamenti delle variabili è associata un'utilità (o costo). Gli assegnamenti non ammissibili possono essere rappresentati tramite uno speciale costo infinito. Gli spazi delle soluzioni possono essere espressi anche in maniera intensionale, basandosi sui vincoli JaCoP [9].

Lo strato degli algoritmi, infine, si occupa dell'implementazione vera e propria degli algoritmi distribuiti utilizzati per risolvere i DCOP. Ogni algoritmo è implementato come uno o più *moduli*, i quali sono a tutti gli effetti delle politiche algoritmiche che descrivono come gli agenti devono comportarsi alla ricezione dei vari messaggi, e quali messaggi devono inviare agli altri agenti o agli altri moduli. Questa struttura modulare rende possibile un alto grado di personalizzazione degli algoritmi, oltre che il riutilizzo e la manutenzione del codice.
Fra gli algoritmi supportati da FRODO, abbiamo utilizzato i seguenti: ADOPT [16], DPOP [21], SynchBB [8], MGM e MGM-2 [15].

4.1.2 Formato di input di FRODO

FRODO richiede in input due tipi di file: i file che definiscono il problema di ottimizzazione da risolvere e i file di configurazione che definiscono l'algoritmo da utilizzare.
Il formato del file utilizzato per descrivere i DCOP è basato su XCSP 2.1 [18], un formato utilizzato per rappresentare reti di vincoli tramite XML, con delle piccole estensioni necessarie a renderlo adatto ai problemi di ottimizzazione distribuita, giacché XCSP è stato progettato per rappresentare problemi di soddisfacimento di vincoli centralizzati. Il formato XCSP risultante è un soprainsieme dello standard XDisCSP 1.0 utilizzato dalla piattaforma DisCHOCO 2 [34]; questo rende possibile utilizzare tale piattaforma come generatore di problemi benchmark per FRODO.
A seconda del parser utilizzato per la lettura del problema in input, FRODO supporta rispettivamente o un sottoinsieme, o un sottoinsieme ristretto di XCSP 2.1.
La Figura 4.1 mostra un esempio di file XCSP per FRODO, utilizzando il sottoinsieme ristretto supportato dal parser *XCSPparser*. Il file consta di cinque sezioni principali:

- la sezione <*agents*> definisce gli agenti del DCOP;
- la sezione <*domains*> definisce i domini dei valori delle variabili del DCOP;
- la sezione <*variables*> definisce le variabili del DCOP, con i corrispondenti domini e agenti cui appartengono. Ogni agente può possedere più di una variabile;
- la sezione <*relations*> definisce generiche relazioni sulle variabili. Una relazione ha una corrispondenza logica con i vincoli analoga alla corrispondenza che intercorre tra i domini e le variabili: essa descrive una nozione generica su un certo numero di variabili, senza specificarne il nome; questa nozione può essere poi implementata come un vincolo, specificando le variabili coinvolte. Il parser XCSPparser supporta solo relazioni di tipo *soft* (espresso nell'attributo *semantics*), che elencano tutti i possibili valori di utilità (o costo, a seconda che l'attributo *maximize* del tag *presentation* sia *true* o *false*), e per ogni utilità, gli assegnamenti delle variabili che la producono. Tuttavia

```
<instance>
  <presentation name="sampleProblem" maxConstraintArity="2"
                maximize="false" format="XCSP 2.1_FRODO" />

  <agents nbAgents="3">
    <agent name="agentX" />
    <agent name="agentY" />
    <agent name="agentZ" />
  </agents>

  <domains nbDomains="1">
    <domain name="three_colors" nbValues="3">1..3</domain>
  </domains>

  <variables nbVariables="3">
    <variable name="X" domain="three_colors" agent="agentX" />
    <variable name="Y" domain="three_colors" agent="agentY" />
    <variable name="Z" domain="three_colors" agent="agentZ" />
  </variables>

  <relations nbRelations="1">
    <relation name="NEQ" arity="2" nbTuples="3" semantics="soft" defaultCost="0">
      infinity: 1 1|2 2|3 3
    </relation>
  </relations>

  <constraints nbConstraints="3">
    <constraint name="X_and_Y_have_different_colors" arity="2" scope="X Y" reference="NEQ" />
    <constraint name="X_and_Z_have_different_colors" arity="2" scope="X Z" reference="NEQ" />
    <constraint name="Y_and_Z_have_different_colors" arity="2" scope="Y Z" reference="NEQ" />
  </constraints>
</instance>
```

Figura 4.1: Esempio di file XCSP per FRODO (XCSPparser) [14].

l'utilizzo del valore speciale *infinity* rende possibile esprimere in questo modo anche vincoli di tipo hard. Nell'esempio in Figura 4.1, la relazione *NEQ* attribuisce un costo infinito agli assegnamenti nei quali le variabili assumono lo stesso valore, e un costo nullo a tutti gli altri assegnamenti;

- la sezione <*constraints*> elenca i vincoli veri e propri del DCOP, riferendosi alle relazioni precedentemente definite, e applicandole a variabili specifiche elencate nell'attributo *scope*.

Il parser *JaCoPxcspParser* rende possibile esprimere i vincoli con una sintassi più ricca, includendo vincoli intensionali basati su predicati, funzioni, oltre che vincoli globali. In Figura 4.2 è mostrato lo stesso esempio espresso in Figura 4.1, ma con l'utilizzo del sottoinsieme esteso di XCSP. Ci sono due differenze rispetto alla precedente rappresentazione:

```
<instance>
  <presentation name="sampleProblem" maxConstraintArity="2"
                maximize="false" format="XCSP 2.1_FRODO" />

  <agents nbAgents="3">
    <agent name="agentX" />
    <agent name="agentY" />
    <agent name="agentZ" />
  </agents>

  <domains nbDomains="1">
    <domain name="three_colors" nbValues="3">1..3</domain>
  </domains>

  <variables nbVariables="3">
    <variable name="X" domain="three_colors" agent="agentX" />
    <variable name="Y" domain="three_colors" agent="agentY" />
    <variable name="Z" domain="three_colors" agent="agentZ" />
  </variables>

  <predicates nbPredicates="1">
    <predicate name="NEQ">
      <parameters> int X1 int X2 </parameters>
      <expression>
        <functional> ne(X1, X2) </functional>
      </expression>
    </predicate>
  </predicates>

  <constraints nbConstraints="3">
    <constraint name="X_and_Y_have_different_colors" arity="2" scope="X Y" reference="NEQ" >
      <parameters> X Y </parameters>
    </constraint>
    <constraint name="X_and_Z_have_different_colors" arity="2" scope="X Z" reference="NEQ" >
      <parameters> X Z </parameters>
    </constraint>
    <constraint name="Y_and_Z_have_different_colors" arity="2" scope="Y Z" reference="NEQ" >
      <parameters> Y Z </parameters>
    </constraint>
  </constraints>
</instance>
```

Figura 4.2: Esempio di file XCSP per FRODO (JaCoPxcspParser) [14].

- la sezione <*predicates*> è l'equivalente hard intensionale delle relazioni soft estensionali. Ogni predicato dichiara una lista di parametri, preceduti dal loro tipo (per il momento è supportato unicamente il tipo intero). L'espressione funzionale è un'espressione logica dei parametri che definisce il vincolo. Sono supportate le seguenti operazioni, combinabili ricorsivamente per comporre espressioni complesse: valore assoluto, valore opposto, somma, sottrazione, moltiplicazione, divisione intera, modulo, potenza, valore minimo, valore massimo, uguaglianza, non uguaglianza, disuguaglianza, disuguaglianza stretta, NOT logico, AND logico, OR logico, XOR logico, equivalenza logica e costrutto if-then-else;
- ogni elemento <*constraint*> deve dichiarare una lista di parametri, che possono essere variabili o valori costanti, nell'ordine corrispondente in cui sono dichiarati nel predicato corrispondente.

Da notare infine che il parser JaCoPxcspParser supporta ancora le relazioni soft estensionali, così come supporta le funzioni, che sono l'equivalente soft intensionale delle relazioni soft estensionali.

FRODO necessita in input anche di un file di configurazione degli agenti che definisce l'algoritmo da utilizzare, oltre che diversi altri parametri (quando applicabili). In Figura 4.3 è mostrato un esempio di file di configurazione, che corrisponde all'implementazione classica dell'algoritmo DPOP.

Sono supportate le seguenti metriche di valutazione delle prestazioni:

- numero e dimensione dei messaggi inviati: FRODO riporta il numero totale dei messaggi inviati ordinati per tipo, così come lo spazio di memoria richiesto dal messaggio più grande, e la somma totale dello spazio richiesto dalla somma di tutti i messaggi inviati. L'utilizzo di questa metrica è però computazionalmente molto oneroso, poiché misurare la dimensione dei messaggi richiede serializzazione;

- Non-Cuncurrent Constraint Checks (NCCCs) [4]: ogni agente possiede un contatore dei controlli effettuati localmente sui vincoli. Ogni messaggio inviato agli altri agenti contiene questo valore. Quando un agente riceve un messaggio, aggiorna il proprio contatore utilizzando il valore maggiore fra quello indicato dal proprio contatore e quello indicato nel messaggio ricevuto. Il valore più grande ottenuto alla terminazione dell'algoritmo è utilizzato per ottenere una misura del lavoro di ricerca concorrente. Tale misura può essere poi utilizzata per contare il numero di NCCC (questo avviene diversamente per ogni algoritmo, in quanto dipende dalle particolari operazioni che svolgono

```
<agentDescription className = "frodo2.algorithms.SingleQueueAgent"
  measureTime = "true" measureMsgs = "false" >

  <parser parserClass = "frodo2.algorithms.XCSPparser"
          displayGraph = "false"
          utilClass = "frodo2.solutionSpaces.AddableInteger"
          countNCCCs = "false" />

  <modules>
    <module className = "frodo2.algorithms.varOrdering.dfs.DFSgenerationParallel"
      reportStats = "true" >
      <rootElectionHeuristic
        className = "frodo2.algorithms.heuristics.ScoringHeuristicWithTiebreaker" >
        <heuristic1
          className = "frodo2.algorithms.heuristics.MostConnectedHeuristic" />
        <heuristic2
          className = "frodo2.algorithms.heuristics.ScoringHeuristicWithTiebreaker" >
          <heuristic1
            className = "frodo2.algorithms.heuristics.SmallestDomainHeuristic" />
          <heuristic2
            className = "frodo2.algorithms.heuristics.VarNameHeuristic" />
        </heuristic2>
      </rootElectionHeuristic>
      <dfsGeneration className = "frodo2.algorithms.varOrdering.dfs.DFSgeneration" >
        <dfsHeuristic className =
          "frodo2.algorithms.varOrdering.dfs.DFSgeneration$ScoreBroadcastingHeuristic">
          <scoringHeuristic
            className = "frodo2.algorithms.heuristics.ScoringHeuristicWithTiebreaker" >
            <heuristic1
              className = "frodo2.algorithms.heuristics.MostConnectedHeuristic" />
            <heuristic2
              className = "frodo2.algorithms.heuristics.SmallestDomainHeuristic" />
          </scoringHeuristic>
        </dfsHeuristic>
      </dfsGeneration>
    </module>

    <module className = "frodo2.algorithms.dpop.UTILpropagation"
      reportStats = "true" />

    <module className = "frodo2.algorithms.dpop.VALUEpropagation"
      reportStats = "true" />
  </modules>

</agentDescription>
```

Figura 4.3: Esempio di file di configurazione di FRODO corrispondente all'algoritmo DPOP [14].

gli agenti), e incorpora lo sforzo computazionale locale di ogni agente. Questa metrica non è supportata dal parser JaCoPxcspParser, ma solo dal parser XCSPparser;

- tempo simulato [28]: ogni agente possiede un orologio interno. Quando invia un messaggio, l'agente aggiunge a esso un timestamp che indica il momento in cui il messaggio è stato inviato secondo il proprio orario. Quando riceve un messaggio, se il timestamp associato riporta un orario maggiore rispetto l'ora indicata dal proprio orologio interno, l'agente aggiorna il proprio orario in modo che corrisponda al timestamp del messaggio appena ricevuto. Quando l'algoritmo termina, il tempo di esecuzione è determinato dall'orario maggiore scelto fra quelli indicati dagli orologi interni di tutti gli agenti;
- altre statistiche: diversi moduli di algoritmi possono riportare altre informazioni sul problema. Ad esempio, nel caso di DPOP, il modulo di generazione DFS può riportare in formato DOT [7] l'albero DFS costruito e utilizzato dall'algoritmo, mentre il modulo di propagazione dei messaggi VALUE può riportare l'assegnamento ottimo delle variabili del DCOP e la corrispondente utilità (o costo) totale.

4.2 Modelli sviluppati

Nel tentativo di ottenere una formulazione del problema che fosse il più possibile espressiva ma allo stesso tempo efficace ed efficiente, abbiamo sviluppato diversi modelli, ognuno con peculiarità differenti.
In Appendice A sono mostrati degli esempi di problemi modellati secondo alcune formulazioni descritte di seguito, nel formato richiesto da FRODO.

4.2.1 Formulazione vettoriale

L'obiettivo da raggiungere, sviluppando questa formulazione, era di creare un modello che fosse il più generale possibile, e che desse l'opportunità di sfruttare al meglio tutti i vantaggi e le possibilità offerte dal problema applicativo formalizzato nel Capitolo 3.

Era di particolare interesse che il modello DCOP risultante comprendesse una rappresentazione esplicita del tempo, e che permettesse agli agenti che controllano gli elettrodomestici di scegliere tra profili di consumo variabili, in modo da avere il più alto grado di libertà possibile.

Per fare questo abbiamo pensato a un'ideale rappresentazione vettoriale dei domini delle variabili di decisione del problema, in cui il dominio D_{ik} di ogni variabile x_i (ognuna assegnata esattamente a un agente) è formato da una serie di vettori, ognuno dei quali rappresenta un possibile profilo di consumo $p_{ik}(t)$ espresso lungo tutta la durata dell'arco temporale di ottimizzazione; ogni elemento di ogni dominio, perciò, è un vettore che rappresenta un possibile profilo di funzionamento che un carico può assumere (o una sequenza di valori di potenza istantanei riferiti a istanti di tempo diversi in successione). Utilizzando l'esempio mostrato in Sezione 3.1, con questa rappresentazione il dominio della variabile x_1 corrispondente all'agente a_1 risulta quindi essere $D_1 = \{[4,4,4,4,4],[9,9,0,0,0],[0,9,9,0,0],[0,0,9,9,0],[0,0,0,9,9],\}$: sono così rappresentati entrambi i possibili profili di consumo che il carico associato può utilizzare; il dominio della variabile x_2 corrispondente all'agente a_2 risulta invece essere $D_2 = \{[6,6,6,6,0],[0,6,6,6,6]\}$.

Un dominio è rappresentato quindi come un vettore esattamente della dimensione del numero dei campioni nei quali è quantizzato l'arco temporale di ottimizzazione (ad esempio per un arco temporale di 24 ore, con campioni di 1 minuto, il tempo considerato è diviso in 1440 campioni, per cui anche la dimensione di ogni vettore deve essere di 1440 elementi).

Non è stato però possibile esprimere esattamente in questa maniera l'idea dei domini vettoriali nella piattaforma di riferimento. Né il formato XCSP utilizzato per esprimere le istanze DCOP, né gli algoritmi implementati in FRODO, infatti, sono stati progettati per gestire vettori: è consentito soltanto l'utilizzo del tipo intero (tranne in alcuni casi in cui è possibile utilizzare numeri reali).

Abbiamo perciò modificato la definizione del modello, in modo tale da poterlo esprimere secondo i canoni standard del framework utilizzato, pur mantenendo l'idea di partenza: invece di modellare le possibili curve di assorbimento di un elettrodomestico tramite vettori, abbiamo assegnato a ogni agente a_i un numero di variabili pari alla dimensione dell'arco temporale di riferimento T. In questo modo ogni variabile $x_{i,t}$ rappresenta il valore di potenza assorbito dal carico i-esimo nel quanto di tempo $t \in T$, e l'insieme delle variabili viste in sequenza rappresenta proprio il profilo di funzionamento variabile che volevamo esprimere attraverso i domini vettoriali. Questo formalismo ha reso inoltre necessaria la creazione di appositi vincoli che collegassero logicamente tra loro tutte le variabili di un agente.

Utilizzando sempre l'esempio precedente, con questa formulazione entrambi gli agenti hanno il controllo di 5 variabili, ognuna con un dominio differente, a seconda dei profili di funzionamento che il carico associato può assumere. Ad esempio, la variabile $x_{1,1}$ che rappresenta la potenza assorbita dal carico associato all'agente a_1 nell'istante di tempo $t = 1$ può assumere i valori del dominio $D_{1,1} = \{0, 4, 9\}$.

In Appendice A.1 è mostrato un esempio espresso tramite questa formulazione.

Formalizzazione

Possiamo dunque formalizzare il modello descritto nel modo seguente:

- $\left(x_{1,1}, \dots, x_{1,t-1}, x_{1,t}, x_{2,1}, \dots, x_{2,t-1}, x_{2,t}, x_{n,1}, \dots, x_{n,t-1}, x_{n,t}\right), t \in T$: insieme delle variabili controllate dagli n agenti. Ogni variabile $x_{i,j}$ rappresenta la potenza istantanea assorbita dal carico corrispondente all'agente i nel quanto di tempo j corrispondente. Ogni agente controlla un numero di variabili pari al numero di campioni $|T|$ in cui è suddiviso l'orizzonte temporale di ottimizzazione;
- $\left(D_{1,1}, \dots, D_{1,t-1}, D_{1,t}, D_{2,1}, \dots, D_{2,t-1}, D_{2,t}, D_{n,1}, \dots, D_{n,t-1}, D_{n,t}\right), t \in T$: insieme dei domini delle variabili. Ogni dominio contiene i possibili valori di potenza istantanea che può assumere il carico associato nel quanto di tempo corrispondente;
- T: insieme che rappresenta i time step in cui è suddiviso l'orizzonte temporale di ottimizzazione. Ad esempio, con un orizzonte temporale giornaliero (quindi di 24 ore), utilizzando un passo di campionamento di 1 minuto, si ha $T = \{1,2, \dots,1440\}$;
- vincoli vettoriali: rappresentano il collegamento logico tra le variabili controllate da un singolo agente. Costringono l'agente ad assegnare valori coerenti con un possibile profilo di funzionamento, impedendo che le variabili possano assumere un qualsiasi valore del proprio dominio. Questo vincolo è stato espresso in FRODO tramite un'espressione funzionale complessa inserita all'interno di un predicato (Figura A.1). Si tratta quindi di un vincolo hard;
- vincoli di potenza massima: per ogni quanto di tempo, la somma delle potenze assorbite dai vari elettrodomestici (indicate dalle variabili degli agenti) non deve superare la potenza massima disponibile P_a. Per esprimere questo vincolo in FRODO è stato particolarmente utile il

costrutto speciale per esprimere i vincoli globali. Il vincolo si può così modellare: $\forall t \in T, \sum_{i=1}^{n} x_{i,t} \leq P_a$;

- funzioni di costo: per ogni agente, il costo associato si può rappresentare come la somma dei singoli contributi in ogni campione di tempo: $\forall i \in [1, n], f_i = \sum_{t \in T} a_t$, dove $a_t = x_{i,t} \cdot c_t$, con c_t che indica il costo della tariffa al tempo t. Per esprimere tale funzione di costo, abbiamo specificato un vincolo per ogni quanto di tempo considerato, ognuno associato alla stessa funzione di costo. Sarebbe stato possibile definire un solo vincolo per ogni agente, ma ciò avrebbe causato un deciso aumento della complessità della funzione di costo associata, la quale avrebbe necessitato di un elevato numero di parametri (pari a $|T|$), determinando così un notevole aumento nella complessità generale del problema;
- funzione obiettivo: la funzione obiettivo da minimizzare è la somma delle funzioni di costo precedentemente riportate: $X^* = \arg min_{\bar{X}} \sum_{i=1}^{n} f_i$. Non è necessario indicarlo esplicitamente in FRODO, poiché utilizza automaticamente come funzione obiettivo da ottimizzare la somma di tutte le funzioni di costo indicate.

Considerazioni e commenti

La formulazione vettoriale presentata ha il vantaggio di essere particolarmente intuitiva e semplice da implementare soprattutto per quanto riguarda la stesura dei vincoli. Inoltre, come già sottolineato in precedenza, permette di sfruttare appieno le possibilità offerte dalla formulazione del problema applicativo (compresa la possibilità di modellare vincoli utente che definiscano delle preferenze circa la ricerca della soluzione).

Il limite emerso sin da subito, e confermato successivamente in fase di test preliminari, riguarda la complessità introdotta dalla rappresentazione

vettoriale. Ogni vettore (o pseudo-vettore, modellato dai vincoli appositi come sequenza di variabili di decisione) è composto da un numero di elementi pari al numero dei campioni nei quali è quantizzato l'arco temporale. Perciò, per includere ogni possibile profilo di funzionamento di un elettrodomestico, bisogna generare per ogni carico un numero molto elevato di vettori diversi. Facciamo un esempio: con un arco temporale di 24 ore, e campionamento di 1 minuto, ogni vettore è composto da 1440 elementi. Nel caso più semplice di un elettrodomestico con un singolo profilo di utilizzo fissato, l'agente associato deve scegliere tra $(1440 - durata\ del\ ciclo)$ vettori da 1440 elementi (il primo vettore rappresenta l'accensione del carico in $t = 1$, il secondo rappresenta l'accensione in $t = 2$, e così via). Si può quindi creare un'esplosione dello spazio di ricerca che potrebbe rendere impossibile risolvere il problema in tempi ragionevoli.

Questo limite è apparso evidente in seguito ad alcuni test preliminari impostati con profili di consumo fissati (al fine di mantenere la complessità minore possibile) ed effettuati con gli algoritmi ottimi di riferimento per il lavoro di tesi (ADOPT, DPOP e SynchBB): utilizzando un campionamento di 1 ora, su un orizzonte temporale di 12 ore, con 2 agenti attivi (quindi 12 variabili per agente, 24 variabili totali), i tempi di risoluzione si sono attestati nel migliore dei casi nell'ordine di qualche secondo; con un campionamento di 1 ora, su un orizzonte temporale di 12 ore, con 3 agenti attivi (quindi 12 variabili per agente, 36 variabili totali), i tempi di risoluzione sono aumentati nel migliore dei casi fino a 20 minuti; aumentando l'orizzonte temporale a 24 ore, mantenendo il campionamento di 1 ora, con 2 agenti attivi (quindi 24 variabili per agente, 48 variabili totali), il problema è esploso in complessità, causando il riempimento della memoria allocata dalla JVM e il fallimento dei test. Per completezza abbiamo effettuato quest'ultima tipologia di test anche utilizzando MB-DPOP [22], un algoritmo ottimo ideato appositamente per affrontare problemi il cui grafo delle dipendenze è molto ampio, ma nonostante l'algoritmo sia riuscito a gestire l'elevata richiesta di memoria, non

è riuscito a trovare alcuna soluzione entro il tempo di timeout impostato a 4 ore.

Gli stessi test hanno dato esito negativo anche quando sono stati impiegati algoritmi non ottimi (MGM, MGM-2): sebbene i problemi siano stati risolti sempre in tempi brevi (nell'ordine di alcuni secondi), le soluzioni trovate sono state generalmente non ammissibili (di costo infinito), in quanto in violazione dei vincoli hard; solo nei problemi di minima complessità gli algoritmi hanno trovato soluzioni sub-ottime nel 20% dei casi.

Questi risultati ci hanno portato ad abbandonare il modello basato sui vettori, poiché di complessità computazionale eccessivamente elevata per risolvere problemi realistici.

4.2.2 Formulazione ad attività

In seguito ai risultati ottenuti con il modello descritto in Sezione 4.2.1, abbiamo cercato di sviluppare una formulazione che ne mantenesse i vantaggi, ma che ne riducesse nel frattempo la complessità, giungendo così alla seguente formulazione.

L'idea che ne sta alla base è completamente differente, così come ciò che le variabili di decisione del problema rappresentano. Nel modello descritto in precedenza, infatti, esse indicano la potenza assorbita dal relativo carico in un dato istante di tempo, e ogni agente controlla un numero di variabili pari ai campioni nel quale è diviso l'arco temporale di ottimizzazione. Con questa formulazione invece, a ogni agente sono associati tre tipi di variabili: una indica l'istante d'inizio di un'*attività*, una indica la durata di tale attività, e l'altra ne indica la potenza assorbita. Con il termine attività intendiamo una fase del profilo di consumo di un carico elettrico che richieda un assorbimento di potenza (all'incirca) costante. Ad esempio, nel profilo di consumo mostrato in Figura 4.4 si possono individuare tre attività, ognuna caratterizzata da una certa durata e un certo assorbimento di potenza. Ogni agente, quindi, controlla

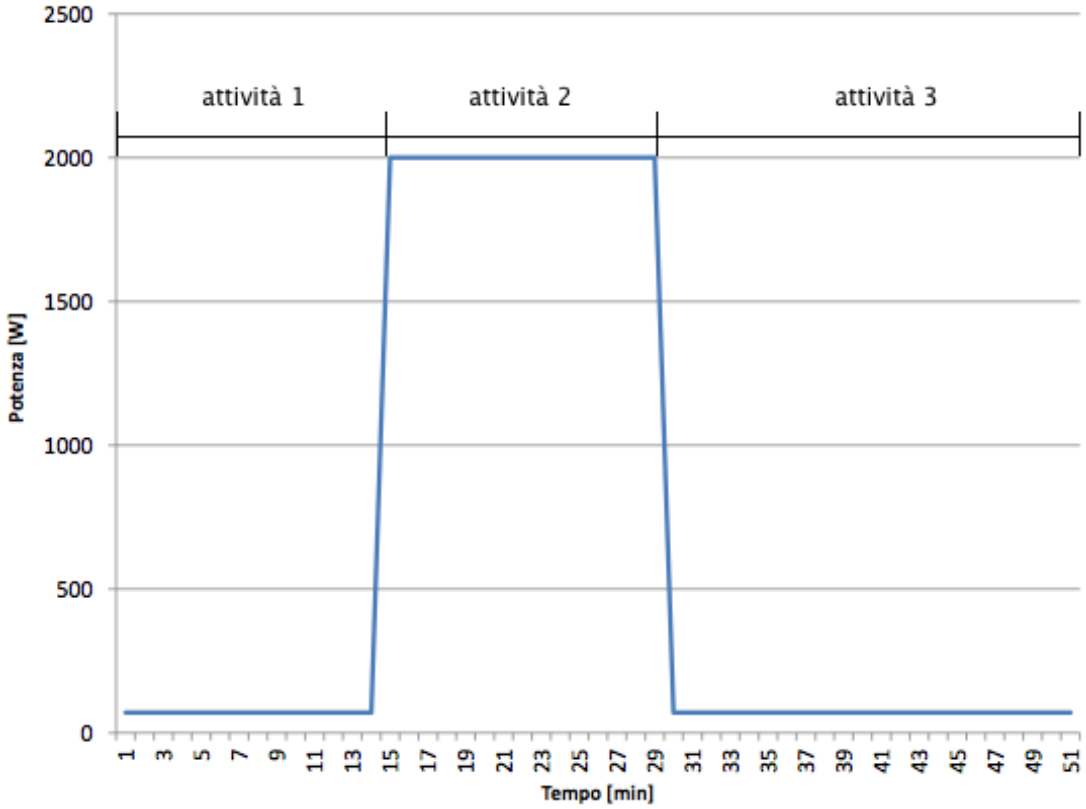

Figura 4.4: Suddivisione ad attività di un profilo di consumo.

un numero di variabili proporzionale al numero di attività che caratterizzano il profilo di consumo del carico associato. Proseguendo l'esempio in Figura 4.4, un agente associato al profilo mostrato controllerebbe in totale 9 variabili, esattamente 3 per ogni attività (istante di inizio dell'attività, durata dell'attività, potenza assorbita dall'attività). Questa formulazione permette ancora di modellare profili di consumo variabili agendo sui domini delle potenze e delle durate delle attività, ma riduce notevolmente il numero di variabili in gioco, giacché un elettrodomestico è generalmente caratterizzato da un basso numero di attività (tipicamente 3-5).

In Appendice A.2 è mostrato un esempio espresso tramite questa formulazione.

Formalizzazione

Possiamo dunque formalizzare il modello descritto nel modo seguente:

- $\left(x_{1,a_1}, \dots, x_{1,a_{m_1}}, x_{1,d_1}, \dots, x_{1,d_{m_1}}, x_{1,p_1}, \dots, x_{1,p_{m_1}}, \dots, x_{n,a_1}, \dots, x_{n,a_{m_n}}, x_{n,d_1}, \dots, x_{n,d_{m_n}}, \dots, x_{n,p_1}, \dots, x_{n,p_{m_n}}\right)$: insieme delle variabili controllate dagli n agenti, dove m_1 e m_n indicano dei generici numeri di attività dei profili di consumo associati. Ogni agente controlla tre variabili per ogni attività: x_{i,a_j} rappresenta l'istante d'inizio dell'attività j-esima dell'agente i-esimo, x_{i,d_j} rappresenta la durata dell'attività j-esima dell'agente i-esimo, x_{i,p_j} rappresenta la potenza assorbita dall'attività j-esima dell'agente i-esimo;
- $\left(D_{1,a_1}, \dots, D_{1,a_{m_1}}, \dots, D_{n,a_1}, \dots, D_{n,a_{m_n}}\right)$: insieme dei domini delle variabili di inizio attività. Ogni dominio contiene i possibili istanti d'inizio di un'attività. La cardinalità del dominio dipende dal campionamento dell'orizzonte temporale T, dalla durata di ogni attività dell'agente e dalla durata complessiva del profilo di consumo;
- $\left(D_{1,d_1}, \dots, D_{1,d_{m_1}}, \dots, D_{n,d_1}, \dots, D_{n,d_{m_n}}\right)$: insieme dei domini delle variabili di durata attività. Ogni dominio contiene le possibili durate dell'attività corrispondente;
- $\left(D_{1,p_1}, \dots, D_{1,p_{m_1}}, \dots, D_{n,p_1}, \dots, D_{n,p_{m_n}}\right)$: insieme dei domini delle variabili di potenza attività. Ogni dominio contiene i possibili valori di potenza assorbita durante un'attività;
- vincoli di terminazione entro la fine dell'orizzonte temporale di ottimizzazione: il modello è stato concepito per permettere un'ottimizzazione su un arco temporale definito. È necessario quindi che ogni attività termini entro la fine dell'orizzonte temporale. Ciò si traduce in un vincolo diretto sui domini delle variabili di inizio attività, i cui valori devono essere settati in maniera opportuna. Il vincolo si può

così modellare: $\forall i \in [1,n], x_{i,a_1} + Duration_i < T_{tot}$, dove $Duration_i$ indica la durata totale del ciclo di funzionamento del carico i-esimo definita come $\sum_j x_{i,d_j}$, e T_{tot} indica la durata totale dell'arco temporale di ottimizzazione;

- vincoli di scelta profilo di consumo: rappresentano il collegamento logico tra le variabili di durata e le variabili di potenza delle attività. Costringono l'agente ad assegnare valori coerenti con un possibile profilo di consumo, impedendo che le variabili possano assumere un qualsiasi valore del proprio dominio. Questo vincolo è stato espresso in FRODO tramite un'espressione funzionale complessa inserita all'interno di un predicato (Figura A.2). Si tratta quindi di un vincolo hard;
- vincoli di continuità tra le attività: rappresentano il collegamento logico tra le variabili di inizio attività. Costringono l'agente ad assegnare come inizio di un'attività l'istante successivo alla fine dell'attività precedente relativa allo stesso carico (oppure un istante entro un determinato intervallo dalla fine dell'attività precedente, se si vuole aggiungere un ulteriore grado di libertà). Questo vincolo è stato espresso in FRODO tramite un'espressione funzionale complessa inserita all'interno di un predicato (si tratta quindi di un vincolo hard). Il vincolo si può così modellare: $\forall i \in [1,n], \forall j \in [2,m], x_{i,a_j} = x_{i,a_{j-1}} + x_{i,d_{j-1}}$, dove m indica un generico numero di attività;
- vincoli di potenza massima: per ogni quanto di tempo, la somma delle potenze assorbite associate alle attività attive in quel quanto di tempo non deve superare la potenza massima disponibile P_a. Questo vincolo è stato espresso in FRODO attraverso l'utilizzo di vincoli hard enunciati tramite relazioni soft, assegnando un costo infinito alle tuple vietate, ossia alle combinazioni di variabili attività, durata e potenza di agenti diversi, la cui somma delle potenze assorbite è maggiore della potenza

massima disponibile, e i cui istanti d'inizio e durate associate indicano una sovrapposizione temporale. La scelta di esplicitare le combinazioni di valori vietate anziché quelle permesse è stata effettuata per questioni di efficienza, in quanto il numero delle combinazioni vietate è generalmente di molto inferiore al numero delle combinazioni permesse;

- vincoli di non sovrapposizione dei cicli di esecuzione dei carichi: ogni ciclo di esecuzione di un carico è stato modellato da un agente dedicato. Ciò significa che è possibile che si verifichi uno scenario in cui più agenti gestiscono lo stesso elettrodomestico, per cui i relativi cicli di esecuzione devono essere eseguiti in maniera seriale. Questo vincolo è stato espresso in FRODO attraverso l'utilizzo di espressioni funzionali inserite in predicati (si tratta quindi di vincoli hard). Il vincolo si può così modellare: $\forall i,j \in [1,n], t.c.\ i \neq j, se\ Type_i = Type_j\ allora$:

$$\begin{cases} x_{i,a_1} + Duration_i < x_{j,a_1}\ sse\ x_{i,a_1} < x_{j,a_1} \\ x_{i,a_1} > x_{j,a_1} + Duration_j\ sse\ x_{i,a_1} > x_{j,a_1} \end{cases}$$

  dove $Type_i$ indica la tipologia dell'elettrodomestico i-esimo, e $Duration_i$ indica la durata totale del ciclo di funzionamento del carico i-esimo;

- vincoli di quanto massimo di terminazione: si tratta di vincoli opzionali a disposizione dell'utente, per permettergli di decidere l'istante entro il quale far terminare il ciclo di funzionamento di un elettrodomestico. Si traduce in un vincolo diretto sui domini delle variabili d'inizio attività, atto a restringerne l'intervallo di valori disponibili. Il vincolo si può così modellare: $\forall i \in [1,n],\ x_{i,a_1} + Duration_i < T_{max}$, dove $Duration_i$ indica la durata totale del ciclo di funzionamento del carico i-esimo, e T_{max} indica l'istante massimo di terminazione segnalato dall'utente;

- vincoli di delay fissato: si tratta di vincoli opzionali a disposizione dell'utente, per consentirgli di stabilire un'ottimizzazione che sposti gli istanti di inizio entro un range limitato, rispetto a una situazione di partenza. Si traduce in un vincolo diretto sui domini delle variabili d'inizio attività, atto a restringerne l'intervallo di valori disponibili. Il vincolo si può così modellare: $\forall i \in [1,n],\ x_{i,a_1} < StartTime_i + Delay_i \wedge x_{i,a_1} > StartTime_i - Delay_i$, con $Delay_i > 0 \wedge StartTime_i - Delay_i \geq 0 \wedge StartTime_i + Duration_i + Delay_i < T_{tot}$, dove $StartTime_i$ indica l'istante di avvio del carico i-esimo nello scenario non ottimizzato, $Delay_i$ indica l'ampiezza dell'intervallo di ricerca fissato per il carico i-esimo, $Duration_i$ indica la durata totale del ciclo di funzionamento del carico i-esimo, e T_{tot} indica la durata totale dell'arco temporale di ottimizzazione;
- funzioni di costo: per ogni agente, il costo associato f_i, con $i \in [1,n]$, equivale alla somma dei contributi di costo generati da ogni attività. Essi dipendono dalle fasce tariffarie in cui le attività degli agenti vengono eseguite, e si possono calcolare come il prodotto tra il valore campionato della potenza e il costo corrispondente della tariffa, pesato secondo la durata dell'attività;
- funzione obiettivo: la funzione obiettivo da minimizzare è la somma delle funzioni di costo di ogni agente: $X^* = \arg min_{\bar{X}} \sum_{i=1}^{n} f_i$. Non è necessario indicarlo esplicitamente in FRODO, poiché utilizza automaticamente come funzione obiettivo da ottimizzare la somma di tutte le funzioni di costo indicate.

Considerazioni e commenti

Con la formulazione presentata, l'obiettivo di diminuire la complessità dei problemi, pur mantenendo la maggior espressività possibile, è stato parzialmente raggiunto. Così come nel modello precedente, infatti, è presente una rappresentazione esplicita del tempo (sono le variabili di decisione stesse a rappresentare il tempo tramite gli istanti di inizio delle attività e le relative durate), ed è possibile modellare profili di consumo variabili dei carichi elettrici, aumentando così la libertà di scelta degli agenti e le opportunità di ottimizzare al meglio gli scenari di utilizzo. È inoltre molto semplice esprimere vincoli che rappresentino le preferenze dell'utente riguardo all'avvio degli elettrodomestici durante l'arco temporale di ottimizzazione.

Il limite maggiore di questa formulazione risiede invece nell'espressione dei vincoli per il rispetto della potenza massima disponibile, che risulta particolarmente complessa. Come già specificato, infatti, tali vincoli sono dichiarati tramite tuple di valori che coinvolgono tre variabili (istante di inizio, durata, potenza assorbita) per ogni agente nel sistema. Per ogni possibile combinazione di attività fra agenti diversi, dunque, è necessario imporre un vincolo che regoli i tempi di esecuzione delle attività degli agenti, in modo che esse, quando in esecuzione, non assorbano più potenza di quella disponibile. Ogni vincolo, inoltre, coinvolge un numero di variabili che cresce proporzionalmente rispetto al numero di agenti nel sistema.

Facciamo un esempio: poniamo di avere 2 agenti (A_1, A_2) ; ogni elettrodomestico è caratterizzato da un profilo di consumo (non variabile) composto da 2 attività, quindi ogni agente controlla 2 variabili che indicano l'istante di inizio di tali attività $(x_{1,a_1}, x_{1,a_2}, x_{2,a_1}, x_{2,a_2})$, 2 variabili che ne indicano la durata $(x_{1,d_1}, x_{1,d_2}, x_{2,d_1}, x_{2,d_2})$ e 2 variabili che ne indicano la potenza assorbita $(x_{1,p_1}, x_{1,p_2}, x_{2,p_1}, x_{2,p_2})$. Dato questo scenario, esistono 4 combinazioni di attività fra agenti diversi (att_{11}-att_{21}, att_{11}-att_{22}, att_{12}-att_{21}, att_{12}-att_{22}), perciò è necessario dichiarare 4 vincoli per il rispetto della potenza

massima, ognuno per esprimere la necessità che due attività siano eseguite in tempi diversi, nel caso in cui la somma delle rispettive potenze assorbite fosse maggiore della potenza totale disponibile. Ogni vincolo, inoltre, coinvolge tutte le variabili relative alle attività interessate (istante di inizio, durata, potenza assorbita). In questo caso, quindi, ogni vincolo coinvolge 6 variabili in totale.

Come si può dedurre dall'esempio, il numero di vincoli di potenza, così come la loro complessità, aumenta rapidamente al crescere del numero degli agenti e delle attività.

I test preliminari hanno confermato queste considerazioni: con problemi molto semplici (2 agenti, ognuno con profili di consumo formati da 1-2 attività), gli algoritmi di riferimento hanno trovato una soluzione in tempi accettabili (pochi secondi, al massimo pochi minuti). Aumentando di poco il numero di variabili (2 agenti, ognuno con profili di consumo formati da 4-5 attività), invece, i problemi sono esplosi in complessità, causando il riempimento della memoria allocata dalla JVM e il fallimento dei test.

Questi risultati ci hanno spinto a creare una formulazione del modello più semplice, mantenendo l'idea di base delle attività, ma eliminando alcuni gradi di libertà concessi agli agenti.

4.2.3 Formulazione ad attività semplificata

Questa formulazione è il risultato di uno snellimento del modello precedente. Ogni agente controlla, per ogni attività che caratterizza il profilo di funzionamento del carico associato, i tre tipi di variabili descritti per il modello di partenza (istante d'inizio, durata, potenza assorbita). Assegnando però un unico valore al dominio delle variabili di durata e potenza assorbita delle attività (rendendole così assimilabili a costanti), è possibile diminuire di molto la complessità dei problemi, pur sacrificando parte dell'espressività del problema. Così facendo, infatti, il modello non è più in grado di rappresentare

profili di consumo variabili; tuttavia, essendo i profili di consumo costanti, è possibile esprimere in maniera più semplice i vincoli di potenza massima disponibile, rendendo così il modello applicabile a scenari realistici.
In Appendice A.3 è mostrato un esempio espresso tramite questa formulazione.

Formalizzazione

Possiamo dunque formalizzare il modello descritto nel modo seguente:

- $\left(x_{1,a_1}, \dots, x_{1,a_{m_1}}, x_{1,d_1}, \dots, x_{1,d_{m_1}}, x_{1,p_1}, \dots, x_{1,p_{m_1}}, \dots, x_{n,a_1}, \dots, x_{n,a_{m_n}}, x_{n,d_1}, \dots, x_{n,d_{m_n}}, \dots, x_{n,p_1}, \dots, x_{n,p_{m_n}}\right)$: insieme delle variabili controllate dagli n agenti, come in Sezione 4.2.2. Sebbene le durate e le potenze assorbite siano modellate come variabili, possiamo considerarle in realtà valori costanti, in quanto i rispettivi domini contengono un unico valore;
- $\left(D_{1,d_1}, \dots, D_{1,d_{m_1}}, \dots, D_{n,d_1}, \dots, D_{n,d_{m_n}}\right)$: insieme dei domini delle variabili di inizio attività, come in Sezione 4.2.2;
- $\left(D_{1,d_1}, \dots, D_{1,d_{m_1}}, \dots, D_{n,d_1}, \dots, D_{n,d_{m_n}}\right)$: insieme dei domini delle variabili di durata attività, come nella Sezione 4.2.2. Ogni dominio contiene un unico valore che indica la durata dell'attività corrispondente;
- $\left(D_{1,p_1}, \dots, D_{1,p_{m_1}}, \dots, D_{n,p_1}, \dots, D_{n,p_{m_n}}\right)$: insieme dei domini delle variabili di potenza attività, come in Sezione 4.2.2. Ogni dominio contiene un unico valore che indica la potenza assorbita dall'attività corrispondente;
- vincoli di terminazione entro la fine dell'orizzonte temporale di ottimizzazione: come in Sezione 4.2.2;
- vincoli di continuità tra le attività: come in Sezione 4.2.2;

- vincoli di potenza massima: come in Sezione 4.2.2. L'utilizzo di valori di potenza costanti ha permesso di eliminare dal vincolo il coinvolgimento delle variabili di potenza, in quanto i profili di consumo sono conosciuti a priori (Figura A.3);
- vincoli di non sovrapposizione dei cicli di esecuzione dei carichi: come in Sezione 4.2.2;
- vincoli di quanto massimo di terminazione: come in Sezione 4.2.2;
- vincoli di delay fissato: come in Sezione 4.2.2;
- funzioni di costo: come in Sezione 4.2.2;
- funzione obiettivo: come in Sezione 4.2.2.

Considerazioni e commenti

Come già anticipato, questa formulazione non descrive nella sua interezza il problema applicativo considerato, giacché è privata della possibilità di modellare profili di consumo variabili dei carichi elettrici, rendendo costanti le variabili di durata attività e di potenza assorbita. Quest'accorgimento, tuttavia, permette innanzitutto di diminuire lo spazio di ricerca delle soluzioni, pur mantenendo il modello in grado di rappresentare realisticamente il problema applicativo considerato. Inoltre, poter considerare come costanti le suddette variabili consente di ridurre notevolmente la complessità dei vincoli di potenza. Infatti, conoscere in anticipo i profili di consumo associati agli agenti permette di sapere a priori quali attività, se eseguite nello stesso periodo di tempo, assorbirebbero una potenza superiore alla soglia massima. Grazie a questa conoscenza pregressa, è possibile indicare all'interno della descrizione di un problema da risolvere soltanto i vincoli relativi a tali attività, diminuendone notevolmente il numero, e riducendo di conseguenza la complessità del problema. È possibile inoltre eliminare il coinvolgimento delle variabili di potenza da tali vincoli, in modo da diminuirne la cardinalità,

riducendo ulteriormente la complessità del problema. Al contrario, benché sia possibile, in quanto anch'esse sono in realtà delle costanti, non è conveniente eliminare dai vincoli di potenza massima le variabili di durata attività: poiché tali variabili sono coinvolte anche in altri tipi di vincoli (ad esempio nei vincoli di continuità tra le attività), eliminare il loro coinvolgimento nei vincoli di potenza non riduce lo spazio di ricerca della soluzione. Mantenerne il coinvolgimento in tali vincoli, invece, permette di originare problemi maggiormente vincolati, con meno possibili strade da esplorare, e quindi meno complessi. Ciò è stato reso evidente da alcuni test di prova: eliminando dai vincoli di potenza il coinvolgimento delle variabili di durata attività, infatti, i tempi di risoluzione sono cresciuti in maniera esponenziale. Abbiamo perciò scelto di conservare tali variabili nella definizione dei vincoli di potenza massima.

Dati i risultati positivi ottenuti con i test preliminari, abbiamo deciso di sperimentare questo modello in maniera approfondita e sistematica con scenari realistici. Per la discussione dei risultati ottenuti si rimanda al Capitolo 5.

4.2.4 Formulazione Dynamic DCOP

Nella formulazione già accennata in Sezione 3.1 e presentata in [3], il problema di distribuzione di potenza fra carichi elettrici è stato affrontato con un modello tale da rendere necessaria la generazione di una nuova istanza DCOP per ogni istante di tempo considerato. Ad esempio, per risolvere un problema di ottimizzazione giornaliera con questo modello, utilizzando un passo di campionamento di 1 minuto, bisogna generare 1440 istanze DCOP, ognuna indipendente dall'altra.

Prendendo spunto da quest'idea, abbiamo sviluppato un'estensione del modello, formulandolo come Dynamic DCOP (DDCOP) [35]. Definiamo come DDCOP una sequenza di DCOP (statici), che differiscono tra di loro per alcuni parametri (ad esempio vincoli o domini di variabili). Risolvere un

DDCOP significa quindi trovare una soluzione ottima per tutti i DCOP appartenenti a tale sequenza.

Formalizzazione

Possiamo dunque formalizzare il modello descritto in modo seguente:

- T: insieme che rappresenta l'orizzonte temporale di ottimizzazione. A ogni time step è associata una singola istanza della sequenza di DCOP;
- $(x_1, x_2, \dots, x_n)$: insieme delle variabili controllate dagli n agenti. Indicano la potenza assorbita all'istante corrente dal carico associato alla variabile. Ad ogni agente è associata esattamente una variabile;
- $(D_1, D_2, \dots, D_n)$: insieme dei domini delle variabili di decisione. Ogni dominio assume la forma: $D_i = \{0, \dots, X_i\}, i \in [1, n]$, con X_i che indica la potenza massima che può assorbire il carico i-esimo all'istante corrente. I valori presenti all'interno del dominio di ogni carico sono strettamente dipendenti dal carico stesso, ed è possibile modellare diversi comportamenti assegnando in maniera appropriata tali insiemi. Ad esempio è possibile modellare un carico che non può essere spento eliminando il valore "0" dal dominio, oppure un carico che può funzionare solo al massimo della potenza impostandolo come unico valore ammissibile, o ancora, modificando il dominio da un time step all'altro, è possibile modellare profili di consumo variabili;
- vincolo di potenza massima: in ogni istante di tempo, la somma delle potenze assorbite dai vari elettrodomestici (indicate dalle variabili degli agenti), non deve superare la potenza massima disponibile P_a. Il vincolo si può modellare nel modo seguente: $\sum_{i=1}^{n} x_i \leq P_a$;
- funzioni di costo: possiamo esprimere le funzioni di costo degli agenti utilizzando una stima del risparmio ottenuto anticipando o posticipando le accensioni dei carichi [29]: $f(x_i, x_j) = \frac{cost_i - savings_i}{cost_i} + \frac{cost_j - savings_j}{cost_j}$,

dove $cost_k = x_k \cdot c_t$ è il costo della potenza assorbita dal carico k-esimo, e $savings_k = x_k \cdot (c_{t+w} - c_t)$ è una stima del risparmio che si ottiene non posticipando di un tempo w l'utilizzo del carico k-esimo. Impostando in maniera appropriata il parametro w è quindi possibile analizzare i vantaggi e gli svantaggi derivanti dallo spostamento dell'avvio dei carichi di un intervallo di tempo desiderato;

- funzione obiettivo: la funzione obiettivo da minimizzare si può così formalizzare: $X^* = \arg min_{\bar{X}} \sum_{x_i,x_j} f(x_i, x_j)$.

Considerazioni e commenti

La formulazione descritta risulta essere piuttosto snella, ma occorre fare alcune precisazioni. Per esprimere correttamente il profilo di consumo di un carico elettrico, è necessario che a ogni istante di tempo (ossia ad ogni nuova istanza statica DCOP) i domini delle variabili di decisione siano impostati opportunamente. Per sfruttare correttamente questa formulazione, quindi, è necessario un modulo software esterno che conosca le caratteristiche di consumo dei carichi collegati e che assegni i domini delle variabili controllate dagli agenti in maniera appropriata a soddisfare le loro richieste energetiche.
Tale modulo (una sorta di "updater") deve agire in tempo reale a valle dell'algoritmo di risoluzione, in modo tale da aggiornare opportunamente, per ogni nuova istanza DCOP, i domini delle variabili controllate dagli agenti.
Non è stato oggetto di questo lavoro di tesi testare in maniera empirica la bontà di questa formulazione, in quanto attualmente non sono disponibili liberamente implementazioni di algoritmi DDCOP nei framework utilizzati per la risoluzione di problemi di ottimizzazione distribuita. In [3] sono mostrati dei test effettuati con diversi algoritmi di risoluzione su istanze statiche di DCOP, e i risultati ottenuti hanno evidenziato tempi di risoluzione

piuttosto soddisfacenti, soprattutto in seguito allo sfruttamento di alcune proprietà delle relazioni, e di tecniche di ordinamento degli agenti.

In [35] è stata proposta una procedura applicabile a due varianti dell'algoritmo ADOPT, che permette di riutilizzare durante la risoluzione di un'istanza DCOP le informazioni ottenute durante la soluzione dell'istanza precedente nella sequenza. Il vantaggio di quest'approccio risiede nel fatto che risolvere un DDCOP così strutturato risulta sicuramente non più difficile che risolvere singolarmente ogni istanza DCOP nella sequenza. Inoltre, i benefici di quest'approccio sono più evidenti quando nella sequenza di DCOP statici sono presenti istanze uguali o con lievi cambiamenti. Infine, maggiore è lo spazio dedicato per memorizzare le informazioni riutilizzate, maggiori sono i benefici in termini di tempi di risoluzione.

Possiamo quindi aspettarci che utilizzando l'approccio DDCOP proposto, i tempi di risoluzione possano essere, nel peggiore dei casi, al massimo uguali ai tempi ottenuti risolvendo singolarmente i DCOP statici appartenenti a una stessa sequenza di problemi.

Capitolo 5

Risultati sperimentali e valutazioni

I test sperimentali mostrati in questo capitolo sono stati effettuati utilizzando il modello ad attività semplificato descritto in Sezione 4.2.3 e gli algoritmi di risoluzione descritti in Sezione 2.3, eseguendo FRODO su un singola macchina.
Per produrre in maniera sistematica le istanze DCOP oggetto dei test abbiamo sviluppato un *generatore di problemi*.

5.1 Generatore di problemi

Il generatore di problemi è un software (scritto in linguaggio C#) che riceve in input tre file: *scenario*, che definisce le informazioni riguardanti uno scenario abitativo e le eventuali preferenze dell'utente, *consumi*, che descrive le informazioni relative ai profili di consumo degli elettrodomestici, e *tariffe*, che contiene il profilo tariffario applicato allo scenario. L'output generato è un file XML che rappresenta il problema come un DCOP, nel formato richiesto da FRODO e specificato in Sezione 4.1.2.
Il file scenario è una tabella (Figura 5.1) creata dal *generatore di scenari*

Day	Appliance	Start_time	Delta_delay	Tmax
0	A2	1165	-2	1440
0	A2	986	-2	1440
1	A1	385	-2	1440
1	A2	1230	-2	1440
1	A2	1078	-2	1440
2	A2	1117	-2	1440
3	A2	448	-2	1440

Figura 5.1: Esempio di file scenario.

descritto in [17], utilizzando due diversi modelli statistici sviluppati sfruttando dati reali raccolti sulla popolazione. In accordo con tali modelli, l'output del programma fornisce delle informazioni riguardo a scenari domestici di consumo energetico, relativamente all'utilizzo di due elettrodomestici: lavatrice e lavastoviglie. La scelta di questi due apparecchi è stata determinata principalmente da due considerazioni: sono caratterizzati da un elevato consumo energetico, e sono degli elettrodomestici su cui è possibile compiere un'ottimizzazione minimizzando i disagi per l'utente (è impensabile, ad esempio, spostare in fasce orarie notturne l'avvio di un forno utilizzato per cucinare, al fine di diminuire i costi in bolletta). Gli scenari prodotti e utilizzati per i test sperimentali, dunque, rappresentano un valore aggiunto per una quantificazione concreta dei risultati ottenuti.

Il file scenario contiene i seguenti campi:

- *Day*: indica il giorno della settimana in cui il ciclo è avviato;

```
Appliance       A1
CycleDuration   125
NbActivities    5
ActivityDuration        10 15 85 10 5
InstantPower    62 2120 140 447 88
Appliance       A2
CycleDuration   145
NbActivities    5
ActivityDuration        10 40 70 15 10
InstantPower    64 2047 148 314 678
Appliance       B1
CycleDuration   100
NbActivities    5
ActivityDuration        15 20 40 20 5
InstantPower    70 2000 70 2000 70
Appliance       B2
CycleDuration   135
NbActivities    5
ActivityDuration        15 25 20 60 15
InstantPower    2069 72 2067 71 2073
Appliance       B3
CycleDuration   115
NbActivities    5
ActivityDuration        15 15 60 15 10
InstantPower    70 2000 70 2000 70
```

Figura 5.2: Esempio di file consumi.

- *Appliance*: indica la tipologia di elettrodomestico e il programma di funzionamento del ciclo;
- *Start_time*: rappresenta l'istante di avvio del ciclo espresso in minuti;
- *Delta_delay*: indica la presenza di vincoli utente relativi allo spostamento dell'inizio di attività;
- *Tmax*: indica, se richiesto dall'utente, l'istante massimo di terminazione del ciclo espresso in minuti.

Il file consumi è una tabella (Figura 5.2) che descrive i profili di consumo associati ai programmi di funzionamento degli elettrodomestici. Per ogni programma di funzionamento sono presenti i seguenti campi:

- *Appliance*: indica la tipologia di elettrodomestico e il programma di funzionamento del ciclo;
- *CycleDuration*: indica la durata del ciclo espressa in minuti;

START	COST
0	0,070
60	0,065
120	0,063
180	0,061
240	0,061
300	0,061
360	0,062
420	0,063
480	0,062
540	0,057
600	0,054
660	0,047
720	0,047
780	0,039
840	0,054
900	0,062
960	0,065
1020	0,075
1080	0,084
1140	0,084
1200	0,075
1260	0,071
1320	0,065
1380	0,063

Figura 5.3: Esempio di file tariffe.

- *NbActivities*: indica il numero di attività che caratterizzano il profilo di consumo;
- *ActivityDuration*: indica la durata di ciascuna attività espressa in minuti;
- *InstantPower*: indica la potenza istantanea, espressa in Watt, richiesta da ciascun'attività;

Il file tariffe è una tabella (Figura 5.3) che descrive il piano tariffario associato al problema di ottimizzazione. Per i test effettuati abbiamo considerato la tariffa estratta dal MGP relativa al giorno venerdì 1 novembre 2013. Il file è composto da un elenco di coppie:

- *START*: indica il quanto di tempo, espresso in minuti, a partire dal quale si applica il corrispondente costo della tariffa;
- *COST*: è il costo della tariffa associata al quanto di tempo indicato, espresso in Euro.

5.2 Ottimizzazione con *n*=2 elettrodomestici

La prima categoria di test è stata effettuata utilizzando 10 scenari caratterizzati dalla presenza di due elettrodomestici (lavastoviglie e lavatrice), i quali possono essere avviati, nell'arco della giornata oggetto di ottimizzazione, per un numero variabile di cicli dipendente dalla configurazione dello scenario. Il numero degli agenti presenti nel sistema, quindi, non è necessariamente uguale a due: si ricorda, infatti, che a ogni ciclo di funzionamento di un carico è associato un agente diverso. Il passo di campionamento utilizzato per la definizione dei domini è di 30 minuti. Ogni istanza DCOP associata ad uno scenario è stata risolta tramite gli algoritmi presentati, impostando il timeout a 30 minuti.

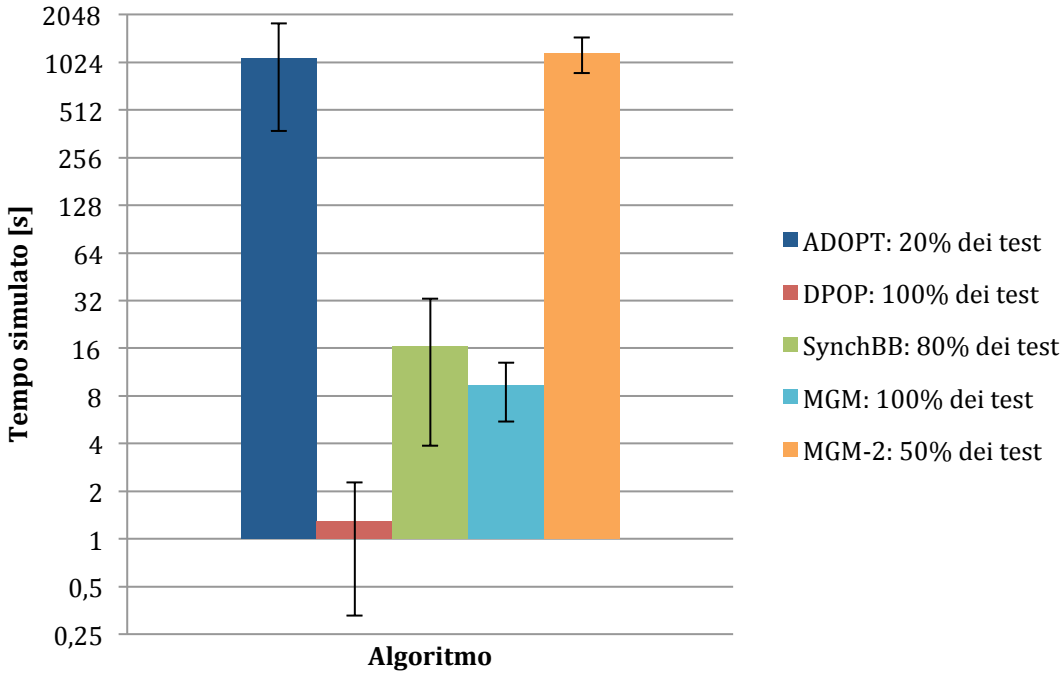

Figura 5.4: Confronto dei tempi medi di esecuzione con *n*=2 elettrodomestici.

Va fatta inoltre una precisazione: i dati mostrati riguardano i test che sono stati terminati con successo entro il timeout.

In Figura 5.4 sono messi a confronto i tempi medi di esecuzione e le deviazioni standard ottenute dai vari algoritmi.

Dal grafico emergono grandi differenze in termini di tempi di soluzione: come si può vedere, con ADOPT solo il 20% dei test è andato a buon fine, con tempi molto più alti (intorno ai 20 minuti) rispetto ai corrispettivi algoritmi ottimi. DPOP è l'algoritmo che ha fornito le migliori prestazioni, trovando per ogni problema una soluzione in pochi secondi, mentre SynchBB, pur non raggiungendo le performance di DPOP, è risultato essere comunque ben più affidabile rispetto ad ADOPT. Per quanto riguarda gli algoritmi non completi, le aspettative sono state confermate: è noto infatti che MGM mediamente trova una soluzione molto più velocemente rispetto a MGM-2.

In Figura 5.5 sono messi a confronto i costi medi (e le rispettive deviazioni standard) delle soluzioni trovate con il costo medio degli scenari prima dell'ottimizzazione. Essendo ADOPT, DPOP e SynchBB algoritmi completi,

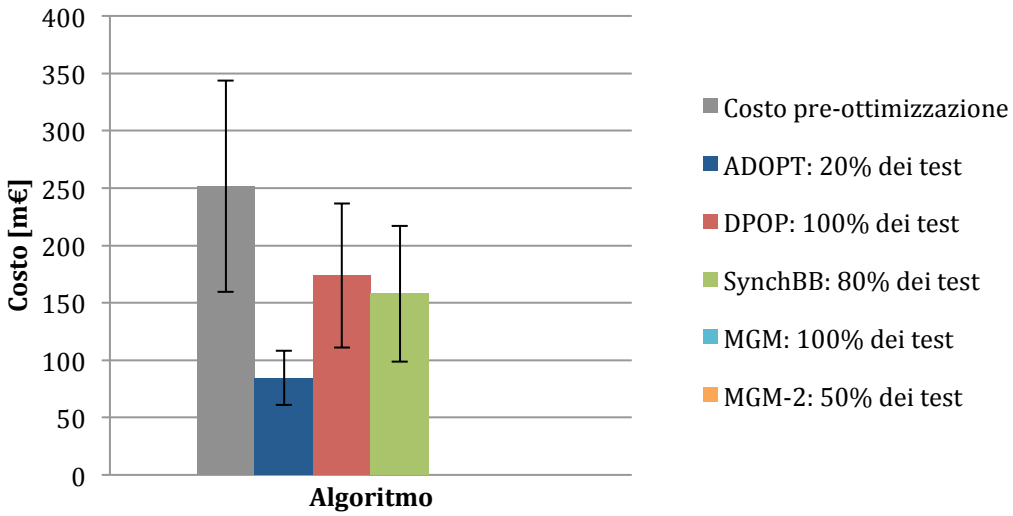

Figura 5.5: Confronto dei costi medi con *n*=2 elettrodomestici.

ci si aspetta che la qualità delle soluzioni trovate sia la stessa. Nel grafico, però, sono mostrati solamente i dati riferiti ai test terminati entro il timeout. Questo spiega le differenze di risultati ottenuti: al contrario di DPOP (e in parte di SynchBB), infatti, ADOPT ha trovato una soluzione soltanto negli scenari più semplici, caratterizzati da un minor numero di agenti, e quindi da un minore consumo. In Figura 5.6 è mostrato il costo medio percentuale ottenuto dagli algoritmi rispetto allo scenario non ottimizzato, calcolato sui test conclusi entro il timeout: si può effettivamente notare come in realtà il

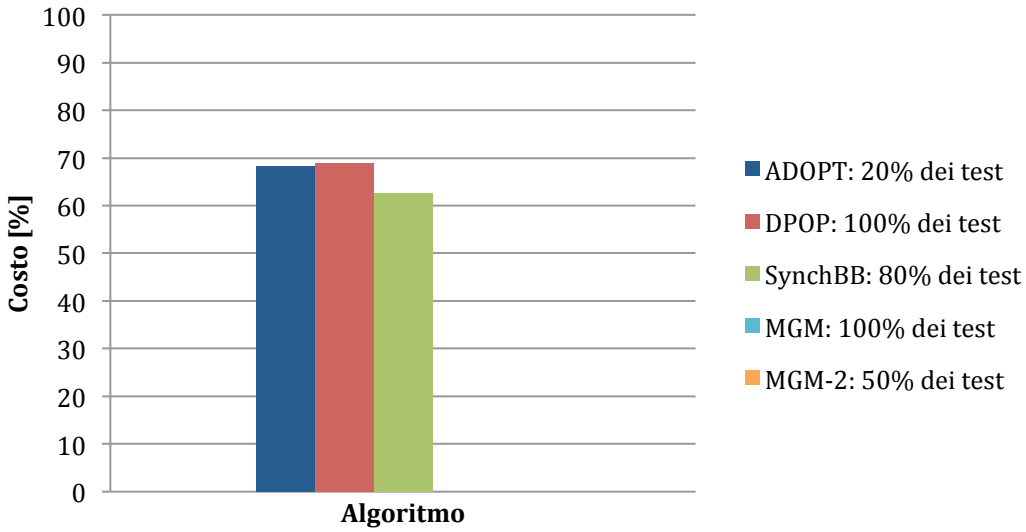

Figura 5.6: Costi medi percentuali rispetto ai test conclusi entro il timeout.

risparmio ottenuto sia pressoché uguale. Se consideriamo il dato riguardante DPOP (l'unico algoritmo che ha trovato una soluzione in tutti gli scenari), il risparmio medio dovuto all'ottimizzazione si attesta al 31%. Considerando invece soltano gli scenari risolti da tutti gli algoritmi ottimi, il risparmio medio ottenuto è intorno al 32%.

Hanno invece dato esito negativo i test riguardanti gli algoritmi non ottimi: in tutti gli scenari ottimizzati, infatti, sia MGM sia MGM-2 hanno riportato soluzioni di costo infinito, in quanto in violazione dei vincoli hard del modello. Ciò è dovuto al fatto che i due algoritmi non esplorano l'intero spazio delle soluzioni.

5.3 Ottimizzazione con *n*>2 elettrodomestici

Questa categoria di test è stata effettuata con lo scopo di verificare la validità della formulazione presentata al crescere del numero di elettrodomestici. Abbiamo perciò escluso dal modello i vincoli di non sovrapposizione dei cicli dei carichi, in modo tale da poter rappresentare una situazione in cui più

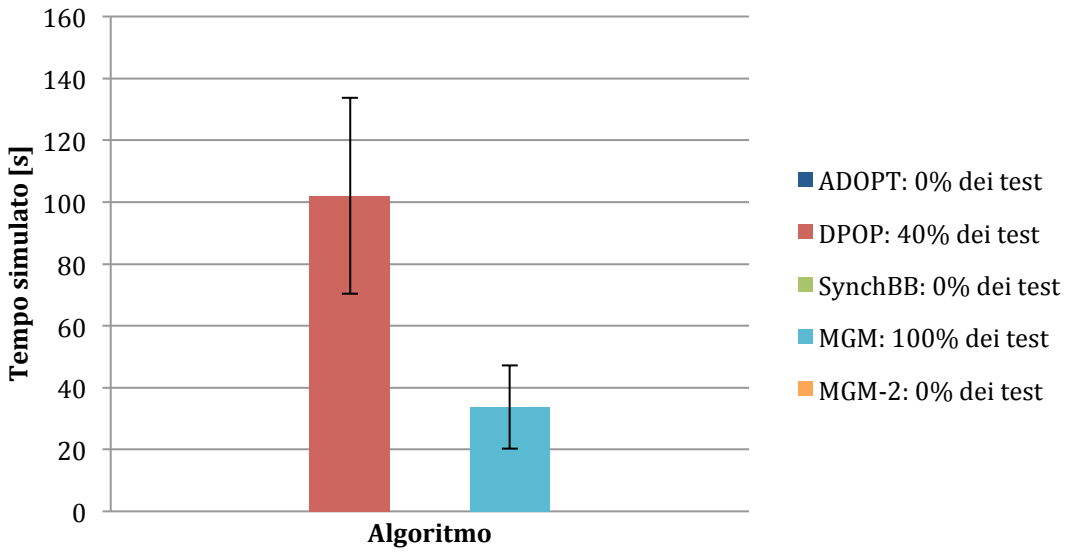

Figura 5.7: Confronto dei tempi medi di esecuzione con *n*=3 elettrodomestici.

elettrodomestici dello stesso tipo funzionano contemporaneamente (si pensi ad esempio al caso di un albergo, la cui lavanderia è formata da più lavatrici che possono operare simultaneamente).

I test sono stati eseguiti aumentando progressivamente il numero *n* di elettrodomestici ogni 10 scenari, con un passo di campionamento di 30 minuti per la definizione dei domini, e timeout impostato a 30 minuti.

In Figura 5.7 sono mostrati i tempi medi di esecuzione e le deviazioni standard ottenuti con n=3 elettrodomestici.

Come ci si aspettava data l'alta percentuale di test terminati in timeout con due elettrodomestici, sia ADOPT sia MGM-2 sono risultati inadatti a risolvere questa tipologia di test. Anche SynchBB, che invece nelle prove precedenti aveva ottenuto risultati più soddisfacenti, non ha trovato soluzione ad alcun problema entro il tempo di timeout. Un notevole degrado nelle prestazioni è stato riscontrato anche con DPOP: più della metà dei test effettuati, infatti, sono terminati in timeout, mentre per quanto riguarda i restanti, si è potuto notare un incremento dei tempi medi di risoluzione estremamente elevato (circa 1 secondo con n=2 elettrodomestici, 102 secondi invece con n=3). Questo peggioramento nelle performance è causato da un insieme di fattori: l'eliminazione dei vincoli di non sovrapposizione dei cicli dei carichi, il

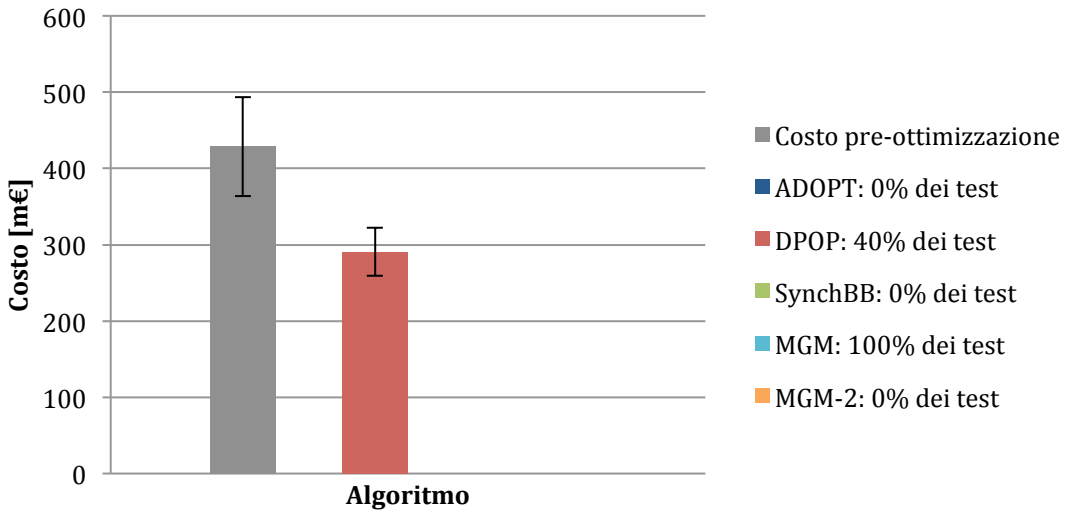

Figura 5.8: Confronto dei costi medi con n=3 elettrodomestici.

conseguente aumento del numero dei vincoli di potenza massima, e l'incremento della complessità di questi ultimi.

Con MGM, invece, i tempi di risoluzione si sono mantenuti bassi. Tuttavia, come già visto in precedenza, l'algoritmo non si adatta bene a questo modello, giacché tutte le soluzioni trovate sono di costo infinito.

In Figura 5.8, perciò, si può confrontare rispetto allo scenario non ottimizzato solamente il costo medio delle soluzioni trovate con DPOP: come si può vedere, il risparmio si attesta intorno al 32%, risultato in linea con il caso con due elettrodomestici.

Aumentando ancora il numero di elettrodomestici (n=4), otteniamo invece risultati negativi con tutti gli algoritmi (eccetto MGM, che trova però soltanto soluzioni di costo infinito), anche aumentando il timeout fino a 60 minuti.

5.4 Ottimizzazione al variare del passo di campionamento

Con questo test abbiamo voluto verificare il cambiamento delle prestazioni dei vari algoritmi al variare del passo di campionamento utilizzato. Per eseguire

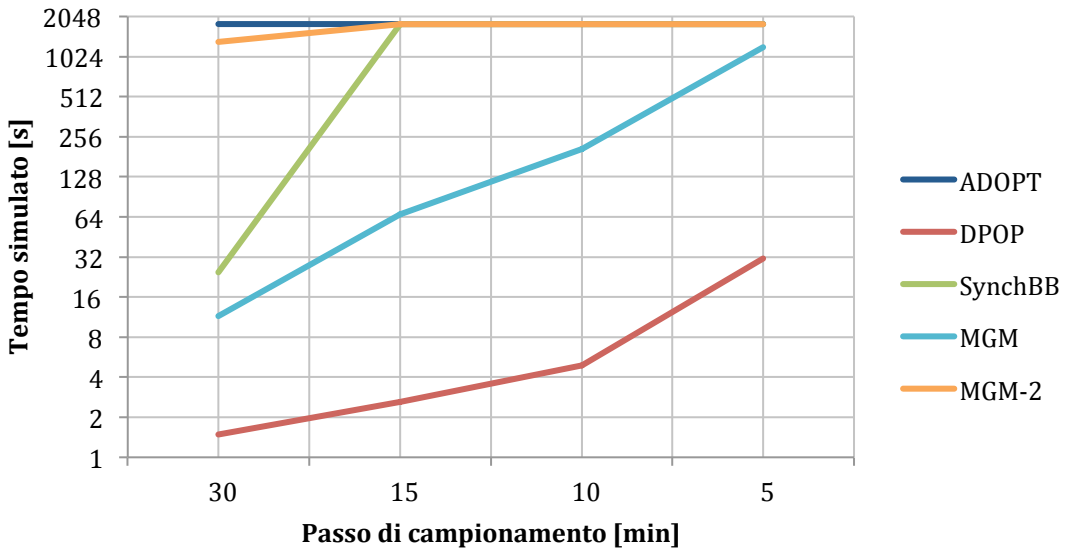

Figura 5.9: Confronto dei tempi di esecuzione al variare del passo di campionamento.

questo esperimento abbiamo scelto di utilizzare, tra i vari scenari sfruttati in precedenza, quello che avesse generato i risultati il più simili possibili ai valori medi trovati nei test in Sezione 5.2. Lo scenario è quindi caratterizzato da n=2 elettrodomestici, mentre il timeout è impostato a 30 minuti.

In Figura 5.9 sono mostrati i tempi di esecuzione dei vari algoritmi, al variare del passo di campionamento (Δt) utilizzato.

Aumentando il passo di campionamento fino a Δt=10, con DPOP e MGM l'aumento dei tempi di risoluzione è stato graduale; si è verificato invece un incremento particolarmente elevato con un passo di definizione di 5 minuti, dovuto all'esplosione delle dimensioni dell'albero di ricerca in conseguenza della notevole cardinalità dei domini delle variabili di inizio attività. Con SynchBB è stato possibile trovare una soluzione in tempo accettabile unicamente con un campionamento di 30 minuti, mentre né MGM-2 né ADOPT sono stati in grado di fornire una soluzione in tempi ragionevoli.

Ai fini dell'ottimizzazione dei costi, tuttavia, abbiamo riscontrato soltanto lievi differenze tra i test effettuati con diversi passi di definizione. In Figura 5.10 sono messi a confronto i costi ottenuti in seguito all'ottimizzazione

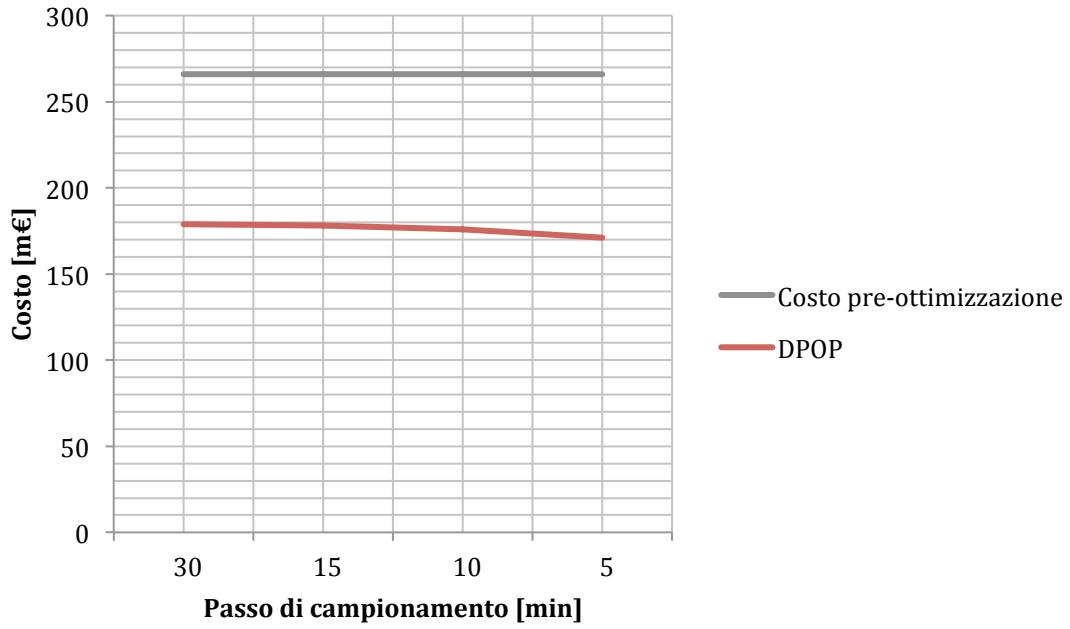

Figura 5.10: Confronto dei costi al variare del passo di campionamento.

effettuata con DPOP, rispetto allo scenario non ottimizzato (MGM, come già evidenziato in precedenza, restituisce soluzioni di costo infinito, mentre gli altri algoritmi non forniscono dati confrontabili). Con il passo di campionamento meno preciso (Δt=30 minuti) si ottiene un risparmio del 32.7%, con Δt=15 minuti il risparmio è del 33.1%, con Δt=10 minuti si ottiene un risparmio del 33.8% mentre con il campionamento più preciso (Δt=5 minuti), il risparmio ottenuto è del 35.7%. Come ci si aspettava dunque, è possibile notare una leggera diminuzione dei costi all'aumentare della precisione della definizione dei domini delle variabili. Tuttavia, tale miglioramento è necessariamente accompagnato da un degrado delle prestazioni in termini di tempi di esecuzione: si è passati da un tempo di risoluzione di 1.5 secondi con il campionamento di 30 minuti, ad un tempo di 31 secondi con il campionamento di 5 minuti. Sebbene in termini assoluti l'incremento ottenuto risulti certamente accettabile, bisogna considerare che in percentuale si tratta di un incremento consistente: potrebbe perciò essere un ostacolo difficilmente superabile in scenari di complessità maggiore.

5.5 Ottimizzazione con vincoli utente

Scopo di questa categoria di test è dimostrare la capacità del modello di permettere all'utente di impostare dei parametri per ottenere un'ottimizzazione maggiormente personalizzata secondo le sue necessità, offrendogli la possibilità di indicare un istante di tempo massimo di terminazione dei cicli, e un delay massimo (fissato intorno a un istante di avvio iniziale) per il loro avvio. Per ognuna delle due opzioni disponibili per l'utente, abbiamo effettuato un test specifico servendoci del medesimo scenario individuato per i test descritti in Sezione 5.4 (indicando opportunamente i vincoli utente desiderati), utilizzando un passo di campionamento di 30 minuti, e impostando con lo stesso valore il timeout.

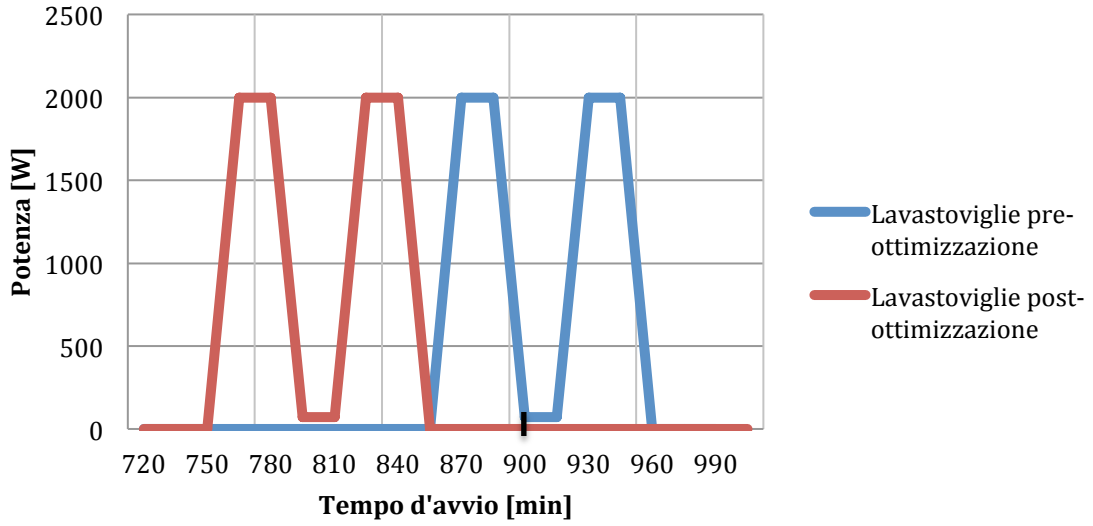

Figura 5.11: Spostamento dell'avvio di un elettrodomestico in seguito all'ottimizzazione con istante di tempo massimo fissato.

Per verificare l'ottimizzazione con tempo massimo di terminazione, abbiamo impostato le 13 come orario limite per la conclusione del ciclo della lavatrice, e le 15 come limite per la lavastoviglie. Come previsto, i tempi di risoluzione dei vari algoritmi si sono attestati sotto la media ottenuta nei test in Sezione 5.2: infatti, il vincolo utente inserito è in realtà modellato andando a eliminare dai domini delle variabili d'inizio attività i valori non ammissibili (ossia quelli che implicano la fine dei cicli degli elettrodomestici oltre la soglia indicata). Il risparmio ottenuto è invece rimasto all'incirca in linea con i valori ottenuti in precedenza, poiché gli algoritmi hanno potuto avviare almeno uno dei due elettrodomestici nelle fasce orarie di costo minore.

In Figura 5.11 è mostrato come esempio il ciclo di esecuzione della lavastoviglie, anticipato in modo da rispettare il tempo massimo di terminazione indicato.

Per verificare l'ottimizzazione con delay, abbiamo impostato lo spostamento massimo concesso per l'avvio degli elettrodomestici a 60 minuti. I tempi di risoluzione degli algoritmi considerati si sono attestati ancora ben sotto i valori medi ottenuti con l'ottimizzazione senza vincoli: anche in questo caso, infatti, il vincolo utente è espresso eliminando dai domini delle variabili di inizio attività i valori non ammissibili, cioè quelli al di fuori dell'intervallo indicato

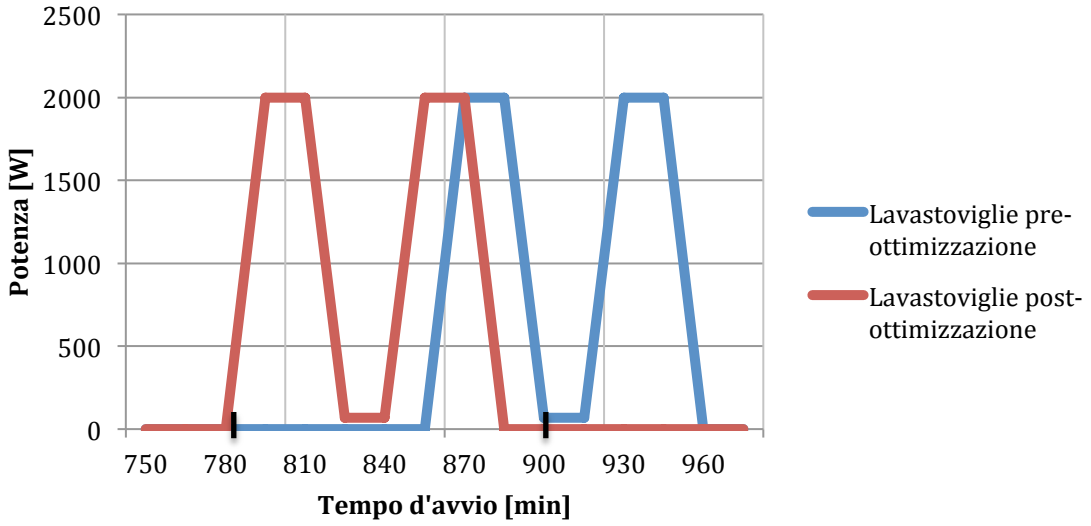

Figura 5.12: Spostamento dell'avvio di un elettrodomestico in seguito all'ottimizzazione con delay fissato.

dall'utente. Con questo tipo di vincolo, la complessità diminuisce notevolmente. Ne è anche dimostrazione il fatto che, ripetendo diverse volte il test, gli algoritmi incompleti utilizzati hanno ottenuto, per la prima volta con questo modello, soluzioni ammissibili: MGM ha trovato soluzioni ammissibili nel 50% dei casi, con un risparmio medio del 2% rispetto allo scenario non ottimizzato, mentre MGM-2 ha trovato soluzioni ammissibili nel 20% dei casi, con un risparmio medio del 7%. La soluzione ottima trovata dagli algoritmi completi ha evidenziato invece un risparmio del 12% rispetto allo scenario ottimizzato. Un risparmio così inferiore rispetto all'ottimizzazione senza vincoli era ampiamente prevedibile, poiché agli algoritmi è stato consentito un raggio d'azione molto limitato, restringendo notevolmente la gamma di valori assegnabili alle variabili di inizio attività.

In Figura 5.12 è mostrato come esempio il ciclo di esecuzione della lavastoviglie, anticipato entro l'intervallo massimo indicato.

Capitolo 6

Conclusioni e sviluppi futuri

Lo scopo di questo lavoro di tesi è stato analizzare in maniera sperimentale l'utilità del paradigma DCOP per affrontare il problema della gestione della potenza all'interno di una rete elettrica a generazione di potenza limitata. Abbiamo formalizzato il problema applicativo nel modo più generale possibile, facendo riferimento all'ambito domestico, e ponendo come obiettivo il risparmio sui costi in bolletta.

Abbiamo quindi sviluppato diverse formulazioni DCOP per il problema applicativo considerato, valutando per ciascuna di esse i vantaggi in termini di espressività del modello, e i problemi dovuti alla sua complessità (è noto infatti che i problemi di ottimizzazione vincolata, compresi quelli distribuiti, siano NP-difficili).

Tra le formulazioni presentate, quella ad attività semplificata si è dimostrata essere il miglior compromesso fra espressività e complessità. Abbiamo perciò sviluppato un software generatore di problemi, per creare in maniera sistematica istanze DCOP coerenti con tale modello e compatibili con il formato accettato da FRODO, il framework utilizzato per la risoluzione dei DCOP generati.

Per la valutazione sperimentale del modello si sono utilizzati diversi algoritmi, sia ottimi (ADOPT, DPOP, SynchBB) sia non ottimi (MGM, MGM-2). I risultati migliori sono stati ottenuti con DPOP, il quale ha permesso di ottenere negli scenari caratterizzati dalla presenza di due elettrodomestici un risparmio medio del 31% sulla spesa giornaliera, con tempi di esecuzione molto contenuti (intorno al secondo).

Abbiamo poi mostrato come la complessità del problema cresca esponenzialmente all'aumentare del numero degli elettrodomestici presenti nel sistema, causando incrementi molto elevati nei tempi di risoluzione.
Infine abbiamo evidenziato come aumentare la precisione della definizione dei domini delle variabili porti ad un aumento del risparmio esiguo (se le tariffe sono quelle del giorno prima), non proporzionato all'elevato aumento dei tempi di esecuzione degli algoritmi: è perciò importante scegliere un campionamento temporale che sia un giusto compromesso tra costi e benefici.

Tra gli sviluppi futuri di questo lavoro di tesi vi è l'implementazione di un algoritmo in grado di risolvere i DDCOP (DCOP dinamici), in modo da poter valutare in maniera approfondita la formulazione Dynamic DCOP presentata.
Un altro spunto riguarda l'implementazione della gestione dei vettori all'interno del framework FRODO e negli algoritmi DCOP. L'adozione di espedienti per aggirare tale mancanza, infatti, ha avuto un impatto negativo sull'efficienza della formulazione vettoriale presentata, aumentandone notevolmente la complessità e compromettendone l'utilizzo in scenari realistici.

Bibliografia

[1] S. Abras, S. Ploix, S. Pesty, and M. Jacomino. A multi-agent home automation system for power management. In *Third International Conference in Control, Automation, and Robotics, ICINCO 2006*, pages 59-68. Setubal, Portugal: Springer, 1-5 August 2006.

[2] A. L. Bazzan. Opportunities for Multiagent Systems and Multiagent Reinforcement Learning in Traffic Control. In *Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems,* pages 342–375. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2009.

[3] V. G. Castellana. Analyzing and exploiting the flexibility of DCOP formulations of energy management problems. Report, Politecnico di Milano, 2012.

[4] A. Gershman, R. Zivan, T. Grinshpoun, A. Grubshtein, and A. Meisels. Measuring distributed constraint optimization algorithms. In *Proceedings of the AAMAS'08 Distributed Constraint Reasoning Workshop (DCR'08)*, pages 17–24, Estoril, Portugal, May 13 2008.

[5] Gestore dei Mercati Energetici. http://www.mercatoelettrico.org/.

[6] K.Ghedira. A distributed approach to partial constraint satisfaction problems. In J. Perram and J.-P. Mller, editors, *Distributed Software Agents and Applications*, volume 1069 of *Lecture Notes in Computer Science*, pages 106–122. Springer Berlin / Heidelberg, 1996.

[7] Graphviz - Graph Visualization Software. http://www.graphviz.org/.

[8] K. Hirayama and M. Yokoo. Distributed partial constraint satisfaction problem. In Gert Smolka, editor, *Proceedings of the Third International Conference on Principles and Practice of Constraint*

Programming (CP'97), volume 1330, pages 222–236, Linz, Austria, Oct. 29–Nov. 1 1997.

[9] JaCoP - Java Constraint Programming solver. http://jacop.osolpro.com/.

[10] J. K. Kok, C. J. Warmer, and I.G. Kamphuis. PowerMatcher: Multiagent Control in the Electricity Infrastructure. In *Proceedings of the 4th International Conference on Autonomous Agents and Multiagent Systems,* pages 75–82. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2005.

[11] F. R. Kschischang, B. Frey, and H. A. Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions On Information Theory*, pages 498–519, February 2001.

[12] A. Kumar, B. Faltings, and A. Petcu. Distributed Constraint Optimization with Structured Resource Constraints. In *Proceedings of the Eighth International Joint Conference on Autonomous Agents and Multiagent Systems,* pages 923–930. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2009.

[13] J. Kwak, P. Varakantham, M. Tambe, L. Klein, F. Jazizadeh, G. Kavulya, B. B. Gerber, and D. J. Gerber. Towards Optimal Planning for Distributed Coordination Under Uncertainty in Energy Domains. In *2nd International Workshop on Agent Technologies for Energy Systems*, Taipei, Taiwan, May 2011.

[14] T. Léauté, B. Ottens, and R. Szymanek. FRODO: a FRamework for Open/Distributed Optimization Version 2.11. http://thomas.leaute.name/frodo/manual/FRODO_User_Manual.html.

[15] R. T. Maheswaran, J. P. Pearce, and M. Tambe. Distributed algorithms for DCOP: A graphical-game-based approach. In David A. Bader and Ashfaq A. Khokhar, editors, *Proceedings of the ISCA Seventeenth International Conference on Parallel and Distributed*

Computing Systems (ISCA PDCS'04), pages 432–439, Francisco, California, USA, September 15–17 2004. ISCA.

[16] P. J. Modi, W. Shen, M. Tambe, and M. Yokoo. ADOPT: Asynchronous distributed constraint optimization with quality guarantees. *Artificial Intelligence*, 161:149–180, 2005.

[17] F. Montecchi. Realizzazione e valutazione di un sistema per il demand side management in ambito domestico. Tesi di Laurea Magistrale, Politecnico di Milano, 2012.

[18] Organising Committee of the Third International Competition of CSP Solvers. *XML Representation of Constraint Networks – Format XCSP 2.1*, January 15 2008. http://www.cril.univ-artois.fr/~lecoutre/research/benchmarks/benchmarks.html.

[19] A. Petcu. FRODO: A FRamework for Open/Distributed constraint Optimization. Technical Report 2006/001, Swiss Federal Institute of Technology (EPFL), Lausanne (Switzerland), 2006.

[20] A. Petcu, and B. Faltings. A distributed, complete method for multiagent constraint optimization. In *CP 2004 - Fifth International Workshop on Distributed Constraint Reasoning (DCR2004) in Toronto, Canada*, September 2004.

[21] A. Petcu, and B. Faltings. A Scalable Method for Multiagent Constraint Optimization. In Leslie Pack Kaelbling and Alessandro Saffiotti, editors, *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI'05)*, pages 266–271, Edinburgh, Scotland, July 31 – August 5 2005. Professional Book Center, Denver, USA.

[22] A. Petcu, and B. Faltings. MB-DPOP: A new memory-bounded algorithm for distributed optimization. In Manuela M. Veloso, editor, *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI'07)*, pages 1452–1457, Hyderabad, India, January 6–12 2007.

[23] J. Pita, M. Jain, C. Western, C. Portway, M. Tambe, F. Ordóñez, S. Kraus, and P. Paruchuri. Deployed ARMOR Protection: The Application of a Game-Theoretic Model for Security at the Los Angeles International Airport. In *Proceedings of the 7th International Conference on Autonomous Agents and Multiagent Systems*, pages 125–132. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2008.

[24] S. D. Ramchurn, P. Vytelingum, A. Rogers, and N. R. Jennings. Agent-Based Control for Decentralised Demand Side Management in the Smart Grid. In *Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Systems*, pages 5–12. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2011.

[25] A. Rogers, S. Maleki, S. Ghosh, and R. J. Nicholas. Adaptive Home Heating Control Through Gaussian Process Prediction and Mathematical Programming. *2nd International Workshop on Agent Technologies for Energy Systems*. Taipei, Taiwan, 2 May 2011.

[26] F. Schweppe, B. Daryanian, and R. Tabors. Algorithms for a spot price responding residential load controller. *Power Engineering Review, IEEE*, 9(5):49 – 50, 1989.

[27] M. M. M. Soliman. Applicazione di tecniche DCOP per il controllo di reti elettriche con generazione di potenza limitata. Tesi di Laurea Magistrale, Politecnico di Milano, 2010.

[28] E. A. Sultanik, R. N. Lass, and W. C. Regli. DCOPolis: A framework for simulating and deploying distributed constraint optimization algorithms. In Jonathan P. Pearce, editor, *Proceedings of the Ninth International Workshop on Distributed Constraint Reasoning (CP-DCR'07)*, Providence, RI, USA, September 23 2007.

[29] M. Tartara. Cooperative Multiagent Systems for Electrical Energy Management. Technical Report, Politecnico di Milano, March 2011.

[30] J. Tsai, N. Fridman, E. Bowring, M. Brown, S. Epstein, G. Kaminka, S. Marsella, A. Ogden, I. Rika, A. Sheel, M. Taylor, X. Wang, A. Zilka, and M. Tambe. ESCAPES — Evacuation Simulation with Children, Authorities, Parents, Emotions, and Social comparison. In *Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Systems*. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2011.

[31] J. M. Vidal. Fundamentals of multiagent systems with NetLogo examples. 2010. http://jmvidal.cse.sc.edu/papers/mas.pdf.

[32] P. Vytelingum, S. D. Ramchurn, T. D. Voice, A. Rogers, and N. R. Jennings. Trading Agents for the Smart Electricity Grid. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems*, pages 897–904. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2010.

[33] P. Vytelingum, T. D. Voice, S. D. Ramchurn, A. Rogers, and N. R. Jennings. Agent-Based Micro-Storage Management for the Smart Grid. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems,* pages 39–46. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2010.

[34] M. Wahbi, R. Ezzahir, C. Bessiere, and E. H. Bouyakhf. DisChoco 2: A platform for distributed constraint reasoning. In *Proceedings of the Thirteenth International Workshop on Distributed Constraint Reasoning (DCR'11)*, pages 112–121, Barcelona, Spain, July 17 2011. http://www.lirmm.fr/coconut/dischoco/.

[35] W. Yeoh, P. Varakantham, X. Sun, and S. Koenig. Incremental DCOP Search Algorithms for Solving Dynamic DCOPs (Extended Abstract). In *Proceedings of the Tenth International Joint Conference on Autonomous Agents and Multiagent Systems,* pages 1069–1070.

Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2011.

[36] M. Yokoo, and K. Hirayama. Distributed breakout algorithm for solving distributed constraint satisfaction and optimization problems. In *Proc. ICMAS*, pages 401–408, 1996.

[37] Worldwatch Institute. State of the World 2013: Is Sustainability Still Possible?. The Worldwatch Institute, 2013. **[ALTRI_21]**

Appendice A

Esempi di istanze DCOP

A.1 Formulazione vettoriale

In Figura A.1 è mostrato un esempio di istanza DCOP modellata tramite la formulazione vettoriale descritta in Sezione 4.2.1.
Nel problema mostrato, è rappresentato un unico elettrodomestico controllato dall'agente *agentX*, caratterizzato da un singolo profilo di consumo suddiviso in 6 campioni di tempo. L'orizzonte temporale dell'ottimizzazione è suddivisa invece in 12 campioni. L'agente, perciò, ha il controllo di 12 variabili, ognuna rappresentante la potenza assorbita dal carico nell'istante di tempo associato. La scelta del profilo di funzionamento del carico è rappresentata dal predicato *vectorX*, il quale collega logicamente le variabili dell'agente, modellandone di fatto la forma vettoriale. Il vincolo di potenza massima è espresso tramite speciali vincoli globali (referenziati con *global:weightedSum*) associati a ciascun campione in cui è suddiviso l'orizzonte temporale. L'obiettivo dell'ottimizzazione, in questo particolare esempio, consiste nel mantenere accesi gli elettrodomestici presenti, utilizzando il più possibile la potenza disponibile. Ciò è rappresentato dalla funzione *costFunction*, al quale fanno riferimento tanti vincoli quanti sono i campioni temporali (esattamente come per i vincolo di potenza massima).

```
<instance>
  <presentation name="modelloVettoriale" maxConstraintArity="12"
                maximize="false" format="XCSP 2.1_FRODO" />

  <agents nbAgents="1">
    <agent name="agentX" />
  </agents>

  <!-- Profilo di consumo di X=[t0:16 t1:15 t2:14 t3:16 t4:9 t5:10]. -->
  <domains nbDomains="12">
    <domain name="DomainX0" nbValues="2">0 16</domain>
    <domain name="DomainX1" nbValues="3">0 15 16</domain>
    <domain name="DomainX2" nbValues="4">0 14 15 16</domain>
    <domain name="DomainX3" nbValues="4">0 14 15 16</domain>
    <domain name="DomainX4" nbValues="5">0 9 14 15 16</domain>
    <domain name="DomainX5" nbValues="6">0 9 10 14 15 16</domain>
    <domain name="DomainX6" nbValues="6">0 9 10 14 15 16</domain>
    <domain name="DomainX7" nbValues="6">0 9 10 14 15 16</domain>
    <domain name="DomainX8" nbValues="5">0 9 10 14 16</domain>
    <domain name="DomainX9" nbValues="4">0 9 10 16</domain>
    <domain name="DomainX10" nbValues="3">0 9 10</domain>
    <domain name="DomainX11" nbValues="2">0 10</domain>
  </domains>

  <!-- Orizzonte temporale di 12 campioni (quindi 12 variabili per agente). -->
  <variables nbVariables="12">
    <variable name="X0" domain="DomainX0" agent="agentX" />
    <variable name="X1" domain="DomainX1" agent="agentX" />
    <variable name="X2" domain="DomainX2" agent="agentX" />
    <variable name="X3" domain="DomainX3" agent="agentX" />
    <variable name="X4" domain="DomainX4" agent="agentX" />
    <variable name="X5" domain="DomainX5" agent="agentX" />
    <variable name="X6" domain="DomainX6" agent="agentX" />
    <variable name="X7" domain="DomainX7" agent="agentX" />
    <variable name="X8" domain="DomainX8" agent="agentX" />
    <variable name="X9" domain="DomainX9" agent="agentX" />
    <variable name="X10" domain="DomainX10" agent="agentX" />
    <variable name="X11" domain="DomainX11" agent="agentX" />
  </variables>

  <functions nbFunctions="1">
    <!-- Funzione di costo per minimizzare energia disponibile inutilizzata. -->
    <function name="costFunction" return="int">
      <parameters> int X </parameters>
      <expression>
        <functional> sub(20,X) </functional>
      </expression>
    </function>
  </functions>

  <predicates nbPredicates="1">
    <!-- Vincolo che esprime il profilo di carico come fosse un vettore. -->
    <predicate name="vectorX">
      <parameters> V0 V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11 </parameters>
      <expression>
        <functional>
or(or(or(or(or(or(and(and(and(and(and(and(and(and(and(eq(V0,16),eq(V1,15))
,eq(V2,14)),eq(V3,16)),eq(V4,9)),eq(V5,10)),eq(V6,0)),eq(V7,0)),eq(V8,0)),eq(V9,0)
),eq(V10,0)),eq(V11,0)),
and(and(and(and(and(and(and(and(and(and(eq(V1,16),eq(V2,15)),eq(V3,14)),eq(V4,
16)),eq(V5,9)),eq(V6,10)),eq(V7,0)),eq(V8,0)),eq(V9,0)),eq(V10,0)),eq(V11,0)),eq(V
0,0))),
and(and(and(and(and(and(and(and(and(and(eq(V2,16),eq(V3,15)),eq(V4,14)),eq(V5,
16)),eq(V6,9)),eq(V7,10)),eq(V8,0)),eq(V9,0)),eq(V10,0)),eq(V11,0)),eq(V0,0)),eq(V
1,0))),
and(and(and(and(and(and(and(and(and(and(eq(V3,16),eq(V4,15)),eq(V5,14)),eq(V6,
16)),eq(V7,9)),eq(V8,10)),eq(V9,0)),eq(V10,0)),eq(V11,0)),eq(V0,0)),eq(V1,0)),eq(V
2,0))),
```

```
and(and(and(and(and(and(and(and(and(and(eq(V4,16),eq(V5,15)),eq(V6,14)),eq(V7,
16)),eq(V8,9)),eq(V9,10)),eq(V10,0)),eq(V11,0)),eq(V0,0)),eq(V1,0)),eq(V2,0)),eq(V
3,0))),
and(and(and(and(and(and(and(and(and(and(eq(V5,16),eq(V6,15)),eq(V7,14)),eq(V8,
16)),eq(V9,9)),eq(V10,10)),eq(V11,0)),eq(V0,0)),eq(V1,0)),eq(V2,0)),eq(V3,0)),eq(V
4,0))),
and(and(and(and(and(and(and(and(and(and(eq(V6,16),eq(V7,15)),eq(V8,14)),eq(V9,
16)),eq(V10,9)),eq(V11,10)),eq(V0,0)),eq(V1,0)),eq(V2,0)),eq(V3,0)),eq(V4,0)),eq(V
5,0)))
        </functional>
      </expression>
    </predicate>
  </predicates>

  <constraints nbConstraints="25" initialCost="0" maximalCost="20">
    <!-- Vincoli hard su potenza disponibile. -->
    <constraint name="PA0" arity="1" scope="X0" reference="global:weightedSum">
      <parameters> [ { 1 X0 } ] <le/> 20 </parameters>
    </constraint>
    <constraint name="PA1" arity="1" scope="X1" reference="global:weightedSum">
      <parameters> [ { 1 X1 } ] <le/> 20 </parameters>
    </constraint>
    <constraint name="PA2" arity="1" scope="X2" reference="global:weightedSum">
      <parameters> [ { 1 X2 } ] <le/> 20 </parameters>
    </constraint>
    <constraint name="PA3" arity="1" scope="X3" reference="global:weightedSum">
      <parameters> [ { 1 X3 } ] <le/> 20 </parameters>
    </constraint>
    <constraint name="PA4" arity="1" scope="X4" reference="global:weightedSum">
      <parameters> [ { 1 X4 } ] <le/> 20 </parameters>
    </constraint>
    <constraint name="PA5" arity="1" scope="X5" reference="global:weightedSum">
      <parameters> [ { 1 X5 } ] <le/> 20 </parameters>
    </constraint>
    <constraint name="PA6" arity="1" scope="X6" reference="global:weightedSum">
      <parameters> [ { 1 X6 } ] <le/> 20 </parameters>
    </constraint>
    <constraint name="PA7" arity="1" scope="X7" reference="global:weightedSum">
      <parameters> [ { 1 X7 } ] <le/> 20 </parameters>
    </constraint>
    <constraint name="PA8" arity="1" scope="X8" reference="global:weightedSum">
      <parameters> [ { 1 X8 } ] <le/> 20 </parameters>
    </constraint>
    <constraint name="PA9" arity="1" scope="X9" reference="global:weightedSum">
      <parameters> [ { 1 X9 } ] <le/> 20 </parameters>
    </constraint>
    <constraint name="PA10" arity="1" scope="X10" reference="global:weightedSum">
      <parameters> [ { 1 X10 } ] <le/> 20 </parameters>
    </constraint>
    <constraint name="PA11" arity="1" scope="X11" reference="global:weightedSum">
      <parameters> [ { 1 X11 } ] <le/> 20 </parameters>
    </constraint>
    <!-- Vincolo hard per modellare vincolo vettoriale. -->
    <contraint name="vectorXconstraint" arity="12" scope="X0 X1 X2 X3 X4 X5 X6 X7
X8 X9 X10 X11" reference="vectorX">
      <parameters> X0 X1 X2 X3 X4 X5 X6 X7 X8 X9 X10 X11 </parameters>
    </contraint>
    <!-- Vincoli soft per minizzare potenza residua libera. -->
    <constraint name="C0" arity="1" scope="X0" reference="costFunction">
      <parameters> X0 </parameters>
    </constraint>
    <constraint name="C1" arity="1" scope="X1" reference="costFunction">
      <parameters> X1 </parameters>
    </constraint>
    <constraint name="C2" arity="1" scope="X2" reference="costFunction">
      <parameters> X2 </parameters>
    </constraint>
    <constraint name="C3" arity="1" scope="X3" reference="costFunction">
```

```
115       <parameters> X3 </parameters>
116     </constraint>
117     <constraint name="C4" arity="1" scope="X4" reference="costFunction">
118       <parameters> X4 </parameters>
119     </constraint>
120     <constraint name="C5" arity="1" scope="X5" reference="costFunction">
121       <parameters> X5 </parameters>
122     </constraint>
123     <constraint name="C6" arity="1" scope="X6" reference="costFunction">
124       <parameters> X6 </parameters>
125     </constraint>
126     <constraint name="C7" arity="1" scope="X7" reference="costFunction">
127       <parameters> X7 </parameters>
128     </constraint>
129     <constraint name="C8" arity="1" scope="X8" reference="costFunction">
130       <parameters> X8 </parameters>
131     </constraint>
132     <constraint name="C9" arity="1" scope="X9" reference="costFunction">
133       <parameters> X9 </parameters>
134     </constraint>
135     <constraint name="C10" arity="1" scope="X10" reference="costFunction">
136       <parameters> X10 </parameters>
137     </constraint>
138     <constraint name="C11" arity="1" scope="X11" reference="costFunction">
139       <parameters> X11 </parameters>
140     </constraint>
141   </constraints>
142 </instance>
143
```

Figura A.1: Esempio di istanza DCOP formulata secondo il modello vettoriale.

A.2 Formulazione ad attività

In Figura A.2 è mostrato un esempio di istanza DCOP modellata tramite la formulazione ad attività descritta in Sezione 4.2.2.

Nel problema mostrato, sono rappresentati due elettrodomestici controllati rispettivamente dagli agenti *agent0* e *agent1*, ciascuno dei quali caratterizzati da profili di consumo suddivisibili in 3 attività. Ogni agente controlla perciò 9 variabili: 3 variabili che rappresentano l'istante di inizio di un'attività (*agent#AgAct#Att*, con *#Ag* che indica il numero dell'agente e *#Att* il numero dell'attività), 3 variabili che rappresentano la durata dell'attività (*agent#AgDur#Att*), e 3 variabili che ne rappresentano la potenza assorbita (*agent#AgPow#Att*). In questo esempio, solo un agente ha la possibilità di scegliere il profilo di consumo da utilizzare: tale possibilità è rappresentata tramite il predicato *predicate_profile_agent0*. Ogni agente, inoltre, ha associato un vincolo di continuità tra le attività, che rappresenta il collegamento logico (e temporale) tra le variabili d'inizio attività. Tali vincoli fanno riferimento ai predicati *predicate_continuity_agent#Ag*. I vincoli di potenza massima sono espressi tramite relazioni che assegnano costo infinito alle tuple (combinazione di variabili attività, durata e potenza) non ammissibili. La relazione *relation_agent0Pow0agent1Pow0*, infatti, attribuisce un costo infinito ai particolari assegnamenti delle variabili che porterebbero a un consumo di potenza superiore alla soglia massima (impostata a 3000). Come si può vedere, il vincolo di potenza massima coinvolge 6 variabili (3 variabili per ogni agente presente nello scenario).

L'obiettivo dell'ottimizzazione consiste nel minimizzare i costi in bolletta ottenuti associando al problema un piano tariffario: a ogni attività è associata una relazione (*relation_agent#AgAct#Att*), che assegna un costo a ogni coppia di valori indicanti l'istante di inizio e la durata di tale attività.

```
<instance>
    <presentation name="modelloAttività" maxConstraintArity="6" maximize="false"
format="XCSP 2.1_FRODO" />

    <agents nbAgents="2">
        <agent name="agent0" />
        <agent name="agent1" />
    </agents>

    <domains nbDomains="18">
        <domain name="domain_agent0Act0" nbValues="44">0 30 60 90 120 150 180 210
 240 270 300 330 360 390 420 450 480 510 540 570 600 630 660 690 720 750 780 810
 840 870 900 930 960 990 1020 1050 1080 1110 1140 1170 1200 1230 1260 1290 </
domain>
        <domain name="domain_agent0Act1" nbValues="44">30 60 90 120 150 180 210
 240 270 300 330 360 390 420 450 480 510 540 570 600 630 660 690 720 750 780 810
 840 870 900 930 960 990 1020 1050 1080 1110 1140 1170 1200 1230 1260 1290 1320 </
domain>
        <domain name="domain_agent0Act2" nbValues="44">120 150 180 210 240 270
 300 330 360 390 420 450 480 510 540 570 600 630 660 690 720 750 780 810 840 870
 900 930 960 990 1020 1050 1080 1110 1140 1170 1200 1230 1260 1290 1320 1350 1380
 1410 </domain>
        <domain name="domain_agent0Dur0" nbValues="2">30 40 </domain>
        <domain name="domain_agent0Dur1" nbValues="1">90 </domain>
        <domain name="domain_agent0Dur2" nbValues="1">30 </domain>
        <domain name="domain_agent0Pow0" nbValues="2">2120 1950 </domain>
        <domain name="domain_agent0Pow1" nbValues="1">140 </domain>
        <domain name="domain_agent0Pow2" nbValues="1">447 </domain>
        <domain name="domain_agent1Act0" nbValues="44">0 30 60 90 120 150 180 210
 240 270 300 330 360 390 420 450 480 510 540 570 600 630 660 690 720 750 780 810
 840 870 900 930 960 990 1020 1050 1080 1110 1140 1170 1200 1230 1260 1290 </
domain>
        <domain name="domain_agent1Act1" nbValues="44">60 90 120 150 180 210 240
 270 300 330 360 390 420 450 480 510 540 570 600 630 660 690 720 750 780 810 840
 870 900 930 960 990 1020 1050 1080 1110 1140 1170 1200 1230 1260 1290 1320 1350 </
domain>
        <domain name="domain_agent1Act2" nbValues="44">120 150 180 210 240 270
 300 330 360 390 420 450 480 510 540 570 600 630 660 690 720 750 780 810 840 870
 900 930 960 990 1020 1050 1080 1110 1140 1170 1200 1230 1260 1290 1320 1350 1380
 1410 </domain>
        <domain name="domain_agent1Dur0" nbValues="1">60 </domain>
        <domain name="domain_agent1Dur1" nbValues="1">60 </domain>
        <domain name="domain_agent1Dur2" nbValues="1">30 </domain>
        <domain name="domain_agent1Pow0" nbValues="1">2047 </domain>
        <domain name="domain_agent1Pow1" nbValues="1">148 </domain>
        <domain name="domain_agent1Pow2" nbValues="1">678 </domain>
    </domains>

    <variables nbVariables="18">
        <variable name="agent0Act0" domain="domain_agent0Act0" agent="agent0"/>
        <variable name="agent0Act1" domain="domain_agent0Act1" agent="agent0"/>
        <variable name="agent0Act2" domain="domain_agent0Act2" agent="agent0"/>
        <variable name="agent0Dur0" domain="domain_agent0Dur0" agent="agent0"/>
        <variable name="agent0Dur1" domain="domain_agent0Dur1" agent="agent0"/>
        <variable name="agent0Dur2" domain="domain_agent0Dur2" agent="agent0"/>
        <variable name="agent0Pow0" domain="domain_agent0Pow0" agent="agent0"/>
        <variable name="agent0Pow1" domain="domain_agent0Pow1" agent="agent0"/>
        <variable name="agent0Pow2" domain="domain_agent0Pow2" agent="agent0"/>
        <variable name="agent1Act0" domain="domain_agent1Act0" agent="agent1"/>
        <variable name="agent1Act1" domain="domain_agent1Act1" agent="agent1"/>
        <variable name="agent1Act2" domain="domain_agent1Act2" agent="agent1"/>
        <variable name="agent1Dur0" domain="domain_agent1Dur0" agent="agent1"/>
        <variable name="agent1Dur1" domain="domain_agent1Dur1" agent="agent1"/>
        <variable name="agent1Dur2" domain="domain_agent1Dur2" agent="agent1"/>
        <variable name="agent1Pow0" domain="domain_agent1Pow0" agent="agent1"/>
        <variable name="agent1Pow1" domain="domain_agent1Pow1" agent="agent1"/>
        <variable name="agent1Pow2" domain="domain_agent1Pow2" agent="agent1"/>
    </variables>
```

```
50
51      <relations nbRelations="7">
52          <!-- Funzioni (relazioni) di costo monetario. -->
53          <relation name="relation_agent0Act0" arity="2" nbTuples="44"
semantics="soft" defaultCost="infinity">
54              34: 0 30|34: 30 30|32: 60 30|32: 90 30|31: 120 30|31: 150 30|30: 180
30|30: 210 30|30: 240 30|30: 270 30|30: 300 30|30: 330 30|30: 360 30|30: 390 30|
31: 420 30|31: 450 30|30: 480 30|30: 510 30|28: 540 30|28: 570 30|27: 600 30|27:
630 30|23: 660 30|23: 690 30|23: 720 30|23: 750 30|19: 780 30|19: 810 30|27: 840
30|27: 870 30|30: 900 30|30: 930 30|32: 960 30|32: 990 30|37: 1020 30|37: 1050 30|
41: 1080 30|41: 1110 30|41: 1140 30|41: 1170 30|37: 1200 30|37: 1230 30|35: 1260
30|35: 1290 30
55          </relation>
56          <relation name="relation_agent0Act1" arity="2" nbTuples="44"
semantics="soft" defaultCost="infinity">
57              99: 30 90|96: 60 90|95: 90 90|93: 120 90|92: 150 90|91: 180 90|91:
210 90|91: 240 90|91: 270 90|91: 300 90|92: 330 90|93: 360 90|94: 390 90|94: 420
90|93: 450 90|90: 480 90|87: 510 90|84: 540 90|82: 570 90|77: 600 90|74: 630 90|
70: 660 90|70: 690 90|66: 720 90|62: 750 90|66: 780 90|73: 810 90|85: 840 90|89:
870 90|94: 900 90|95: 930 90|102: 960 90|107: 990 90|116: 1020 90|121: 1050 90|
125: 1080 90|125: 1110 90|121: 1140 90|116: 1170 90|110: 1200 90|108: 1230 90|103:
1260 90|100: 1290 90|96: 1320 90
58          </relation>
59          <relation name="relation_agent0Act2" arity="2" nbTuples="44"
semantics="soft" defaultCost="infinity">
60              31: 120 30|31: 150 30|30: 180 30|30: 210 30|30: 240 30|30: 270 30|
30: 300 30|30: 330 30|30: 360 30|30: 390 30|31: 420 30|31: 450 30|30: 480 30|30:
510 30|28: 540 30|28: 570 30|27: 600 30|27: 630 30|23: 660 30|23: 690 30|23: 720
30|23: 750 30|19: 780 30|19: 810 30|27: 840 30|27: 870 30|30: 900 30|30: 930 30|
32: 960 30|32: 990 30|37: 1020 30|37: 1050 30|41: 1080 30|41: 1110 30|41: 1140 30|
41: 1170 30|37: 1200 30|37: 1230 30|35: 1260 30|35: 1290 30|32: 1320 30|32: 1350
30|31: 1380 30|31: 1410 30
61          </relation>
62          <relation name="relation_agent1Act0" arity="2" nbTuples="44"
semantics="soft" defaultCost="infinity">
63              69: 0 60|67: 30 60|64: 60 60|63: 90 60|63: 120 60|62: 150 60|60: 180
60|60: 210 60|60: 240 60|60: 270 60|60: 300 60|61: 330 60|61: 360 60|62: 390 60|
63: 420 60|62: 450 60|61: 480 60|59: 510 60|56: 540 60|55: 570 60|54: 600 60|50:
630 60|46: 660 60|46: 690 60|46: 720 60|43: 750 60|39: 780 60|46: 810 60|54: 840
60|58: 870 60|61: 900 60|63: 930 60|64: 960 60|69: 990 60|75: 1020 60|79: 1050 60|
83: 1080 60|83: 1110 60|83: 1140 60|79: 1170 60|75: 1200 60|73: 1230 60|71: 1260
60|67: 1290 60
64          </relation>
65          <relation name="relation_agent1Act1" arity="2" nbTuples="44"
semantics="soft" defaultCost="infinity">
66              64: 60 60|63: 90 60|63: 120 60|62: 150 60|60: 180 60|60: 210 60|60:
240 60|60: 270 60|60: 300 60|61: 330 60|61: 360 60|62: 390 60|63: 420 60|62: 450
60|61: 480 60|59: 510 60|56: 540 60|55: 570 60|54: 600 60|50: 630 60|46: 660 60|
46: 690 60|46: 720 60|43: 750 60|39: 780 60|46: 810 60|54: 840 60|58: 870 60|61:
900 60|63: 930 60|64: 960 60|69: 990 60|75: 1020 60|79: 1050 60|83: 1080 60|83:
1110 60|83: 1140 60|79: 1170 60|75: 1200 60|73: 1230 60|71: 1260 60|67: 1290 60|
64: 1320 60|63: 1350 60
67          </relation>
68          <relation name="relation_agent1Act2" arity="2" nbTuples="44"
semantics="soft" defaultCost="infinity">
69              31: 120 30|31: 150 30|30: 180 30|30: 210 30|30: 240 30|30: 270 30|
30: 300 30|30: 330 30|30: 360 30|30: 390 30|31: 420 30|31: 450 30|30: 480 30|30:
510 30|28: 540 30|28: 570 30|27: 600 30|27: 630 30|23: 660 30|23: 690 30|23: 720
30|23: 750 30|19: 780 30|19: 810 30|27: 840 30|27: 870 30|30: 900 30|30: 930 30|
32: 960 30|32: 990 30|37: 1020 30|37: 1050 30|41: 1080 30|41: 1110 30|41: 1140 30|
41: 1170 30|37: 1200 30|37: 1230 30|35: 1260 30|35: 1290 30|32: 1320 30|32: 1350
30|31: 1380 30|31: 1410 30
70          </relation>
71          <!-- Relazioni relative ai vincoli di potenza massima disponibile. -->
72          <relation name="relation_agent0Pow0agent1Pow0" arity="6" nbTuples="87"
semantics="soft" defaultCost="0">
73              infinity: 0 30 2120 0 60 2047|30 30 2120 0 60 2047|30 30 2120 30 60
2047|60 30 2120 30 60 2047|60 30 2120 60 60 2047|90 30 2120 60 60 2047|90 30 2120
```

```
90 60 2047|120 30 2120 90 60 2047|120 30 2120 120 60 2047|150 30 2120 120 60 2047|
150 30 2120 150 60 2047|180 30 2120 150 60 2047|180 30 2120 180 60 2047|210 30
2120 180 60 2047|210 30 2120 210 60 2047|240 30 2120 210 60 2047|240 30 2120 240
60 2047|270 30 2120 240 60 2047|270 30 2120 270 60 2047|300 30 2120 270 60 2047|
300 30 2120 300 60 2047|330 30 2120 300 60 2047|330 30 2120 330 60 2047|360 30
2120 330 60 2047|360 30 2120 360 60 2047|390 30 2120 360 60 2047|390 30 2120 390
60 2047|420 30 2120 390 60 2047|420 30 2120 420 60 2047|450 30 2120 420 60 2047|
450 30 2120 450 60 2047|480 30 2120 450 60 2047|480 30 2120 480 60 2047|510 30
2120 480 60 2047|510 30 2120 510 60 2047|540 30 2120 510 60 2047|540 30 2120 540
60 2047|570 30 2120 540 60 2047|570 30 2120 570 60 2047|600 30 2120 570 60 2047|
600 30 2120 600 60 2047|630 30 2120 600 60 2047|630 30 2120 630 60 2047|660 30
2120 630 60 2047|660 30 2120 660 60 2047|690 30 2120 660 60 2047|690 30 2120 690
60 2047|720 30 2120 690 60 2047|720 30 2120 720 60 2047|750 30 2120 720 60 2047|
750 30 2120 750 60 2047|780 30 2120 750 60 2047|780 30 2120 780 60 2047|810 30
2120 780 60 2047|810 30 2120 810 60 2047|840 30 2120 810 60 2047|840 30 2120 840
60 2047|870 30 2120 840 60 2047|870 30 2120 870 60 2047|900 30 2120 870 60 2047|
900 30 2120 900 60 2047|930 30 2120 900 60 2047|930 30 2120 930 60 2047|960 30
2120 930 60 2047|960 30 2120 960 60 2047|990 30 2120 960 60 2047|990 30 2120 990
60 2047|1020 30 2120 990 60 2047|1020 30 2120 1020 60 2047|1050 30 2120 1020 60
2047|1050 30 2120 1050 60 2047|1080 30 2120 1050 60 2047|1080 30 2120 1080 60
2047|1110 30 2120 1080 60 2047|1110 30 2120 1110 60 2047|1140 30 2120 1110 60
2047|1140 30 2120 1140 60 2047|1170 30 2120 1140 60 2047|1170 30 2120 1170 60
2047|1200 30 2120 1170 60 2047|1200 30 2120 1200 60 2047|1230 30 2120 1200 60
2047|1230 30 2120 1230 60 2047|1260 30 2120 1230 60 2047|1260 30 2120 1260 60
2047|1290 30 2120 1260 60 2047|1290 30 2120 1290 60 2047
          </relation>
     </relations>

     <predicates nbPredicates="3">
          <!-- Predicati che esprimono necessarietà di continuità tra le attività
di un agente. -->
          <predicate name="predicate_continuity_agent0">
               <parameters> agent0Act0 agent0Act1 agent0Act2 agent0Dur0 agent0Dur1
agent0Dur2 </parameters>
               <expression>
                    <functional>
and(eq(agent0Act1,add(agent0Act0,agent0Dur0)),eq(agent0Act2,add(agent0Act1,agent0D
ur1))) </functional>
               </expression>
          </predicate>
          <predicate name="predicate_continuity_agent1">
               <parameters> agent1Act0 agent1Act1 agent1Act2 agent1Dur0 agent1Dur1
agent1Dur2 </parameters>
               <expression>
                    <functional>
and(eq(agent1Act1,add(agent1Act0,agent1Dur0)),eq(agent1Act2,add(agent1Act1,agent1D
ur1))) </functional>
               </expression>
          </predicate>
          <!-- Predicato che esprime il vincolo di scelta profilo di consumo. -->
          <predicate name="predicate_profile_agent0">
               <parameters> agent0Dur0 agent0Pow0 </parameters>
               <expression>
                    <functional>
or(and(eq(agent0Dur0,30),eq(agent0Pow0,2120)),and(eq(agent0Dur0,40),eq(agent0Pow0,
1950))) </functional>
               </expression>
          </predicate>
     </predicates>

     <constraints nbConstraints="9">
          <!-- Vincoli hard per modellare continuità dei profili di consumo. -->
          <constraint name="Constr_continuity_agent0" arity="6" scope="agent0Act0
agent0Act1 agent0Act2 agent0Dur0 agent0Dur1 agent0Dur2 "
reference="predicate_continuity_agent0">
               <parameters> agent0Act0 agent0Act1 agent0Act2 agent0Dur0 agent0Dur1
agent0Dur2 </parameters>
          </constraint>
```

```
105          <constraint name="Constr_continuity_agent1" arity="6" scope="agent1Act0
     agent1Act1 agent1Act2 agent1Dur0 agent1Dur1 agent1Dur2 "
     reference="predicate_continuity_agent1">
106              <parameters> agent1Act0 agent1Act1 agent1Act2 agent1Dur0 agent1Dur1
     agent1Dur2 </parameters>
107          </constraint>
108          <!-- Vincoli hard per modellare potenza massima disponibile. -->
109          <constraint name="Constr_agent0Pow0agent1Pow0" arity="6"
     scope="agent0Act0 agent0Dur0 agent0Pow0 agent1Act0 agent1Dur0 agent1Pow0"
     reference="relation_agent0Pow0agent1Pow0" />
110          <!-- Vincoli hard per modellare vincoli di scelta profilo di consumo. -->
111          <constraint name="Constr_profile_agent0" arity="2" scope="agent0Dur0
     agent0Pow0" reference="predicate_profile_agent0">
112              <parameters> agent0Dur0 agent0Pow0 </parameters>
113          </constraint>
114          <!-- Vincoli soft relativi alle funzioni di costo monetarie. -->
115          <constraint name="Constr_agent0Act0" arity="2" scope="agent0Act0
     agent0Dur0" reference="relation_agent0Act0" />
116          <constraint name="Constr_agent0Act1" arity="2" scope="agent0Act1
     agent0Dur1" reference="relation_agent0Act1" />
117          <constraint name="Constr_agent0Act2" arity="2" scope="agent0Act2
     agent0Dur2" reference="relation_agent0Act2" />
118          <constraint name="Constr_agent1Act0" arity="2" scope="agent1Act0
     agent1Dur0" reference="relation_agent1Act0" />
119          <constraint name="Constr_agent1Act1" arity="2" scope="agent1Act1
     agent1Dur1" reference="relation_agent1Act1" />
120          <constraint name="Constr_agent1Act2" arity="2" scope="agent1Act2
     agent1Dur2" reference="relation_agent1Act2" />
121      </constraints>
122  </instance>
```

Figura A.2: Esempio di istanza DCOP formulata secondo il modello ad attività.

A.3 Formulazione ad attività semplificata

In Figura A.3 è mostrato un esempio di istanza DCOP modellata tramite la formulazione ad attività descritta in Sezione 4.2.3.

Il problema mostrato è lo stesso rappresentato in Figura A.2 (a meno del profilo di consumo utilizzato dall'agente *agent1*), con le differenze dovute alla semplificazione del modello: ogni agente ha un unico profilo di consumo fissato; inoltre, nei vincoli di potenza massima espressi tramite relazioni che assegnano costo infinito alle tuple non ammissibili, non sono più presenti in maniera esplicita le variabili di potenza attività (come si può vedere, infatti, le variabili coinvolte nei vincoli di potenza non sono più 6, bensì 4).

```
<instance>
     <presentation name="modelloAttivitàSemplificato" maxConstraintArity="6"
maximize="false" format="XCSP 2.1_FRODO" />

     <agents nbAgents="2">
          <agent name="agent0" />
          <agent name="agent1" />
     </agents>

     <domains nbDomains="18">
          <domain name="domain_agent0Act0" nbValues="44">0 30 60 90 120 150 180 210
240 270 300 330 360 390 420 450 480 510 540 570 600 630 660 690 720 750 780 810
840 870 900 930 960 990 1020 1050 1080 1110 1140 1170 1200 1230 1260 1290 </
domain>
          <domain name="domain_agent0Act1" nbValues="44">30 60 90 120 150 180 210
240 270 300 330 360 390 420 450 480 510 540 570 600 630 660 690 720 750 780 810
840 870 900 930 960 990 1020 1050 1080 1110 1140 1170 1200 1230 1260 1290 1320 </
domain>
          <domain name="domain_agent0Act2" nbValues="44">120 150 180 210 240 270
300 330 360 390 420 450 480 510 540 570 600 630 660 690 720 750 780 810 840 870
900 930 960 990 1020 1050 1080 1110 1140 1170 1200 1230 1260 1290 1320 1350 1380
1410 </domain>
          <domain name="domain_agent0Dur0" nbValues="1">30 </domain>
          <domain name="domain_agent0Dur1" nbValues="1">90 </domain>
          <domain name="domain_agent0Dur2" nbValues="1">30 </domain>
          <domain name="domain_agent0Pow0" nbValues="1">2120 </domain>
          <domain name="domain_agent0Pow1" nbValues="1">140 </domain>
          <domain name="domain_agent0Pow2" nbValues="1">447 </domain>
          <domain name="domain_agent1Act0" nbValues="46">0 30 60 90 120 150 180 210
240 270 300 330 360 390 420 450 480 510 540 570 600 630 660 690 720 750 780 810
840 870 900 930 960 990 1020 1050 1080 1110 1140 1170 1200 1230 1260 1290 1320
1350 </domain>
          <domain name="domain_agent1Act1" nbValues="46">30 60 90 120 150 180 210
240 270 300 330 360 390 420 450 480 510 540 570 600 630 660 690 720 750 780 810
840 870 900 930 960 990 1020 1050 1080 1110 1140 1170 1200 1230 1260 1290 1320
1350 1380 </domain>
          <domain name="domain_agent1Act2" nbValues="46">60 90 120 150 180 210 240
270 300 330 360 390 420 450 480 510 540 570 600 630 660 690 720 750 780 810 840
870 900 930 960 990 1020 1050 1080 1110 1140 1170 1200 1230 1260 1290 1320 1350
1380 1410 </domain>
          <domain name="domain_agent1Dur0" nbValues="1">30 </domain>
          <domain name="domain_agent1Dur1" nbValues="1">30 </domain>
          <domain name="domain_agent1Dur2" nbValues="1">30 </domain>
          <domain name="domain_agent1Pow0" nbValues="1">2000 </domain>
          <domain name="domain_agent1Pow1" nbValues="1">70 </domain>
          <domain name="domain_agent1Pow2" nbValues="1">2000 </domain>
     </domains>

     <variables nbVariables="18">
          <variable name="agent0Act0" domain="domain_agent0Act0" agent="agent0"/>
          <variable name="agent0Act1" domain="domain_agent0Act1" agent="agent0"/>
          <variable name="agent0Act2" domain="domain_agent0Act2" agent="agent0"/>
          <variable name="agent0Dur0" domain="domain_agent0Dur0" agent="agent0"/>
          <variable name="agent0Dur1" domain="domain_agent0Dur1" agent="agent0"/>
          <variable name="agent0Dur2" domain="domain_agent0Dur2" agent="agent0"/>
          <variable name="agent0Pow0" domain="domain_agent0Pow0" agent="agent0"/>
          <variable name="agent0Pow1" domain="domain_agent0Pow1" agent="agent0"/>
          <variable name="agent0Pow2" domain="domain_agent0Pow2" agent="agent0"/>
          <variable name="agent1Act0" domain="domain_agent1Act0" agent="agent1"/>
          <variable name="agent1Act1" domain="domain_agent1Act1" agent="agent1"/>
          <variable name="agent1Act2" domain="domain_agent1Act2" agent="agent1"/>
          <variable name="agent1Dur0" domain="domain_agent1Dur0" agent="agent1"/>
          <variable name="agent1Dur1" domain="domain_agent1Dur1" agent="agent1"/>
          <variable name="agent1Dur2" domain="domain_agent1Dur2" agent="agent1"/>
          <variable name="agent1Pow0" domain="domain_agent1Pow0" agent="agent1"/>
          <variable name="agent1Pow1" domain="domain_agent1Pow1" agent="agent1"/>
          <variable name="agent1Pow2" domain="domain_agent1Pow2" agent="agent1"/>
     </variables>
```

```
50
51      <relations nbRelations="8">
52          <!-- Funzioni (relazioni) di costo monetario. -->
53          <relation name="relation_agent0Act0" arity="2" nbTuples="44"
    semantics="soft" defaultCost="infinity">
54              34: 0 30|34: 30 30|32: 60 30|32: 90 30|31: 120 30|31: 150 30|30: 180
    30|30: 210 30|30: 240 30|30: 270 30|30: 300 30|30: 330 30|30: 360 30|30: 390 30|
    31: 420 30|31: 450 30|30: 480 30|30: 510 30|28: 540 30|28: 570 30|27: 600 30|27:
    630 30|23: 660 30|23: 690 30|23: 720 30|23: 750 30|19: 780 30|19: 810 30|27: 840
    30|27: 870 30|30: 900 30|30: 930 30|32: 960 30|32: 990 30|37: 1020 30|37: 1050 30|
    41: 1080 30|41: 1110 30|41: 1140 30|41: 1170 30|37: 1200 30|37: 1230 30|35: 1260
    30|35: 1290 30
55          </relation>
56          <relation name="relation_agent0Act1" arity="2" nbTuples="44"
    semantics="soft" defaultCost="infinity">
57              99: 30 90|96: 60 90|95: 90 90|93: 120 90|92: 150 90|91: 180 90|91:
    210 90|91: 240 90|91: 270 90|91: 300 90|92: 330 90|93: 360 90|94: 390 90|94: 420
    90|93: 450 90|90: 480 90|87: 510 90|84: 540 90|82: 570 90|77: 600 90|74: 630 90|
    70: 660 90|70: 690 90|66: 720 90|62: 750 90|66: 780 90|73: 810 90|85: 840 90|89:
    870 90|94: 900 90|95: 930 90|102: 960 90|107: 990 90|116: 1020 90|121: 1050 90|
    125: 1080 90|125: 1110 90|121: 1140 90|116: 1170 90|110: 1200 90|108: 1230 90|103:
    1260 90|100: 1290 90|96: 1320 90
58          </relation>
59          <relation name="relation_agent0Act2" arity="2" nbTuples="44"
    semantics="soft" defaultCost="infinity">
60              31: 120 30|31: 150 30|30: 180 30|30: 210 30|30: 240 30|30: 270 30|
    30: 300 30|30: 330 30|30: 360 30|30: 390 30|31: 420 30|31: 450 30|30: 480 30|30:
    510 30|28: 540 30|28: 570 30|27: 600 30|27: 630 30|23: 660 30|23: 690 30|23: 720
    30|23: 750 30|19: 780 30|19: 810 30|27: 840 30|27: 870 30|30: 900 30|30: 930 30|
    32: 960 30|32: 990 30|37: 1020 30|37: 1050 30|41: 1080 30|41: 1110 30|41: 1140 30|
    41: 1170 30|37: 1200 30|37: 1230 30|35: 1260 30|35: 1290 30|32: 1320 30|32: 1350
    30|31: 1380 30|31: 1410 30
61          </relation>
62          <relation name="relation_agent1Act0" arity="2" nbTuples="46"
    semantics="soft" defaultCost="infinity">
63              34: 0 30|34: 30 30|32: 60 30|32: 90 30|31: 120 30|31: 150 30|30: 180
    30|30: 210 30|30: 240 30|30: 270 30|30: 300 30|30: 330 30|30: 360 30|30: 390 30|
    31: 420 30|31: 450 30|30: 480 30|30: 510 30|28: 540 30|28: 570 30|27: 600 30|27:
    630 30|23: 660 30|23: 690 30|23: 720 30|23: 750 30|19: 780 30|19: 810 30|27: 840
    30|27: 870 30|30: 900 30|30: 930 30|32: 960 30|32: 990 30|37: 1020 30|37: 1050 30|
    41: 1080 30|41: 1110 30|41: 1140 30|41: 1170 30|37: 1200 30|37: 1230 30|35: 1260
    30|35: 1290 30|32: 1320 30|32: 1350 30
64          </relation>
65          <relation name="relation_agent1Act1" arity="2" nbTuples="46"
    semantics="soft" defaultCost="infinity">
66              34: 30 30|32: 60 30|32: 90 30|31: 120 30|31: 150 30|30: 180 30|30:
    210 30|30: 240 30|30: 270 30|30: 300 30|30: 330 30|30: 360 30|30: 390 30|31: 420
    30|31: 450 30|30: 480 30|30: 510 30|28: 540 30|28: 570 30|27: 600 30|27: 630 30|
    23: 660 30|23: 690 30|23: 720 30|23: 750 30|19: 780 30|19: 810 30|27: 840 30|27:
    870 30|30: 900 30|30: 930 30|32: 960 30|32: 990 30|37: 1020 30|37: 1050 30|41:
    1080 30|41: 1110 30|41: 1140 30|41: 1170 30|37: 1200 30|37: 1230 30|35: 1260 30|
    35: 1290 30|32: 1320 30|32: 1350 30|31: 1380 30
67          </relation>
68          <relation name="relation_agent1Act2" arity="2" nbTuples="46"
    semantics="soft" defaultCost="infinity">
69              32: 60 30|32: 90 30|31: 120 30|31: 150 30|30: 180 30|30: 210 30|30:
    240 30|30: 270 30|30: 300 30|30: 330 30|30: 360 30|30: 390 30|31: 420 30|31: 450
    30|30: 480 30|30: 510 30|28: 540 30|28: 570 30|27: 600 30|27: 630 30|23: 660 30|
    23: 690 30|23: 720 30|23: 750 30|19: 780 30|19: 810 30|27: 840 30|27: 870 30|30:
    900 30|30: 930 30|32: 960 30|32: 990 30|37: 1020 30|37: 1050 30|41: 1080 30|41:
    1110 30|41: 1140 30|41: 1170 30|37: 1200 30|37: 1230 30|35: 1260 30|35: 1290 30|
    32: 1320 30|32: 1350 30|31: 1380 30|31: 1410 30
70          </relation>
71          <!-- Relazioni relative ai vincoli di potenza massima disponibile. -->
72          <relation name="relation_agent0Pow0agent1Pow0" arity="4" nbTuples="44"
    semantics="soft" defaultCost="0">
73              infinity: 0 30 0 30|30 30 30 30|60 30 60 30|90 30 90 30|120 30 120
    30|150 30 150 30|180 30 180 30|210 30 210 30|240 30 240 30|270 30 270 30|300 30
```

```
300 30|330 30 330 30|360 30 360 30|390 30 390 30|420 30 420 30|450 30 450 30|480
30 480 30|510 30 510 30|540 30 540 30|570 30 570 30|600 30 600 30|630 30 630 30|
660 30 660 30|690 30 690 30|720 30 720 30|750 30 750 30|780 30 780 30|810 30 810
30|840 30 840 30|870 30 870 30|900 30 900 30|930 30 930 30|960 30 960 30|990 30
990 30|1020 30 1020 30|1050 30 1050 30|1080 30 1080 30|1110 30 1110 30|1140 30
1140 30|1170 30 1170 30|1200 30 1200 30|1230 30 1230 30|1260 30 1260 30|1290 30
1290 30
        </relation>
        <relation name="relation_agent0Pow0agent1Pow2" arity="4" nbTuples="42"
semantics="soft" defaultCost="0">
            infinity: 60 30 60 30|90 30 90 30|120 30 120 30|150 30 150 30|180 30
180 30|210 30 210 30|240 30 240 30|270 30 270 30|300 30 300 30|330 30 330 30|360
30 360 30|390 30 390 30|420 30 420 30|450 30 450 30|480 30 480 30|510 30 510 30|
540 30 540 30|570 30 570 30|600 30 600 30|630 30 630 30|660 30 660 30|690 30 690
30|720 30 720 30|750 30 750 30|780 30 780 30|810 30 810 30|840 30 840 30|870 30
870 30|900 30 900 30|930 30 930 30|960 30 960 30|990 30 990 30|1020 30 1020 30|
1050 30 1050 30|1080 30 1080 30|1110 30 1110 30|1140 30 1140 30|1170 30 1170 30|
1200 30 1200 30|1230 30 1230 30|1260 30 1260 30|1290 30 1290 30
        </relation>
    </relations>

    <predicates nbPredicates="2">
        <!-- Predicati che esprimono necessarietà di continuità tra le attività
di un agente. -->
        <predicate name="predicate_continuity_agent0">
            <parameters> agent0Act0 agent0Act1 agent0Act2 agent0Dur0 agent0Dur1
agent0Dur2 </parameters>
            <expression>
                <functional>
and(eq(agent0Act1,add(agent0Act0,agent0Dur0)),eq(agent0Act2,add(agent0Act1,agent0D
ur1))) </functional>
            </expression>
        </predicate>
        <predicate name="predicate_continuity_agent1">
            <parameters> agent1Act0 agent1Act1 agent1Act2 agent1Dur0 agent1Dur1
agent1Dur2 </parameters>
            <expression>
                <functional>
and(eq(agent1Act1,add(agent1Act0,agent1Dur0)),eq(agent1Act2,add(agent1Act1,agent1D
ur1))) </functional>
            </expression>
        </predicate>
    </predicates>

    <constraints nbConstraints="10">
        <!-- Vincoli hard per modellare continuità dei profili di consumo. -->
        <constraint name="Constr_continuity_agent0" arity="6" scope="agent0Act0
agent0Act1 agent0Act2 agent0Dur0 agent0Dur1 agent0Dur2 "
reference="predicate_continuity_agent0">
            <parameters> agent0Act0 agent0Act1 agent0Act2 agent0Dur0 agent0Dur1
agent0Dur2 </parameters>
        </constraint>
        <constraint name="Constr_continuity_agent1" arity="6" scope="agent1Act0
agent1Act1 agent1Act2 agent1Dur0 agent1Dur1 agent1Dur2 "
reference="predicate_continuity_agent1">
            <parameters> agent1Act0 agent1Act1 agent1Act2 agent1Dur0 agent1Dur1
agent1Dur2 </parameters>
        </constraint>
        <!-- Vincoli hard per modellare potenza massima disponibile. -->
        <constraint name="Constr_agent0Pow0agent1Pow0" arity="4"
scope="agent0Act0 agent0Dur0 agent1Act0 agent1Dur0"
reference="relation_agent0Pow0agent1Pow0" />
        <constraint name="Constr_agent0Pow0agent1Pow2" arity="4"
scope="agent0Act0 agent0Dur0 agent1Act2 agent1Dur2"
reference="relation_agent0Pow0agent1Pow2" />
        <!-- Vincoli soft relativi alle funzioni di costo monetarie. -->
        <constraint name="Constr_agent0Act0" arity="2" scope="agent0Act0
agent0Dur0" reference="relation_agent0Act0" />
```

```
109             <constraint name="Constr_agent0Act1" arity="2" scope="agent0Act1
    agent0Dur1" reference="relation_agent0Act1" />
110             <constraint name="Constr_agent0Act2" arity="2" scope="agent0Act2
    agent0Dur2" reference="relation_agent0Act2" />
111             <constraint name="Constr_agent1Act0" arity="2" scope="agent1Act0
    agent1Dur0" reference="relation_agent1Act0" />
112             <constraint name="Constr_agent1Act1" arity="2" scope="agent1Act1
    agent1Dur1" reference="relation_agent1Act1" />
113             <constraint name="Constr_agent1Act2" arity="2" scope="agent1Act2
    agent1Dur2" reference="relation_agent1Act2" />
114         </constraints>
115 </instance>
116
```

Figura A.3: Esempio di istanza DCOP formulata secondo il modello ad attività semplificato.