Politecnico Di Milano
Corso di Laurea Specialistica in Ingegneria Informatica
Dipartimento di Elettronica e Informazione

Studio di sistemi idrici come problemi di ottimizzazione distribuiti multiobiettivo: modelli e algoritmi

Laboratorio di Intelligenza Artificiale
e Robotica del Politecnico di Milano

Relatore: Prof. Francesco AMIGONI
Correlatore: Ing. Matteo GIULIANI

Tesi di Laurea di:
Enrico BONTEMPI, matricola 754979

Anno Accademico 2011-2012

Sommario

Questa tesi si colloca nell'ambito della modellazione e analisi di sistemi ambientali, per i quali vengono utilizzati metodi e algoritmi sviluppati nel campo dei sistemi multiagente.
Lo scopo della tesi è la ricerca sia di un formalismo in grado di modellare sistemi idrici nei quali interagiscono molteplici agenti, sia di un algoritmo in grado di risolvere il problema dell'ottimizzazione distribuita degli obiettivi di ogni singolo agente e del sistema globale.
In questo lavoro di tesi è stato adottato il formalismo MO-DCOP, la cui espressività permette di modellare adeguatamente i problemi di ottimizzazione dei sistemi idrici oggetto di studio, ed è stato implementato un algoritmo in grado di analizzare e risolvere i problemi così formulati.

Ringraziamenti

Desidero ringraziare la mia famiglia per avermi garantito le condizioni per affrontare il mio ciclo di studi dedicandovi sempre la massima priorità.

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Introduzione

1.1 Inquadramento generale

Questa tesi si colloca nell'ambito della modellazione e analisi di sistemi ambientali, per i quali vengono utilizzati metodi e algoritmi sviluppati nel campo dei sistemi multiagente.
Lo scopo della tesi è la ricerca di un formalismo in grado di modellare sistemi idrici nei quali interagiscono molteplici agenti, ed un algoritmo in grado di risolvere il problema dell'ottimizzazione distribuita degli obiettivi di ogni singolo agente e del sistema globale.
Questo lavoro di tesi è stato focalizzato sulla ricerca nello stato dell'arte di formalismi dotati dell'espressività necessaria per modellare i problemi idrici oggetto di questo studio; tra questi è stato identificato un formalismo particolarmente adeguato, chiamato MO-DCOP, del quale sono state analizzate le proprietà in riferimento al particolare caso di studio di questa tesi. Si sono quindi delineate le regole per la modellazione di problemi idrici attraverso MO-DCOP, ed è stato infine implementato un algoritmo risolutivo.

1.2 Breve descrizione del lavoro

I sistemi idrici sono spesso modellati come sistemi multiagente. In questi sistemi [26] vi sono diversi agenti, ciascuno con un obiettivo locale, che compiono determinate decisioni per ottimizzare il proprio obiettivo; esiste inoltre un obiettivo globale del sistema, gestito da un'autorità che impone delle restrizioni ai vari agenti affinchè il sistema nel suo complesso non venga a trovarsi in condizioni sconvenienti.
In campo ambientale, tuttavia, gli agenti vengono spesso usati solo come simulatori (che applicano politiche fisse e definite a priori) e non come ot-

timizzatori (che decidono come comportarsi in base allo stato attuale del sistema). Non esistono pertanto nella letteratura sui sistemi multiagente per problemi ambientali metodi in grado di risolvere problemi di ottimizzazione in cui i diversi agenti di un sistema effettuano delle scelte per ottimizzare un proprio obiettivo, identificando le scelte migliori per gli agenti e per il sistema.
Nell'ambito dell'Intelligenza Artificiale, invece, i sistemi multiagente offrono diversi formalismi per la modellazione di problemi di ottimizzazione. In particolare, i formalismi DCSP [27] e DCOP [19] rappresentano rispettivamente problemi di soddisfacimento e ottimizzazione di vincoli distribuiti tra diversi agenti. Un sistema idrico può essere facilmente modellato con uno di questi formalismi, rappresentando attraverso vincoli le dipendenze e le relazioni tra gli agenti.
Tuttavia, nel caso dei sistemi idrici, l'ottimizzazione non coinvolge un unico obiettivo, poiché i vari agenti e l'autorità competente perseguono obiettivi distinti. Pertanto, per analizzare correttamente questi sistemi, occorre affrontarli utilizzando un approccio multiobiettivo. Per questi motivi, è stato individuato il formalismo MO-DCOP [7], che generalizza un DCOP al caso multiobiettivo e possiede l'espressività necessaria per modellare i sistemi idrici oggetto di questo lavoro.
In questo lavoro di tesi viene utilizzato il concetto di Pareto-ottimalità per identificare l'insieme delle soluzioni ottimali, chiamato *frontiera di Pareto*, dei sistemi idrici oggetto di studio, in quanto l'obiettivo dell'analisi di questi sistemi è la ricerca di tutte le possibili soluzioni ottimali, e non di una singola soluzione ottimale, per poterle confrontare e per valutare come diverse decisioni degli agenti indirizzino verso soluzioni diverse.
Sono stati quindi analizzati gli algoritmi risolutivi per MO-DCOP: B-MOMS [7] e MO-Adopt [18], che, però, producono come risultato un'unica soluzione ottimale appartenente alla frontiera di Pareto. Si è pertanto cercato di modificare gli algoritmi per calcolare l'intera frontiera di Pareto del problema da risolvere.
La modifica dell'algoritmo MO-Adopt si è rivelata sconveniente e computazionalmente onerosa, mentre l'algoritmo B-MOMS è stato modificato efficientemente, approfondendo i concetti sui quali l'algoritmo stesso è costruito. Il risultato della modifica dell'algoritmo B-MOMS è l'algoritmo MO-Max-Sum presentato in questo lavoro.

1.3 Struttura della tesi

La tesi è strutturata nel modo seguente:
Nel Capitolo 2 vengono introdotte le nozioni e gli strumenti su cui si basa questa tesi, sono descritti i sistemi multiagente e i problemi di soddisfacimento e ottimizzazione di vincoli distribuiti.
Nel Capitolo 3 viene introdotto l'obiettivo di questo lavoro di tesi, vengono descritti i sistemi idrici oggetto di questo studio e discussi i formalismi attraverso cui modellarli.
Nel Capitolo 4 è descritta la soluzione per il problema oggetto di questo studio, viene motivata la scelta del formalismo MO-DCOP come strumento di modellazione e viene delineata la struttura dell'algoritmo risolutivo implementato.
Nel Capitolo 5 vengono presentati i risultati sperimentali del lavoro svolto, discutendone la validità, con un'analisi sulla complessità dell'algoritmo risolutivo.
Nel Capitolo 6 sono riassunti gli obiettivi di questa tesi, le valutazioni del lavoro svolto e le prospettive future.

Capitolo 2

Stato dell'arte

In questo capitolo vengono introdotte le nozioni e gli strumenti su cui si basa questo lavoro di tesi. Sono descritti in particolare i problemi di soddisfacimento e ottimizzazione di vincoli distribuiti, applicati all'ambito dei sistemi multiagente.

2.1 Sistemi multiagente

Un sistema multiagente [25] è un sistema composto da molteplici agenti intelligenti che interagiscono tra loro in un determinato ambiente. I sistemi multiagente possono essere usati per risolvere problemi la cui risoluzione si rivela difficile o impossibile per un singolo agente o per un sistema monolitico. La ricerca sui sistemi multiagente trova applicazioni in campi come i mercati online, la reazione ai disastri, la modellazione di strutture sociali.

2.1.1 Agenti intelligenti

Un *agente* è un'entità che riceve percezioni dall'ambiente in cui è collocato attraverso dei sensori e agisce su tale ambiente attraverso degli attuatori. Un agente umano usa occhi, orecchie come sensori, gli arti come attuatori. Un agente robotico può avere telecamere o accelerometri come sensori, motori come attuatori. Un agente software riceve input da tastiera, contenuti di file e pacchetti di rete come input sensoriali e agisce sull'ambiente visualizzando informazioni, scrivendo file e inviando pacchetti di rete. Una percezione consiste negli input percettivi di un agente in un dato istante; per un agente, una sequenza di percezioni costituisce la storia completa di tutto ciò che l'agente ha percepito. In generale, una scelta d'azione di un agente in un istante qualunque dipende dall'intera sequenza di percezioni osservata fino

a quell'istante. In termini matematici, il comportamento di un agente è descritto dalla *funzione agente* che associa una qualsiasi sequenza di percezioni ad un'azione. Per un agente artificiale, la funzione agente sarà internamente implementata da un *programma agente*. La funzione agente è una descrizione matematica astratta, mentre il programma agente è un'implementazione concreta che funziona sull'architettura dell'agente.
Un *agente razionale* è un agente che compie l'azione corretta; concettualmente è un agente che possiede una funzione agente costruita correttamente. In prima approssimazione, l'azione corretta è l'azione che garantisce all'agente il maggiore successo. Per definire il concetto di successo, è necessario definire una *misura di performance*, che incorpora i criteri di successo per il comportamento di un agente. La regola generale per definire la misura delle performance prevede che queste vengano descritte in riferimento allo stato dell'ambiente piuttosto che al comportamento dell'agente. Ad esempio, la misura delle performance di un agente che deve pulire una stanza non è quanto lavoro l'agente effettua per pulire, ma quanto è pulita la stanza. La razionalità di un agente in un determinato istante dipende dalla misura delle performance che definisce i criteri di successo, dalle precedenti conoscenze dell'agente, dalle azioni che l'agente può compiere, e dalla sequenza attuale delle percezioni dell'agente. Da qui la definizione di agente razionale, che per ogni possibile sequenza di percezioni, sceglie l'azione che si prevede massimizzi la propria misura di performance, data l'evidenza fornita dalla sequenza delle percezioni e dalla conoscenza interna dell'agente.
Esiste una vasta gamma di ambienti in cui si collocano i sistemi multiagente; è possibile identificare alcune proprietà secondo le quali categorizzare questi ambienti. Queste proprietà determinano le linee guida per il progetto degli agenti e l'applicabilità di ciascuna delle principali famiglie di tecniche per la loro implementazione. Nell'ambito di questa tesi, l'ambiente in cui sono collocati gli agenti è caratterizzato dalle seguenti proprietà:

- Parzialmente osservabile: gli agenti non hanno accesso allo stato completo dell'ambiente e a tutte le informazioni rilevanti per la scelta delle azioni, ogni agente possiede solamente le proprie informazioni locali e quelle che può rilevare dagli agenti con cui comunica o entra in contatto.

- Deterministico: dato un determinato istante, lo stato successivo in cui l'ambiente verrà a trovarsi è completamente determinato dallo stato attuale e dalle azioni eseguite dagli agenti.

- Episodico: l'esperienza di ogni agente è divisa in episodi atomici, consistenti di un insieme di percezioni e dall'esecuzione di una singola azione; la scelta di un'azione in un singolo episodio dipende solo dall'episodio stesso.
- Statico: la struttura dell'ambiente non muta mentre un agente delibera un'azione, non occorre che un agente controlli continuamente l'ambiente mentre decide quale azione compiere.
- Discreto: un sistema idrico, come tutti i sistemi naturali, è un sistema continuo; è tuttavia necessaria una discretizzazione del tempo, dei domini delle variabili e dei parametri del sistema per effettuarne un'analisi algoritmica, che sarebbe impraticabile se occorresse considerare un'infinità di valori implicata dalla continuità di un sistema. Pertanto l'ambiente viene modellato come ambiente discreto.
- Multiagente: ogni agente deve affrontare, oltre all'ambiente, anche gli altri agenti. In queste condizioni si distinguono due tipi di comportamento degli agenti: quello competitivo, in cui ogni agente propende a scegliere azioni che massimizzino le proprie performance locali, e quello cooperativo, in cui gli agenti adottano un comportamento che porti a massimizzare le prestazioni globali di tutti gli agenti. La progettazione degli agenti negli ambienti multiagente è spesso differente dalla progettazione negli ambienti monoagente. Negli ambienti multiagente la comunicazione emerge come comportamento razionale.

2.1.2 Struttura di un agente

Il compito dell'Intelligenza Artificiale è la progettazione del programma agente che implementa la funzione agente associando alle percezioni delle azioni. Assumendo che questo programma funzioni su una sorta di dispositivo informatico con sensori e attuatori fisici, chiamato architettura, si ha che un agente è composto da un'architettura e da un programma appropriato.
Esistono quattro tipi di programma agente che interpretano i principi alla base di tutti i sistemi intelligenti:

- Gli agenti passivi semplici selezionano le azioni sulla base della percezione corrente, ignorando il resto della storia delle percezioni; tipicamente agiscono secondo un insieme di regole condizione-azione (if-then).
- Gli agenti passivi basati su modelli mantengono una sorta di stato interno che tiene traccia delle parti del mondo che non sono visibili in

un dato istante, costruendo così un modello del modo in cui il modo funziona ed evolve.

- Gli agenti basati su obiettivi scelgono le azioni da compiere sulla base di un obiettivo che perseguono. Ricerca e pianificazione sono sottocampi dell'Intelligenza Artificiale dediti a trovare sequenze di azioni che permettano di raggiungere l'obiettivo di un agente.

- Gli agenti basati su utilità usano delle funzioni di utilità che misurano le preferenze tra diversi stati del mondo. In un modello probabilistico, la scelta delle azioni si basa sull'ottenimento della migliore utilità attesa, computata come la media di tutte le possibili utilità ottenibili negli stati raggiungibili, pesate dalla probabilità di ottenere tali guadagni.

Nei sistemi in analisi in questo lavoro di tesi possono essere presenti sia agenti attivi, che possono agire sull'ambiente prendendo determinate decisioni, che agenti passivi, che subiscono semplicemente le decisioni degli altri agenti in quanto non possono prendere decisioni e compiere azioni che influenzano l'ambiente o gli altri agenti. Gli agenti attivi sono agenti basati su utilità, poiché per ciascuno di essi è definita una funzione che ne rappresenta il guadagno locale, e pertanto basano le proprie decisioni sull'ottimizzazione della propria funzione di utilità locale.

2.2 Distributed Constraint Satisfaction Problem

2.2.1 Constraint Satisfaction Problem

Un Constraint Satisfaction Problem (CSP) [2] è un problema matematico definito su un insieme di variabili il cui valore deve soddisfare un certo numero di vincoli o restrizioni. Un CSP rappresenta le entità in un problema come una serie omogenea di vincoli posti tra delle variabili. I CSP sono oggetto di intense ricerche nel campo dell'Intelligenza Artificiale e della Ricerca Operativa, data la regolarità con cui la loro formulazione procura una base comune per analizzare e risolvere molteplici famiglie di problemi. Spesso la risoluzione dei CSP richiede un'elevata complessità computazionale, poiché richiede una combinazione di euristiche e ricerca combinatoria.

Definizione 1 (CSP [2])**.** *Un CSP consiste di n variabili* $x_1, x_2, \ldots, x_n$*, che assumono valori appartenenti a domini finiti discreti* $D_1, D_2, \ldots, D_n$*, rispettivamente, e un insieme di vincoli tra le variabili. Un vincolo è definito da un predicato, ovvero il vincolo* $p_k(x_{k1}, \ldots, x_{kj})$ *è un predicato definito sul prodotto Cartesiano* $D_{k1} \times \cdots \times D_{kj}$*. Questo predicato è vero se e solo se*

l'assegnamento di valori a queste variabili soddisfa il vincolo. Risolvere un CSP equivale a trovare un assegnamento di valori a tutte le variabili in modo che tutti i vincoli siano soddisfatti.

2.2.2 Distributed CSP

Un Distributed CSP [27] è un CSP in cui le variabili e i vincoli sono distribuiti tra più agenti autonomi, per i quali si assume il seguente modello di comunicazione:

- Gli agenti comunicano tra di loro tramite scambio di messaggi. Un agente può inviare messaggi ad altri agenti unicamente se esso conosce gli indirizzi di questi agenti.
- Il ritardo dovuto alla consegna dei messaggi è finito, ma casuale. Nella trasmissione tra qualunque coppia di agenti, i messaggi vengono ricevuti nel medesimo ordine nel quale sono inviati.

Occorre notare che questo modello non impone necessariamente che la rete fisica di comunicazione tra gli agenti debba essere completamente connessa (cioè un grafo completo). A differenza di altri studi su algoritmi distribuiti, nei quali la topologia della rete di comunicazione fisica ricopre un ruolo importante, per i DCSP si assume l'esistenza di una struttura di comunicazione sottostante affidabile tra gli agenti senza quindi interessarsi ai dettagli della rete di comunicazione.
Ogni agente controlla qualche variabile e cerca di determinarne i valori. Esistono comunque dei vincoli inter-agenti, e l'assegnamento di valori deve soddisfare questi vincoli. Formalmente:

Definizione 2 (DCSP [27])**.** *Un DCSP consiste di m agenti $1, 2, \ldots, m$. Ogni variabile x_j appartiene ad un agente i (relazione rappresentata come $belongs(x_j, i)$). I vincoli sono distribuiti tra gli agenti: il fatto che un agente l conosca un predicato di vincolo p_k viene rappresentato come $known(p_k, l)$. Un DCSP è risolto se e solo se sono soddisfatte le seguenti condizioni:*

- $\forall i, \forall x_j$ *tali che* $belongs(x_j, i)$*, il valore di* x_j *è assegnato a* d_j*;*
- $\forall l, \forall p_k$ *tali che* $known(p_k, l)$*,* p_k *è vero per l'assegnamento* $x_j = d_j$*.*

Ogni agente può pertanto controllare più di una variabile. Si può anche considerare il caso in cui diversi agenti controllano una stessa variabile: questo caso può essere formalizzato come se gli stessi agenti controllassero variabili differenti, con l'aggiunta di dei vincoli che impongano che queste variabili assumano lo stesso valore. In generale si assume che ogni agente controlli una sola variabile.

2.2.3 Algoritmi risolutivi

I metodi per la risoluzione di un CSP (e Distributed CSP) possono essere divisi in due classi: algoritmi di ricerca (come per esempio *backtracking*) e algoritmi di consistenza. Gli algoritmi di consistenza sono procedure di pre-elaborazione invocate prima della ricerca.
Tra gli algoritmi di ricerca rientrano algoritmi banali come l'approccio centralizzato in cui viene eletto un leader tra gli agenti che raccoglie informazioni su variabili, domini e vincoli, e risolve il problema da solo con algoritmi CSP centralizzati. Questo metodo comporta un notevole overhead di comunicazioni e crea un singolo punto di fallimento, oltre a non considerare la privacy dei singoli agenti che sono costretti a comunicare le proprie informazioni all'agente leader.
Si può modificare l'algoritmo di backtracking standard per CSP per creare un algoritmo di backtracking sincrono per DCSP. Assumendo che gli agenti si accordino su un ordine di istanziazione delle variabili, ogni agente riceve una soluzione parziale (l'istanziazione delle variabili) dagli agenti che lo precedono, e istanzia la propria variabile basandosi sui vincoli che esso conosce. Se trova un valore consistente per l'assegnamento, lo aggiunge alla soluzione parziale e procede passando il messaggio al prossimo agente. Se invece non trova nessuna istanziazione che soddisfi i vincoli, invia un messaggio di backtracking agli agenti che lo precedono. Questo algoritmo richiede tuttavia un certo costo di comunicazione per la determinazione dell'ordine di istanziazione, e non può in alcun modo sfruttare il parallelismo poiché, in un dato momento, un solo agente riceve la soluzione parziale e vi lavora, pertanto il problema è risolto sequenzialmente.

2.2.4 Asynchronous Backtracking

L'Asynchronous Backtracking [27] rimuove gli svantaggi del synchronous backtracking consentendo agli agenti di operare in modo concorrente e asincrono. Ogni agente istanzia la propria variabile e comunica il valore agli agenti pertinenti. Assumendo che i vincoli siano binari e monodirezionali (ovvero in cui un agente impone un vincolo ad un secondo agente), per ogni vincolo uno dei due agenti coinvolti (*value-sending*) invia il proprio valore all'altro agente (*constraint-evaluating*) che controlla e valuta il vincolo.
Se gli agenti dovessero cambiare i propri valori continuamente senza mai raggiungere uno stato stabile, si verrebbero allora a trovare in un ciclo infinito. Un metodo per evitare cicli è l'uso di una relazione d'ordine definendo un ordine di priorità tra gli agenti.
Gli agenti scambiano tra loro due tipi di messaggi:

- messaggio *ok?* inviato dall'agente value-sending per chiedere all'agente constraint-evaluating se il valore che ha scelto è accettabile.

- messaggio *nogood* inviato dall'agente constraint-evaluating all'agente value-sending nel caso il valore che ha scelto non sia accettabile.

Ogni agente può coprire entrambi i ruoli (value-sending e constraint-evaluating) in base ai vincoli che lo riguardano. L'insieme dei valori che un agente riceve da altri agenti tramite i collegamenti corrispondenti ai vincoli da questi imposti costituisce la *vista* dell'agente. Ogni volta che un agente riceve un messaggio *ok?* aggiunge il dato alla propria vista (o lo aggiorna) e verifica che il proprio assegnamento (rappresentato dalla coppia $(x_i, current\_value)$) sia consistente con la vista. L'assegnamento è consistente con la vista se tutti i vincoli che l'agente valuta sono soddisfatti per l'assegnamento di valori descritto nella vista e per $(x_i, current\_value)$, e se tutti i messaggi *nogood* sono non compatibili con la vista dell'agente e $(x_i, current\_value)$ (compatibile significa che tutte le variabili contenute nel messaggio nogood hanno gli stessi valori della vista e del proprio assegnamento). Se l'assegnamento non è consistente con la vista, l'agente x_i prova a cambiare il proprio valore per renderlo consistente con la vista. Se nessun valore rende consistente l'assegnamento con la vista, l'agente stesso opera un *backtrack* inviando un messaggio *nogood*, contenente il sottoinsieme della propria vista che rende inconsistente ogni assegnamento possibile, ad un agente con priorità superiore.
In questo caso, per ogni vincolo, l'agente con priorità minore sarà il valutatore, mentre l'agente con priorità maggiore invierà messaggi *ok?* al primo agente. Pertanto il flusso dei messaggi *nogood* procederà verso agenti con priorità sempre superiore.
L'Asynchronous Backtracking è un algoritmo completo in quanto, se esiste una soluzione, raggiunge uno stato in cui tutti gli assegnamenti alle variabili soddisfano tutti i vincoli, mentre se non esiste soluzione al problema l'algoritmo termina senza generare alcuna soluzione.

2.3 Distributed Constraint Optimization Problem

Una vasta classe di problemi di coordinamento multiagente quali pianificazione distribuita, scheduling distribuito, allocazione di risorse distribuita ed altri, possono essere modellati come Distributed Constraint Optimization Problem (DCOP). Vi sono molteplici applicazioni multiagente, tra cui costellazioni satellitari, robot riconfigurabili, organizzazioni uomo-agenti, reti

di sensori, in cui si presentano problematiche di ragionamento e coordinamento distribuiti. DCOP fornisce un utile framework per studiare come degli agenti possono coordinare le proprie decisioni in questi domini.
Un DCOP include un insieme di variabili, ciascuna variabile è assegnata ad un agente che ne controlla il valore, e gli agenti devono coordinare le proprie scelte affinchè una funzione obiettivo globale venga ottimizzata. Questa funzione obiettivo globale viene modellata come un insieme di vincoli, e ogni agente conosce i vincoli nei quali la propria variabile è coinvolta. In un problema DCSP i vincoli sono descritti da predicati che restituiscono un valore booleano Vero/Falso, mentre in un DCOP i vincoli restituiscono dei valori numerici appartenenti ad un certo range. Pertanto il framework DCOP generalizza il framework Distributed Constraint Satisfaction Problem (DCSP), in cui le soluzioni del problema sono caratterizzate con una designazione di "soddisfacibile o insoddisfacibile" e quindi non è in grado di modellare problemi dove le soluzioni hanno un grado di qualità o costo.

Definizione 3 (DCOP [19])**.** *Un DCOP consiste di un insieme V di n variabili $V = \{x_1, x_2, \ldots, x_n\}$, ciascuna assegnata ad un agente, che assumono valori da domini finiti e discreti $D_1, D_2, \ldots, D_n$, rispettivamente. Soltanto l'agente che controlla una variabile ne controlla il valore e conosce il dominio. L'obiettivo per gli agenti è la scelta di valori per le variabili che minimizzino una data funzione obiettivo globale. Questa funzione obiettivo è descritta come la somma su un insieme di funzioni costo. Una funzione costo per una coppia di variabili x_i, x_j è definita come $f_{ij} : D_i \times D_j \to \mathbb{N}$. Le funzioni costo in DCOP sono l'analogo dei vincoli in DCSP, e vengono talvolta citate come vincoli "valutati" o "soft". Due agenti x_i, x_j sono considerati vicini se esiste un vincolo tra i due. L'obiettivo è trovare un assegnamento $\mathcal{A}^*$ di valori alle variabili tale che la funzione aggregata F sia minimizzata. F è definita come:*

$$F(\mathcal{A}) = \sum_{x_i, x_j \in V} f_{ij}(d_i, d_j) \text{ , dove } x_i \leftarrow d_i, x_j \leftarrow d_j \text{ in } \mathcal{A}$$

I termini agente e variabile vengono in questo ambito spesso intercambiati. Per un problema di massimizzazione, si parla di funzioni utilità anzichè di funzioni costo.

2.3.1 Algoritmi risolutivi

La risoluzione di DCOP richiede tecniche che oltrepassano i metodi per trovare soluzioni distribuite soddisfacibili e la loro semplice estensione per l'ottimizzazione. Un metodo DCOP, per poter essere usato per le applicazioni sopra citate, deve soddisfare tre requisiti chiave. Primo, poiché i domini

sono inerentemente distribuiti, occorre un metodo in cui gli agenti possono ottimizzare una funzione globale in stile distribuito usando soltanto comunicazioni locali (comunicazioni con gli agenti vicini); metodi in cui tutti gli agenti devono comunicare con un singolo agente centrale sono inaccettabili. Secondo, occorre un metodo in grado di trovare soluzioni velocemente consentendo agli agenti di operare asincronamente; un metodo sincrono in cui un agente si pone in attesa di un particolare messaggio da parte di un altro agente non è accettabile poiché comporta uno spreco di tempo che potrebbe essere invece potenzialmente usato per altri scopi. Infine, occorrono qualità di garanzia dimostrabili sulle performance del sistema. Uno dei principali ostacoli per la risoluzione di DCOP è la combinazione di garanzie di qualità con asincronismo.
Gli algoritmi per la risoluzione di DCOP si possono suddividere in tre classi. La prima contiene algoritmi progettati per trovare soluzioni ottimali, come Adopt [19] e DPOP [22], che hanno complessità computazionale e di comunicazione esponenziale con il numero di agenti. La seconda classe è composta da algoritmi come l'algoritmo stocastico distribuito (DSA) [9] o Maximum Gain Message (MGM) [15], progettati per sistemi composti da un grande numero di agenti, ma che spesso convergono a soluzioni di bassa qualità. Esiste una terza classe di algoritmi tipicamente riferiti alla Generalised Distributive Law (GDL) [1], che costituiscono un compromesso tra gli estremi rappresentati dalle prime due classi, e possono essere usati per elaborare soluzioni approssimate di qualità accettabile. In particolare, un algoritmo GDL, l'algoritmo *Max-Sum*, è in grado di generare soluzioni che si avvicinano all'ottimo più che (ad esempio) DSA, restando comunque robusto contro la perdita di messaggi e mostrando alta scalabilità nei costi computazionali e di comunicazione [23]. Questo algoritmo non garantisce la convergenza ad una soluzione nel caso di grafi di vincoli ciclici. *Bounded* Max-Sum [23] è un'estensione all'algoritmo Max-Sum standard, che risolve questo difetto potando il grafo dei vincoli e riducendolo ad un albero. In questo modo è possibile fornire delle garanzie di qualità sulle soluzioni elaborate.

2.3.2 Asynchronous Distributed Optimization

L'Asynchronous Distributed Optimization (Adopt [19]) è un algoritmo DCOP in grado di trovare la soluzione ottimale al problema, o una soluzione che rientra in una distanza dall'ottimo definita dall'utente, utilizzando solamente comunicazioni asincrone localizzate e con complessità spaziale polinomiale per ciascun agente. La comunicazione è locale in quanto un agente non invia messaggi a tutti gli altri agenti, ma solamente agli agenti vicini, ovvero legati

all'agente da un vincolo. I principi su cui si basa Adopt per la risoluzione di un DCOP sono equivalenti a quelli su cui si basa l'Asynchronous Backtracking per risolvere un DCSP.
L'idea da cui nasce Adopt è di ottenere asincronismo permettendo ad ogni agente di modificare il valore della propria variabile quando individua la possibilità che qualche altra soluzione possa essere migliore di quella attualmente sotto indagine. Questa strategia di ricerca permette la computazione asincrona perché un agente non necessita di informazioni globali per effettuare le proprie decisioni locali, ma può prendere decisioni solo con le informazioni locali. La seconda idea chiave in Adopt è la ricostruzione efficiente di soluzioni parziali considerate in precedenza (con complessità spaziale polinomiale) tramite l'uso di una soglia di backtrack, una tolleranza sul costo della soluzione che previene il backtracking. La terza idea è fornire un meccanismo di individuazione della terminazione costruito internamente all'algoritmo; gli agenti terminano nel momento in cui trovano una soluzione completa il cui costo è inferiore alla loro soglia di backtrack attuale. Questo meccanismo elimina la necessità di aggiungere un meccanismo esterno per individuare la terminazione, che richiederebbe lo scambio di ulteriori messaggi.
In Adopt, gli agenti sono prioritizzati in una struttura ad albero in cui ogni agente ha un singolo padre e molteplici figli. Assumendo che i vincoli siano binari e monodirezionali (ovvero imposti da un agente ad un altro), l'albero viene costruito dal grafo dei vincoli facendo sì che vi siano vincoli solo tra un agente e un proprio antenato o discendente. È ovviamente possibile costruire diversi alberi per un dato grafo dei vincoli.
Attraverso questo ordinamento prioritario, Adopt effettua una ricerca con backtrack distribuita usando una strategia di ricerca best-first (ricerca in profondità) opportunistica, ovvero in cui ogni agente sceglie per la propria variabile il miglior valore basandosi sulle informazioni correnti. Queste informazioni si traducono in due limiti (superiore e inferiore) sulla qualità delle soluzioni, che vengono elaborati e comunicati tra i vari agenti. Per un problema di minimizzazione, questa strategia di ricerca implica che un agente scelga sempre il valore con il minimo limite inferiore (nella massimizzazione sceglierà il valore con il massimo limite superiore). Questi limiti vengono inizializzati basandosi esclusivamente sui costi locali di un agente, e iterativamente perfezionati quando l'agente riceve nuove informazioni sui costi dai vicini. Questa strategia consente agli agenti di abbandonare soluzioni parziali prima che si possa dimostrare che queste siano definitivamente non ottime, potrebbe quindi rivelarsi necessario riconsiderare soluzioni considerate in precedenza.
I messaggi che vengono scambiati tra gli agenti sono:

- messaggi *value* inviati da un nodo x_i ai propri figli attraverso un ramo che identifica un vincolo e contenenti l'assegnamento attuale scelto da x_i.

- messaggi *cost* inviati da un nodo x_i al padre e contenenti il costo computato in x_i sommato ai costi che x_i ha ricevuto dai propri figli.

- messaggi *threshold* usati per ridurre ridondanze nella ricerca e inviati solo da un nodo ai propri figli.

Ogni agente mantiene nota degli assegnamenti di tutti i vicini con priorità maggiore (i suoi antenati nell'albero); per fare questo possiede un contesto, ovvero una soluzione parziale contenente gli assegnamenti noti all'agente. Due contesti sono compatibili se non discordano per alcun assegnamento. Il contesto di un determinato agente viene iterativamente aggiornato ogni qualvolta viene ricevuto un messaggio value da un antenato o quando l'agente stesso decide di modificare il proprio valore in seguito alla ricezione di messaggi cost dai figli. Un messaggio cost inviato da un nodo figlio al padre contiene informazioni sul contesto del figlio e sui limiti superiore e inferiore; confrontando questi valori con i propri, il padre è in grado di capire quale azione compiere per ottimizzare ulteriormente la soluzione.
Adopt è un algoritmo corretto e completo, in quanto trova sempre una soluzione che ottimizza la funzione di utilità globale, mostrando però un overhead per il coordinamento esponenziale con il numero di variabili nel sistema.

2.3.3 Max-Sum

L'algoritmo Max-Sum [23] è un algoritmo di ottimizzazione basato sullo scambio di messaggi che appartiene al framework GDL. Gli algoritmi GDL sfruttano la fattorizzabilità di molti problemi di ottimizzazione, risolvendoli in maniera efficiente. In particolare, questi problemi sono caratterizzati come problemi di ottimizzazione la cui algebra di valutazione della funzione di vincolo globale è un *semi-anello commutativo*. Tutti i problemi DCOP standard possiedono questa caratteristica.
La più generica caratterizzazione di un DCOP coinvolge M agenti, ciascuno dei quali controlla un'unica variabile discreta x_j, $j \in [1, M]$. I vincoli tra gli agenti sono rappresentati da funzioni $U_i(\mathbf{x}_i)$, $i \in [1, N]$ su queste variabili. Lo scope $\mathbf{x}_i \subseteq \mathbf{x}$ della funzione di vincolo U_i contiene le variabili degli agenti presso i quali è definito il vincolo. Lo scopo del problema di coordinamento è quindi la scelta di assegnamenti alle variabili che massimizzino la somma delle funzioni vincolo:

$$U(\mathbf{x}) = \sum_{i=1}^{N} U_i(\mathbf{x}_i)$$

Per poter usare l'algoritmo Max-Sum il problema viene codificato come uno speciale grafo bipartito chiamato grafo fattorizzato, nel quale i vertici rappresentano variabili e funzioni, e gli archi le dipendenze tra esse.
Max-Sum definisce due tipi di messaggi che vengono scambiati tra variabili e funzioni:

Da variabile x_j a funzione U_i :

$$q_{j\to i}(x_j) = \sum_{k\in M(j)\setminus i} r_{k\to j}(x_j)$$

dove $M(j)$ rappresenta l'insieme degli indici delle funzioni connesse alla variabile x_j (le funzioni in cui x_j compare come argomento).

Da funzione U_i a variabile x_j :

$$r_{i\to j}(x_j) = \max_{\mathbf{x}_i \setminus\ x_j} \left(U_i(\mathbf{x}_i) + \sum_{k\in N(i)\setminus j} q_{k\to i}(x_k) \right)$$

dove $N(i)$ rappresenta l'insieme degli indici delle variabili connesse alla funzione U_i. Sia $q_{j\to i}(x_j)$ che $r_{i\to j}(x_j)$ sono funzioni scalari della variabile x_j. Quando il grafo fattorizzato è aciclico, questi messaggi rappresentano la massima utilità aggregata possibile tra i rispettivi componenti del grafo formato rimuovendo la dipendenza tra U_i e x_j, per ogni valore $d \in D_{x_j}$ nel dominio della variabile x_j. Per ogni variabile x_j, viene computata una funzione marginale $z_j(x_j)$:

$$z_j(x_j) = \sum_{i\in M(j)} r_{i\to j}(x_j) = \arg\max_{\mathbf{x}\setminus x_j} \sum_{i=1}^{N} U_i(\mathbf{x}_i)$$

dalla quale si ricava l'assegnamento ottimale di x_j come:

$$a_j = \arg\max_{x_j} z_j(x_j)$$

2.3.4 Bounded Max-Sum

Quando il grafo dei vincoli di un DCOP è ciclico, l'applicazione diretta dell'algoritmo Max-Sum non garantisce la convergenza verso una soluzione. Potando alcuni degli archi è possibile ottenere un sottografo aciclico del grafo originale; questo sottografo corrisponde ad un'approssimazione del problema

originale. L'algoritmo Bounded Max-Sum [23] prevede una fase preliminare, denominata Bounding, nella quale vengono rimossi degli archi dal grafo dei vincoli originale fino alla completa rimozione delle ciclicità; in questa fase vengono calcolati dei pesi per gli archi del grafo che permettono di elaborare un fattore di approssimazione di quanto il problema originale è stato approssimato tramite la potatura degli archi. L'applicazione di Max-Sum a questo sottografo aciclico permette di computare una soluzione approssimata la cui distanza dalla soluzione ottimale del problema originale è misurabile attraverso i pesi calcolati durante la fase di potatura.
Nello specifico, l'obiettivo è la computazione di un assegnamento alle variabili $\tilde{\mathbf{a}}$ nel grafo fattorizzato aciclico tale che:

$$V^* = \sum_{i=1}^{N} U_i(\mathbf{a}_i^*) \leq \rho \sum_{i=1}^{N} U_i(\tilde{\mathbf{a}}_i) = \rho \tilde{V}$$

dove $\mathbf{a}^*$ è la soluzione ottima del grafo fattorizzato ciclico, e ρ è l'indice di approssimazione. Per assicurare che l'indice di approssimazione sia il più piccolo possibile, l'algoritmo pota quegli archi del grafo dei vincoli originale che hanno il minimo impatto sulla qualità della soluzione. L'impatto di un arco tra x_j e U_i è definito come il suo peso w_{ij}, e computato come:

$$w_{ij} = \max_{\mathbf{x}_i \setminus x_j} \left[\max_{x_j} U_i(\mathbf{x}_i) - \min_{x_j} U_i(\mathbf{x}_i) \right]$$

Una volta computati tutti i pesi, viene usato l'algoritmo GHS [10] per computare uno spanning tree massimale del grafo fattorizzato in stile decentralizzato. Il nuovo grafo fattorizzato viene usato nella seconda fase, nella quale viene usato l'algoritmo Max-Sum per computare $\tilde{\mathbf{a}}$, ovvero l'assegnamento alle variabili ottimale al problema modificato:

$$\tilde{V}_m = \sum_i \min_{\mathbf{x}_i^c} U_i(\tilde{\mathbf{a}}_i)$$

dove $\mathbf{x}_i^c$ è l'insieme delle variabili eliminate dallo scope della funzione U_i, corrispondenti agli archi potati dal grafo fattorizzato. L'indice di approssimazione è dato da:

$$\rho = 1 + \left(\tilde{V}_m + W - \tilde{V} \right) / \tilde{V}$$

dove W è la somma dei pesi degli archi potati. Quindi, un limite superiore sulla soluzione ottima può essere computato come segue:

$$\tilde{V}_m + W \geq V^*$$

2.3.5 Criteri di ottimalità locale in DCOP

Nel formalismo DCOP, in presenza di reti di vincoli di grandi dimensioni, è spesso complicato o impossibile trovare l'ottimo globale usando un algoritmo completo; questa condizione porta quindi all'applicazione di algoritmi incompleti, che sono solo in grado di trovare soluzioni localmente ottime senza garanzie sulla qualità delle soluzioni.
Uno studio su *k-size-optimality* [21] fornisce garanzie teoriche sulla qualità delle soluzioni localmente ottimali attraverso alcune proprietà. Il concetto di *k-size-optimality* si appoggia su gruppi locali composti da k agenti, dove k è il minimo numero di variabili che devono cambiare il proprio valore per migliorare la qualità della soluzione globale.
In un problema di massimizzazione, sia $\mathcal{A}$ un assegnamento DCOP, con utilità $R(\mathcal{A})$ e $\mathcal{A}(i)$ il valore della variabile v_i in $\mathcal{A}$.

Definizione 4. *Il gruppo deviante $D(\mathcal{A}, \mathcal{A}')$ è definito come l'insieme dei nodi con differente assegnamento in $\mathcal{A}$ e $\mathcal{A}'$, $D(\mathcal{A}, \mathcal{A}') = \{v_i | \mathcal{A}(i) \neq \mathcal{A}'(i)\}$.*

Definizione 5. *Un assegnamento DCOP $\mathcal{A}$ è k-size-optimal se $R(\mathcal{A}) \geq R(\mathcal{A}')$ per qualsiasi $\mathcal{A}'$ tale che $|D(\mathcal{A}, \mathcal{A}')| \leq k$.*

Questo concetto presenta tuttavia alcuni difetti. In primo luogo, il limite inferiore di qualità per la *k-size-optimality* è inversamente proporzionale alla densità del grafo dei vincoli, e per valori elevati del numero di variabili questo limite diventa inaccettabilmente basso. La *k-size-optimality* considera qualunque gruppo di variabili, alcuni dei quali potrebbero contenere nodi distanti tra loro. In questi gruppi, il coordinamento per ottenere un'azione congiunta può essere costoso, poiché richiede la trasmissione di messaggi tra nodi distanti dal nodo centrale. Infine, il numero di possibili gruppi di dimensione k cresce combinatoriamente con il valore di k, poiché ogni combinazione di k nodi connessi può formare un gruppo; la complessità per enumerare ed ottimizzare ciascun gruppo rende difficile la progettazione di algoritmi per *k-size-optimality* con $k \geq 3$.
Un secondo criterio di ottimalità è la *t-distance-optimality* [29], che considera un numero fisso di gruppi di ottimizzazione locale definiti dalla distanza sul grafo dei vincoli.
Per una coppia di variabili u e v, sia $T(u, v)$ il cammino minimo tra le due variabili nel grafo dei vincoli.

Definizione 6. *Si denota con $\Omega_t(v)$ il gruppo delle variabili raggiungibili da v con t passi, $\Omega_t(v) = \{u | T(u, v) \leq t\}$*

Definizione 7. *Un assegnamento DCOP $\mathcal{A}$ è t-distance-optimal se $R(\mathcal{A}) \geq R(\mathcal{A}')$ per ogni $\mathcal{A}'$, dove $D(\mathcal{A}, \mathcal{A}') \subseteq \Omega_t(v)$ per qualche $v \in V$.*

Il numero massimo di gruppi t-distanti centrati su n variabili è n. Le soluzioni *t-distance-ottimali* possiedono dei limiti di qualità formali sia in conoscenza della struttura del grafo sia nel caso in cui essa non sia nota; questi limiti offrono maggiori garanzie rispetto a soluzioni *k-size-ottimali* comparabili.
La differenza chiave tra *k-size-optimality* e *t-distance-optimality* è che nella prima si può avere un alto numero di gruppi di dimensione fissa, soprattutto per grafi di vincoli densi, mentre nella seconda il numero di gruppi è fisso, e la dimensione di ogni gruppo varia e può diventare grande. Una soluzione *t-distance-optimal* garantisce una $(2t+1)$-*size-optimality*, in quanto ogni gruppo di dimensione $2t+1$ è sicuramente contenuto in qualche gruppo di distanza t. Inoltre la *t-distance-optimality* cattura in modo naturale la località nel grafo, fattore che può migliorare l'efficienza degli algoritmi di ricerca locale in particolare in ambienti distribuiti dove il ritardo di comunicazione è il costo dominante; inoltre potrebbe ridurre la perdita di privacy poiché le informazioni private di un agente possono essere ottenute solo da agenti entro una distanza fissa.

2.4 Estensioni del framework DCOP

2.4.1 Distributed Hierarchical Constraint Satisfaction

Nei sistemi multiagente spesso diversi agenti cercano una combinazione consistente di azioni sottostanti a determinati vincoli; problemi distribuiti come interpretazione di dati, allocazione di risorse, scheduling e altri sono descritti in modo naturale come problemi di soddisfacimento di vincoli distribuiti (DCSP), in cui variabili e vincoli sono distribuiti tra i diversi agenti. Diversi algoritmi risolutivi per la ricerca di una soluzione di un DCSP sono in grado di fornire la soluzione, se ne esiste una, ma nessuno può fornire ulteriori informazioni se invece non esistono soluzioni. Ad esempio, un algoritmo completo quale l'Asynchronous Backtracking [27] descritto in 2.2.4 riporta solamente il fatto che non vi è alcuna soluzione, mentre un algoritmo incompleto come il Distributed Breakout Algorithm [28] non termina. In molti problemi applicativi può essere richiesta una soluzione parziale che soddisfi alcuni criteri, per esempio una soluzione parziale che soddisfi il maggior numero possibile di vincoli.
Un Distributed Partial CSP [12] può essere adottato come modello generale per la manipolazione di un over-constrained DCSP (eccessivamente vincolato, presumibilmente privo di soluzione). Intuitivamente, in un Distributed Partial CSP gli agenti cercano di ottenere un DCSP risolvibile e la sua so-

luzione tramite il rilassamento di un DCSP over-constrained. Attraverso il rilassamento di un DCSP, vengono introdotte diverse soluzioni parziali.
Un Distributed Maximal CSP [12] può essere considerato come una sottoclasse di un Distributed Partial CSP, e in questo caso gli agenti cercano valori da assegnare alle variabili che minimizzino il numero di vincoli violati dagli agenti.
Un altro approccio è introdotto come Distributed Hierarchical CSP [13], ovvero il problema in cui gli agenti cercano di assegnare alle variabili valori che minimizzano il massimo valore di importanza dei vincoli violati. Questa soluzione parziale è ritenuta importante in quanto vi sono diversi problemi in MAS dove ogni agente desidera ottenere una soluzione (parziale) che non violi i propri vincoli importanti (ad esempio la programmazione oraria di un calendario in cui molteplici agenti sono interessati). In un Distributed Hierarchical CSP ad ogni vincolo è associato un intero positivo denominato *valore di importanza* che rappresenta l'importanza del vincolo; un vincolo cui è associato un valore maggiore è considerato più importante. Questa assunzione è ritenuta ragionevole poiché ogni vincolo nel mondo reale ha una qualche semantica che ne definisce un'importanza. Gli agenti quindi cercano un assegnamento di valori alle variabili che minimizzi il massimo valore di importanza dei vincoli violati; questo tipo di soluzione parziale può essere utile quale ragionevole compromesso quando ogni agente cerca di soddisfare più vincoli (propri) possibile, con valore di importanza più grande possibile. Formalmente, risolvere un Distributed Hierarchical CSP corrisponde a trovare una soluzione ottimale ad un Distributed Partial CSP specializzato con i seguenti concetti:

- Ogni agente conosce solo i vincoli che coinvolgono le proprie variabili. Si definisce P_i l'insieme delle variabili controllate dall'agente i e dei vincoli ad esse collegati. P_i è un CSP sottoproblema del problema originale, corrispondente al sottoproblema visto dall'agente i.

- Per ogni agente i è definito un insieme $PS_i = \{P_i^0, P_i^1, \dots\}$ composto da diversi sottoproblemi del problema originale. P_i^α è un CSP ottenuto dal sottoproblema P_i rimuovendo ogni vincolo con valore di importanza minore o uguale ad α.

- Per ogni agente i, la distanza d_i tra P_i e P_i^α è definita come α.

- La distanza globale del problema è misurata come $\max_{i \in A} d_i$, dove A è l'insieme di tutti gli agenti.

Un Distributed Hierarchical CSP può non avere soluzione per una certa gamma di distanza globale (0 o anche superiore), ma può avere una soluzione per una distanza globale più alta, poiché un aumento della distanza globale fino a un valore W corrisponde alla rimozione dal problema originale dei vincoli con valore di importanza inferiore a W. La distanza globale ottima è il minimo valore per il quale il problema ha una soluzione.
L'algoritmo sviluppato per la risoluzione di un Distributed Hierarchical CSP utilizza gli algoritmi DCSP precedenti, con l'idea di suddividere il processo di risoluzione in due parti: la *ricerca dello spazio dei valori* e la *ricerca dello spazio del problema*. Nella prima parte gli agenti cercano di trovare una soluzione a qualche DCSP, sfruttando algoritmi esistenti come l'asynchronous backtracking, mentre nella seconda gli agenti cercano di ricavare un DCSP risolvibile dai propri spazi di problema distribuiti. L'algoritmo in sè è una combinazione delle due parti.

2.4.2 Multiply-Constrained DCOP

Un DCOP standard è un problema in cui si cerca di ottimizzare un unico obiettivo per tutti gli agenti. Esistono diversi domini in cui gli agenti devono soddisfare ulteriori vincoli riguardanti le risorse consumate localmente, come budget, risorse fisiche, energetiche. La presenza di questi vincoli richiede un algoritmo DCOP che ottimizzi un obiettivo globale assicurando che i limiti sulle risorse non vengano superati. In alcuni casi gli agenti dovranno mantenere privati i vincoli sulle risorse (budget di viaggio nell'organizzazione distribuita di incontri), mentre in altri potrebbero non essere privati (limiti di lavoro straordinario nell'allocazione distribuita di staff per lo sviluppo di software). Multiply-Constrained DCOP [5] è un framework ideato appositamente per questi casi.
Un Multiply-Constrained DCOP contiene una nuova funzione di costo g_i definita su un sottoinsieme dei collegamenti della variabile x_i e con un budget G_i che non dev'essere superato da g_i. La funzione g-costo e il budget insieme costituiscono un g-vincolo. Poiché le funzioni g-costo non possono essere unite alle funzioni f-costo, relative ai vincoli classici DCOP, questo modello è multi-vincolato (multiply-constrained), con due classi di vincoli distinte e la cui ottimizzazione (o soddisfazione del limite di budget) avviene separatamente. I g-vincoli possono essere privati o condivisi.
Questo modello evidenzia tre caratteristiche: vincoli n-ari (costi g), g-vincoli privati e lo sfruttamento dell'interazione tra f-vincoli e g-vincoli per la potatura dello spazio di ricerca. Nessuno dei classici algoritmi DCOP è in grado di gestire e risolvere tutte queste problematiche.

L'algoritmo proposto in [5] per la risoluzione di un Multiply-Constrained DCOP, Multiply-Constrained Adopt (MCA), si basa su una strategia di ricerca con reciproco intervento tra agenti: un agente può immediatamente intervenire nella ricerca dell'ottimo globale se i propri vincoli sulle risorse locali sono violati, mentre una strategia opportunista per la ricerca della soluzione globalmente ottima ignora l'analisi della soddisfacibilità dei vincoli sulle risorse nell'elaborazione delle soluzioni parziali.
L'algoritmo si basa su tre idee chiave. La prima è la trasformazione del grafo dei vincoli, motivata dall'obiettivo di sfruttare gli algoritmi DCOP già esistenti. Viene introdotta un'innovazione consistente nell'impiego di variabili virtuali, indicanti la violazione dei vincoli con comunicazioni asincrone di elevati costi negativi che tagliano così rami di ricerca preventivamente, per applicare i vincoli sulle risorse agli algoritmi DCOP standard. Con la ricerca dinamicamente-vincolante gli agenti rivelano ai vicini un limite superiore per ogni vincolo sulle risorse non privato, così che durante l'ottimizzazione della funzione obiettivo globale i vicini selezioneranno solo i valori che rispettano i limiti. La terza idea è l'aciclicità locale, che migliora l'efficienza della ricerca nei grafi DCOP localmente aciclici consentendo di restringere i limiti sulle risorse condivisi senza sacrificare la correttezza dell'algoritmo. In particolare questo avviene mediante l'uso di T-nodi, variabili la cui aciclicità locale consente l'uso dinamico di limiti esatti e non richiede variabili virtuali.

2.4.3 Stakeholder Search

Lo studio in [8] considera degli scenari nei quali sono coinvolte diverse organizzazioni nel ruolo di stakeholder, e dove l'organizzazione leader e gli altri individui hanno interessi che possono essere tra loro complementari, conflittuali o scorrelati. Ad esempio nella realizzazione di un aeroporto si contrappongono l'interesse del costruttore a minimizzare i costi e il desiderio delle associazioni dei residenti locali di minimizzare i livelli di rumore. Un approccio a questo tipo di problemi è la raccolta di tutti gli obiettivi e l'elaborazione centralizzata di un piano che li soddisfi o che vi si avvicini il più possibile, mediante una forma di ottimizzazione obiettivo. Tuttavia, per grandi e complessi progetti la rivelazione e prioritizzazione degli obiettivi potrebbe non essere possibile principalmente perché gli stakeholders tendono a mantenere privati i propri obiettivi al fine di proteggere i propri segreti o di mantenere vantaggi strategici.
L'approccio stakeholder prevede che un'organizzazione leader deleghi l'analisi del problema agli stakeholder, permettendo loro di proporre soluzioni ad essi gradite; quindi consente il perfezionamento di queste soluzioni tramite

una serie di interazioni, ed infine sceglie una tra le soluzioni proposte che meglio si avvicini all'obiettivo del leader. In questo approccio gli stakeholder hanno l'opportunità di influenzare l'esito del processo poiché la soluzione sarà una di quelle che loro hanno proposto, inoltre possono mantenere privati i propri obiettivi e metodi. L'organizzazione leader potrebbe non trovare una soluzione che soddisfa il proprio obiettivo, ma avendo l'ultima decisione sulla scelta dovrebbe poter trovare una buona soluzione.
Nell'ambito dei sistemi multiagente, si possono sfruttare reti di calcolatori e concetti di ricerca cooperativa per risolvere questo tipo di problemi. Un gruppo di agenti rappresentanti l'organizzazione leader e gli stakeholder lavora cooperativamente per trovare buone soluzioni; molti concetti di ricerca cooperativa assumono una cooperazione totale tra gli agenti con un flusso di informazioni altruistico e agenti che operano al solo fine di trovare la soluzione migliore al problema. Nel caso di grandi progetti che coinvolgono molti stakeholder, questa assunzione non è realistica, poiché oltre all'interesse globale curato dal leader, ogni singolo stakeholder ha i propri interessi individuali. Tradizionalmente questo genere di problemi è stato affrontato con approcci come meccanismi di mercato o aste, che mirano principalmente a garantire una minima qualità dei risultati. Sono di recente concezione approcci che intrecciano ricerca cooperativa con obiettivi locali degli agenti, per lo più relativamente a problemi di soddisfacimento di vincoli distribuiti risolti dividendoli in più sottoproblemi e mantenendo i vincoli il più privati possibile.
Stakeholder search è un framework per ricerca cooperativa con un obiettivo globale e diversi obiettivi locali agli agenti, che usa il paradigma generale di cooperazione per il miglioramento dell'approccio competitivo tra agenti. Gli agenti stakeholder possono implementare strategie che vanno dal completamente cooperativo, in cui ignorano l'obiettivo individuale e cercano soluzioni che soddisfano solo l'obiettivo globale, strategie competitive dove l'obiettivo globale viene ignorato, e strategie che si pongono tra questi estremi. Il framework identifica le decisioni chiave, ovvero derivanti dalle scelte strategiche degli agenti, che devono essere prese per impostare un'istanza della ricerca, sia al livello di ricerca che al livello degli stakeholder, per raggiungere il risultato desiderato. Si assume che la selezione finale della soluzione sia basata sull'obiettivo globale, e che o il tempo disponibile per la ricerca della soluzione è limitato o il problema è così vasto che non è possibile trovare una soluzione globalmente ottima in tempi pratici.

Capitolo 3

Impostazione del problema di ricerca

In questo capitolo viene introdotto l'obiettivo di questo lavoro di tesi, che consiste nella ricerca di formalismi attraverso cui modellare sistemi idrici composti da molteplici agenti, e di algoritmi in grado di risolverli fornendo agli agenti informazioni per una gestione globalmente efficiente.

3.1 Modellazione MAS di sistemi idrici

Un sistema idrico può essere modellato come un sistema multiagente (MAS) [3, 4, 14, 20, 26] composto da diversi agenti che usufruiscono delle risorse del sistema (tipicamente flussi d'acqua). Questi sistemi sono caratterizzati da processi decisionali distribuiti al livello di ciascun agente, e da un meccanismo di coordinamento atto a organizzare le interazioni tra i vari processi decisionali individuali ad un livello di sistema globale. La gestione sostenibile dei sistemi idrici porta a considerare gli stessi come abbinamenti di sistemi naturali-umani, al fine di focalizzare l'attenzione sia sui bisogni umani che sui requisiti degli ecosistemi. Le attività umane sono distribuite negli ecosistemi e strettamente connesse alla rete dei flussi idrici; gli stakeholder condividono terreni comuni e risorse di bacini idrici. Le problematiche specifiche della gestione di questi sistemi includono il bilanciamento delle necessità degli agenti umani che hanno accesso ai flussi con le esigenze ambientali dei flussi stessi, la regolazione delle quantità e qualità d'acqua e dei regimi di flusso, e le tecnologie da adottare e inserire nel contesto del sistema.
I modelli di gestione convenzionali utilizzano un approccio centralizzato per esplorare la soluzione ottimale a livello di sistema globale, assumendo che esista un processo di gestione top-down in cui tutti gli stakeholder obbedi-

scono ad un supervisore, che vi sia uno scambio completo di informazioni tra stakeholder appartenenti a diverse posizioni e settori di produzione, e che si ottenga una perfetta efficienza economica, ovvero che il benessere marginale dell'uso delle risorse idriche sia identico per tutti gli agenti. Queste assunzioni sono tipicamente inadatte in situazioni reali; approcci di modellazione decentralizzata possono prescindere da alcune di queste assunzioni. Tuttavia, in campo ambientale, gli agenti vengono usati solo come simulatori, per valutare le proprietà che emergono a livello globale a causa delle interazioni a basso livello tra gli agenti, e non come ottimizzatori [4].

3.2 Un caso di studio

In [26] viene proposto un metodo di ottimizzazione decentralizzata che rappresenta un sistema idrico come sistema multiagente, con lo scopo di fornire un supporto bottom-up pilotato dai vari stakeholder per la gestione del sistema. In questo contesto, un sistema idrico è definito come un sistema autonomamente organizzato caratterizzato da processi decisionali disaggregati ma interagenti al livello dei singoli agenti. Un sistema viene qui modellato attraverso una formulazione matematica in cui ogni agente i possiede e controlla una variabile decisionale x_i. Per ogni agente è definita una funzione obiettivo, che descrive l'utilità locale dell'agente i in funzione della variabile decisionale locale; ogni agente persegue il proprio benessere economico, e pertanto cerca di massimizzare la propria utilità. Nel sistema sono definiti inoltre una serie di vincoli che si distinguono tra: vincoli fisici, che descrivono le condizioni in cui il sistema o una sua porzione possono fisicamente trovarsi distinguendole da quelle fisicamente inamissibili, e vincoli normativi, che rappresentano delle norme istituite per una regolazione e un buon funzionamento del sistema e la cui violazione comporta una penalità da pagare. In questo studio viene adottato un metodo decentralizzato basato su penalità, in cui ogni agente possiede una propria struttura decisionale che cattura solamente le dinamiche locali e i vincoli associati a queste dinamiche. L'ottimizzazione avviene attraverso uno schema di negoziazione, in cui gli agenti ottimizzano il proprio obiettivo seguendo un ordine di priorità preimpostato, e inviano la propria soluzione agli agenti vicini, direttamente collegati da un vincolo, con cui sono in grado di interagire. Tutti gli agenti effettuano queste scelte simultaneamente, e le diverse soluzioni sono valutate da un processore centrale, che rende disponibili a tutti gli agenti informazioni quali costi di sistema e costi di violazione dei vincoli per il prossimo turno di negoziazione. L'approccio usato si divide in due passi: in primo luogo si cerca una soluzione basata sulle scelte individuali di tutti gli agenti, consentendo la violazione

di alcuni vincoli definiti nel modello, e quindi si cerca di ridurre la violazione di questi vincoli a livello di sistema. Questo metodo segue una procedura di risoluzione bottom-up in cui ogni agente effettua delle decisioni interagendo con altri agenti, e infine un coordinatore risolve i conflitti bilanciando le varie decisioni.
Il difetto principale di questo approccio tuttavia è l'incapacità di gestire in modo ottimale i vincoli fisici e normativi, in quanto l'algoritmo di ottimizzazione implementato non è realmente multiagente, e non sfrutta appieno le potenzialità dei MAS quali garanzie di ottimalità ed efficienza.

3.2.1 Modello di riferimento

Il caso di studio trattato in [26] propone il modello MAS di un ipotetico sistema idrico, illustrato in Figura 3.1a), come riferimento per l'analisi teorica dell'approccio adottato e sperimentale dell'algoritmo implementato. Si tratta di un sistema fluviale con struttura a Y, costituito da un flusso principale e un affluente. Una città, collocata nei pressi della parte superiore del flusso principale, preleva parte del flusso per uso municipale e industriale. Il resto dell'acqua scorre a valle verso un serbatoio per la generazione di energia idroelettrica. Due fattorie deviano acqua per l'irrigazione dal flusso principale e dall'affluente. Le portate dei fiumi a valle delle deviazioni per l'irrigazione sono entrambe identificate come habitat ittici.
Nel modello MAS del sistema, schematizzato in Figura 3.1b), vi sono tre agenti collocati esternamente al flusso che decidono quanta acqua prelevare deviandola dal flusso stesso (off-stream): OHA_1 per la città, OHA_2 e OHA_3 per le fattorie. La diga posizionata sul flusso principale superiore è definita come agente umano interno al flusso (in-stream): IHA_1. Gli habitat ittici collocati presso l'affluente e il flusso principale inferiore sono definiti come agenti ecosistema, EA_1 e EA_2 rispettivamente.
Le funzioni obiettivo dei singoli agenti hanno diverse forme; in questo studio vengono adottate delle funzioni quadratiche concave (parabole) per tutte le funzioni obiettivo per testare l'algoritmo in maniera semplice senza perdere le caratteristiche nonlineari tipiche delle funzioni obiettivo reali.

Nel contesto di modello MAS, diversi agenti prendono diverse decisioni nel rispetto dei propri obiettivi. Gli agenti umani esterni ai flussi (OHA_1, OHA_2 e OHA_3) decidono quanta acqua deviare per soddisfare le proprie esigenze (x_1, x_4 e x_6 rispettivamente); l'agente umano collocato all'interno del flusso (IHA_1) decide la quantità d'acqua da rilasciare (x_2) per la generazione di energia idroelettrica. Tuttavia, i modelli di tutti gli agenti sono interconnessi da un'istituzione che gestisce terreni e risorse idriche e da relazioni

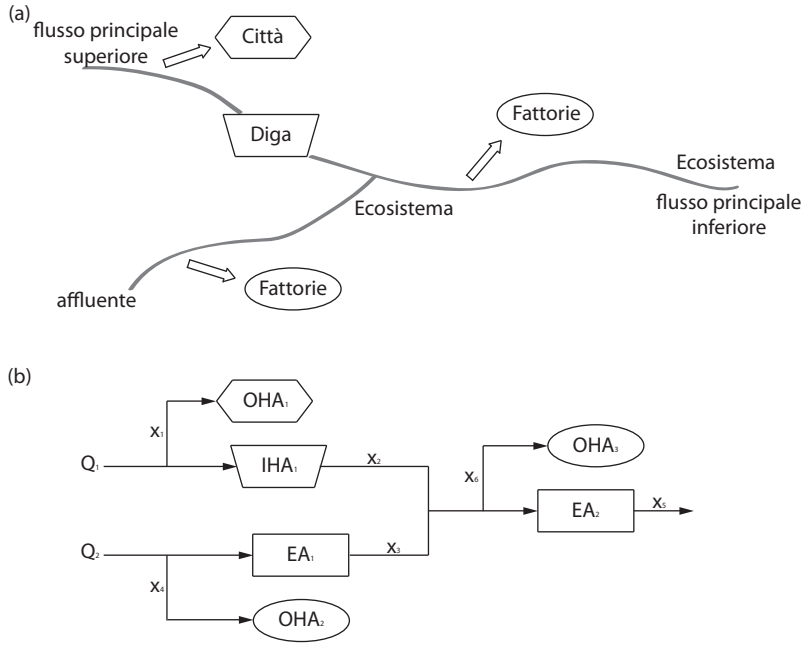

Figura 3.1: Schema di un ipotetico sistema idrico e del suo modello MAS

per la condivisione dell'acqua dei vari flussi. Quando un agente persegue il proprio ottimo locale, potrebbe violare dei vincoli con il risultato di una situazione peggiore per gli altri agenti. Nella realtà questo avviene tipicamente quando le utenze del flusso a monte abusano dei flussi e gli utenti del flusso a valle subiscono di conseguenza scarsità d'acqua. L'analisi di questo modello si basa su tre assunzioni. Gli agenti possono violare qualunque vincolo, anche quelli fisici: la violazione dei vincoli fisici è impossibile nella realtà, tuttavia questa assunzione permette di valutare la quantità di risorse aggiuntive che occorrerebbero per poter soddisfare i requisiti di tutti gli agenti. La seconda assunzione prevede che le informazioni locali del flusso a valle non scorrano verso il flusso a monte, questo implica che quando un agente collocato nel flusso a monte prende una decisione, raramente considera la situazione degli agenti a valle a meno che non vengano fornite delle informazioni globali sui vincoli relativi al flusso a valle. Si assume infine una situazione statica: lo studio si focalizza su un unico passo temporale, e pertanto non considera le variazioni nel tempo di disponibilità e richiesta delle varie risorse.

Agente	Sottoproblema di ottimizzazione	Notazione
OHA_1	$\max_{x_1} f_1(x_1) = a_1x_1^2 + b_1x_1 + c_1$ soggetto a $\begin{cases}\alpha_1 - x_1 \leq 0 \\ \alpha_2 - Q_1 + x_1 \leq 0\end{cases}$	x_1: prelievo d'acqua per uso municipale e industriale Q_1: afflusso mensile in ingresso nel flusso principale α_1: fabbisogno idrico minimo per la città α_2: afflusso minimo in ingresso alla diga
OHA_2	$\max_{x_4} f_4(x_4) = a_4x_4^2 + b_4x_4 + c_4$ soggetto a $\begin{cases}\alpha_3 - x_4 \leq 0 \\ \alpha_4 - Q_2 + x_4 \leq 0\end{cases}$	x_4: prelievo d'acqua per irrigazione Q_2: afflusso mensile in ingresso nell'affluente α_3: fabbisogno idrico minimo per la fattoria α_4: afflusso minimo in uscita per l'affluente
OHA_3	$\max_{x_6} f_6(x_6) = a_6x_6^2 + b_6x_6 + c_6$ soggetto a $\begin{cases}\alpha_5 - x_6 \leq 0 \\ \alpha_6 - x_2 - x_3 + x_6 \leq 0\end{cases}$	x_6: prelievo d'acqua per irrigazione α_5: fabbisogno idrico minimo per la fattoria α_6: afflusso minimo in uscita per il flusso principale
IHA_1	$\max_{x_2} f_2(x_2) = a_2x_2^2 + b_2x_2 + c_2$ soggetto a $x_2 - S - Q_1 + x_1 \leq 0$	x_2: flusso rilasciato dalla diga S: capacità della diga
EA_1	$\max_{x_3} f_3(x_3) = a_3x_3^2 + b_3x_3 + c_3$ soggetto a $\begin{cases}x_3 = Q_2 - x_4 \\ \alpha_4 - x_3 \leq 0\end{cases}$	x_3: flusso che attraversa l'affluente
EA_2	$\max_{x_5} f_5(x_5) = a_5x_5^2 + b_5x_5 + c_5$ soggetto a $\begin{cases}x_5 = x_2 + x_3 - x_6 \\ \alpha_6 - x_5 \leq 0\end{cases}$	x_5: flusso principale nel ramo inferiore

Tabella 3.1: Formulazione del modello MAS in [26]

3.3 Approccio distribuito

Il modello di sistema idrico descritto in [26] viene riconsiderato nello studio [11], che si propone di affrontare il medesimo problema con un approccio distribuito, con il fine di risolvere i problemi di gestione dei vincoli emersi nello studio originale. L'obiettivo di questo secondo studio è lo sviluppo di un metodo di ottimizzazione multiagente per la progettazione di un sistema di gestione di sistemi idrici distribuito dal punto di vista dell'autorità in carica, che coordina molteplici decisori indipendenti. Partendo da una configurazione inefficiente in cui gli agenti sono totalmente scoordinati, viene sviluppata una strategia che impone un insieme di vincoli normativi sulle decisioni di ogni agente, per pilotare la soluzione verso una situazione più efficiente. La massima efficienza può essere ottenuta con un approccio di gestione centralizzato, in cui gli agenti sono completamente vincolati e agiscono in accordo con le decisioni imposte dall'autorità centrale. Il problema viene quindi formulato come problema di vincoli distribuiti, adottando per la risoluzione i framework DCSP e DCOP, che offrono garanzie sulla qualità delle soluzioni

globali operando efficientemente attraverso comunicazioni locali.
La soluzione ideale centralizzata viene qui confrontata con tre alternative distribuite:

- Una soluzione indipendente in cui ogni decisore agisce considerando esclusivamente il suo obiettivo.
- Una soluzione DCSP in cui gli agenti cercano di massimizzare le proprie funzioni obiettivo, in cui gli assegnamenti di valori alle variabili decisionali devono essere ammissibili, ovvero devono soddisfare tutti i vincoli imposti nel problema.
- Una soluzione DCOP in cui le decisioni degli agenti, sempre mirate a ottimizzare le funzioni obiettivo locali, sono qui influenzate anche dai vincoli normativi, che potrebbero essere violati, con la garanzia che questa soluzione minimizzi la somma delle violazioni.

I risultati mostrano come, tra gli estremi di una gestione completamente centralizzata e completamente indipendente, esiste la possibilità di progettare soluzioni distribuite intermedie più efficienti di quella indipendente e più realistiche e politicamente fattibili rispetto alla soluzione centralizzata. Gli approcci DCSP e DCOP permettono all'autorità di imporre vincoli normativi sulle decisioni degli agenti per indirizzare la soluzione verso una gestione più efficiente. La soluzione centralizzata ideale rimane l'alternativa più efficiente, in quanto produce il massimo profitto globale, ma i miglioramenti delle soluzioni DCSP/DCOP rispetto alla soluzione di gestione indipendente sono significativi.

3.4 Modellazione DCOP

I sistemi idrici oggetto degli studi [26] e [11] possono essere facilmente modellati come problemi di ottimizzazione di vincoli distribuiti. Le caratteristiche fisiche del sistema sono ragionevolmente rappresentabili mediante dei vincoli che, in una soluzione ammissibile, non possono essere violati. Questa considerazione porta a ritenere consono per il problema un approccio DCSP, in cui le soluzioni devono obbligatoriamente soddisfare i diversi vincoli. Tuttavia, il formalismo DCSP non rappresenta sufficientemente le caratteristiche di questi problemi, in cui sono definite per ciascun agente delle funzioni obiettivo da ottimizzare; la risoluzione del problema non si limita quindi alla ricerca di soluzioni soddisfacibili, ma tra queste occorre cercare quelle che producono il massimo profitto globale. Inoltre, i vincoli normativi non devono essere

obbligatoriamente considerati come i vincoli fisici, in quanto in determinate condizioni, come la scarsa disponibilità di risorse, potrebbero essere necessariamente violati da alcuni agenti.
Da queste considerazioni deriva quindi la necessità di adottare un approccio di modellazione DCOP, in cui la risoluzione del problema prevede l'ottimizzazione (dei costi) dei vincoli. Il formalismo DCOP non prevede la descrizione esplicita di una funzione obiettivo per un singolo agente (ovvero per una singola variabile), ma solo la definizione delle funzioni costo relative ai vincoli. La traduzione di una funzione obiettivo in un vincolo è banale: la funzione costo del vincolo coincide alla funzione obiettivo. Inoltre, poiché per definizione la violazione di un vincolo normativo si traduce in un costo da pagare globalmente in quanto tale violazione produce teoricamente un danno al sistema, la formulazione di un vincolo normativo come vincolo DCOP è banalmente riconducibile alla definizione di un costo globale da pagare nel caso in cui il vincolo non sia soddisfatto.

3.4.1 Vincoli hard

Nella modellazione DCOP, i vincoli (fisici) non violabili, che possono rendere la soluzione insoddisfacibile, vengono denominati vincoli *hard*. Nell'ambito del framework DCOP, la violazione di un vincolo hard trova una naturale corrispondenza in un costo di vincolo pari ad ∞. Nel dominio dei sistemi idrici, prendendo a titolo d'esempio un canale di portata massima Q, e indicando con x la portata che vi passa attraverso, il vincolo fisico che rappresenta l'impossibilità di superare la relativa soglia è rappresentato in forma di disequazione come $x \leq Q$, mentre nella formulazione DCOP è rappresentato da un vincolo C_H con la seguente funzione costo:

$$C_H(x) = \begin{cases} 0 & x \leq Q \\ \infty & \text{altrimenti} \end{cases}$$

In generale il costo di un vincolo hard $C_H \in \{0, \infty\}$ ed è espresso nella forma

$$C_H(x_i) = \begin{cases} 0 & x_i \text{ accettabile} \\ \infty & x_i \text{ non accettabile} \end{cases}$$

dove x_i può essere una singola variabile o un vettore con un elemento per ogni agente coinvolto nel vincolo.

3.4.2 Vincoli soft

Le decisioni di un singolo agente influenzano direttamente sia la propria funzione di utilità locale che le funzioni di costo relative ai vincoli in cui

l'agente è coinvolto. I vincoli *soft* rappresentano vincoli normativi definiti allo scopo di fornire delle garanzie agli agenti del sistema. Questi vincoli sono violabili, tuttavia costituiscono un costo da pagare nel caso un agente operi delle scelte che in qualche modo danneggiano il resto del sistema. Ad esempio, se in un sistema idrico un agente a monte trattiene una quantità d'acqua che implica una scarsa disponibilità di acqua per gli agenti a valle, questa situazione si traduce in un costo da pagare nella forma di vincolo soft. Così il prelievo eccessivo di una risorsa, oltre a corrispondere ad un maggiore profitto locale per un agente, ne determinerà un maggiore costo da pagare.
Come nell'esempio appena descritto, un vincolo soft può rappresentare un costo da associare ad un agente che sfrutta eccessivamente il sistema; questo tipo di vincolo può però anche rappresentare la situazione opposta, ovvero un costo che il resto del sistema dovrà pagare quando per un agente non siano verificati determinati requisiti, come la quantità minima d'acqua disponibile. La differenza tra i due casi è che nel primo è lo stesso agente che eccede a dover pagare il costo della violazione del vincolo (come decurtazione dal proprio guadagno locale), mentre quando un agente subisce un danno dal resto del sistema sarà il sistema che globalmente paga un costo per il danno arrecatogli. Questa distinzione non è possibile in un DCOP standard, mentre con un modello che mantiene separati tutti gli obiettivi (dei singoli agenti e globale) si può allocare un costo presso l'obiettivo di un qualsiasi agente (o l'obiettivo globale).
Ad esempio il vincolo che rappresenta la quantità minima Q richiesta da un agente che riceve un flusso x si può esprimere come

$$C_S(x) = \begin{cases} 0 & x \geq Q \\ f(Q-x) & \text{altrimenti} \end{cases}$$

dove $f(Q-x)$ rappresenta il costo che il sistema dovrà pagare come funzione dello scarto dalla quantità minima richiesta Q. In generale il costo di un vincolo soft $C_S \in \{0, c \in \mathbb{Z}\}$ viene espresso come

$$C_S(x_i) = \begin{cases} 0 & x_i \text{ accettabile} \\ c(x_i) \in \mathbb{Z} & x_i \text{ non accettabile} \end{cases}$$

3.5 Separazione degli obiettivi

I metodi di risoluzione presentati in [11], come del resto il metodo originale introdotto in [26], risolvono il problema descritto nella Sezione 3.2.1 producendo una singola soluzione ottimale. Le soluzioni trovate dai diversi metodi

possono essere diverse per via delle differenti formulazioni e approcci usati, ma ogni metodo fornisce un'unica soluzione. Nell'ambito di sistemi distribuiti in cui molteplici agenti prendono diverse decisioni è tuttavia possibile (e frequente) che esista più di una singola soluzione ottimale. Si rivela di fondamentale interesse, per un'analisi completa ed approfondita di questi problemi, la ricerca di un approccio che permetta di estrarre tutte le soluzioni potenzialmente ottimali, sia per offrire ai gestori del sistema una scelta tra più opzioni, che per confrontare le soluzioni analizzandone le differenze. Da questa considerazione emerge che il problema da affrontare è un problema di ottimizzazione multiobiettivo. Il formalismo DCOP non dispone della potenzialità espressiva per estrarre tutte le soluzioni ottimali possibili, in quanto non prevede la definizione di obiettivi distinti, e pertanto aggrega i costi di tutti i vincoli in un'unica funzione, da cui deriva come risultato una singola soluzione. Per ottenere l'insieme delle soluzioni ottimali (distinte) di un problema multiobiettivo occorre formulare distintamente i contributi di ogni singolo obiettivo e mantenerli separati durante l'intero processo di risoluzione del problema.

3.6 Discussione sui criteri di ottimalità locale e sulle estensioni del framework DCOP

I concetti di *k-size-optimality* e *t-distance-optimality* introdotti nella Sezione 2.3.5 offrono importanti garanzie di ottimalità locale. L'ottimalità locale trattata in questi studi però è un concetto che non si riferisce ad un'ottimizzazione locale in cui ogni singolo agente cerca di ottenere il massimo per sè, ma fa sempre riferimento alla soluzione globale, in quanto considera il massimo apporto, in termini di utilità, che ogni gruppo di agenti può fornire alla soluzione globale. È chiaro che queste proprietà non sono adatte per l'analisi di problemi multiobiettivo, poiché il grafo dei vincoli di questi ultimi non è omogeneo nel senso che i vari contributi sono in genere correlati a obiettivi distinti.
Si potrebbe applicare questa analisi a dei sottografi ottenuti separando gli obiettivi del problema originale; questi sottografi possono tuttavia non essere connessi poiché i vincoli non hanno necessariamente contributi relativi a tutti gli obiettivi. Isolando ad esempio l'obiettivo globale, l'analisi di ottimalità locale potrebbe delineare come i singoli agenti ne influenzano il relativo valore di utilità, e come gruppi di agenti potrebbero coordinarsi per migliorare questo obiettivo. Ovviamente in questo ambito l'analisi avviene soltanto in relazione ai contributi dell'obiettivo globale, trascurando gli obiettivi privati

degli agenti coinvolti.
La tematica affrontata nell'approccio Distributed Hierarchical CSP, descritto nella Sezione 2.4.1, ha suscitato interesse in quanto nasce dal problema dei singoli agenti di violare il minimo numero di vincoli (ordinati secondo la loro importanza) che li riguardano, con una panoramica di interesse locale nei confronti del problema. Il framework Distributed Hierarchical CSP tuttavia è focalizzato su un ambito ristretto che esula dalle problematiche di ottimizzazione locale in sistemi multiagente che vogliamo affrontare in questo lavoro di tesi.
Il framework MC-DCOP descritto nella Sezione 2.4.2 è di particolare interesse in quanto si inserisce nelle problematiche di collaborazione multiagente e multiobiettivo mantenendo un approccio distribuito. Però il framework MC-DCOP non delinea una distinzione tra i concetti di ottimizzazione locale e globale, ma aggrega di nuovo i contributi dei vari vincoli in un'unica funzione globale. In generale, si può affermare che i vincoli sulle risorse non possiedono l'espressività necessaria a rappresentare i problemi multiobiettivo di interesse per questo lavoro di tesi.
La struttura del processo decisionale di Stakeholder Search introdotto nella Sezione 2.4.3 rimanda ad un approccio diverso da quello desiderato, in quanto prevede una serie di meeting nei quali gli agenti propongono delle soluzioni sottostanti al regolamento fornito dal leader, scegliendo alla fine del processo la soluzione che meglio soddisfa l'obiettivo globale. Questo approccio basato su meeting coordinati da un leader esula dalle condizioni in cui si presenta il nostro caso, pertanto quest'idea è stata scartata.

3.7 Ottimizzazione multiobiettivo

L'ottimizzazione multiobiettivo (anche denominata programmazione multiobiettivo o ottimizzazione di Pareto) è il processo di ottimizzazione simultanea di due o più obiettivi conflittuali soggetti ad alcuni vincoli. Problemi di ottimizzazione multiobiettivo possono essere massimizzazione di profitto e minimizzazione del costo di un prodotto, massimizzazione delle performance e minimizzazione del consumo di un veicolo, minimizzazione del peso e massimizzazione della forza di un componente. Per problemi multiobiettivo non banali, non è possibile trovare una singola soluzione che ottimizzi simultaneamente ciascun obiettivo. Nella ricerca delle soluzioni, si raggiungono situazioni in cui, nel tentativo di migliorare un obiettivo, se ne peggiora un altro.
Un problema di ottimizzazione multiobiettivo è definito come l'ottimizzazione simultanea di k funzioni obiettivo incommensurabili, definite su un

insieme $\mathbf{x} = \{x_1, \ldots, x_M\}$ di M variabili, dove ogni variabile x_j può assumere valori appartenenti al proprio dominio $D_{x_j} = \{d_j^1, \ldots, d_j^{|D_{x_j}|}\}$. Una soluzione di un problema di ottimizzazione multiobiettivo è un assegnamento $\mathbf{a}^* = \{x_1 = d_1^{(1)}, \ldots, x_M = d_M^{(M)}\}$ di valori alle variabili che massimizza ogni singolo elemento del vettore delle funzioni obiettivo $\mathbf{U}(\mathbf{x})$:

$$\mathbf{a}^* = \arg\max_{\mathbf{a} \in D_x} \mathbf{U}(\mathbf{x}) = [U^1(\mathbf{x}), \ldots, U^k(\mathbf{x})]^T \tag{3.1}$$

dove $D_x = \times_{j=1}^{M} D_{x_j}$ è il dominio delle variabili $\mathbf{x}$. Ogni funzione obiettivo U^i può essere definita su un sottoinsieme $\mathbf{x}_i \subseteq \mathbf{x}$ delle variabili del problema. Un approccio intuitivo per la risoluzione di problemi multiobiettivo è la costruzione di una funzione aggregata, combinando tutte le funzioni obiettivo in un'unica funzione, ad esempio la somma pesata degli obiettivi; viene quindi ottimizzata questa singola funzione specificando quanto un singolo obiettivo può venire sacrificato per ottenere migliori risultati inerenti altri obiettivi. I metodi che selezionano una singola soluzione come preferibile rispetto alle altre sono soggettivi, e richiedono che qualche gestore delle decisioni fornisca i pesi per gli obiettivi; inoltre la soluzione ottima calcolata con questi metodi dipenderà dai valori relativi dei pesi assegnati agli obiettivi.

3.7.1 Pareto-ottimalità

Le funzioni che caratterizzano un problema di ottimizzazione multiobiettivo sono incommensurabili, ed è possibile che molteplici assegnamenti massimizzino l'Equazione 3.1, generando quindi soluzioni ottimali. Per caratterizzare le soluzioni ottimali di un problema di ottimizzazione multiobiettivo viene usato il concetto di Pareto-ottimalità.
La Pareto-ottimalità, o Pareto-efficienza, è un concetto appartenente alle scienze economiche che trova applicazioni anche in campo ingegneristico. In una allocazione economica Pareto-efficiente, non è possibile aumentare il guadagno (la rendita) di un individuo senza peggiorare quello di un altro. Data un'allocazione iniziale di beni tra un insieme di individui, il passaggio ad una allocazione differente che aumenta il guadagno di almeno un individuo senza diminuire la rendita di alcun altro individuo viene definito come Pareto-miglioramento. Un'allocazione è definita Pareto-efficiente o Pareto-ottimale quando non è possibile effettuare ulteriori Pareto-miglioramenti. La Pareto-ottimalità è una nozione minimale di efficienza e non risulta necessariamente in una distribuzione di risorse socialmente desiderabile: non considera l'uguaglianza tra gli individui o il benessere globale di una società. Nell'ambito ingegneristico, il concetto di Pareto-efficienza viene usato per

la selezione di alternative, in cui ogni opzione viene valutata secondo molteplici criteri, identificando un sottoinsieme di opzioni con la proprietà che nessun'altra opzione può fornire risultati migliori.
Dato un insieme di opzioni e un criterio per valutarle, la frontiera di Pareto è l'insieme delle opzioni che sono Pareto-ottimali.

Definizione 8 (Pareto-ottimalità)**.** *Un assegnamento* $\mathbf{a}^*$ *è Pareto-ottimale se e solo se non esiste nessun altro assegnamento* $\mathbf{a}$ *tale che* $\mathbf{U}(\mathbf{a}) \geq \mathbf{U}(\mathbf{a}^*)$*, e* $U^i(\mathbf{a}) > U^i(\mathbf{a}^*)$ *per almeno una funzione obiettivo.*

Il vettore delle utilità $\mathbf{U}(\mathbf{a}^*)$ corrispondente ad un assegnamento Pareto-ottimale $\mathbf{a}^*$ viene indicato come vettore *non-dominato*. Il concetto di *non-dominanza* viene definito come:

Definizione 9 (Non-dominanza)**.** *Un vettore* $\mathbf{U}(\mathbf{a}^*) \in D_x$ *è non-dominato se e solo se non esiste alcun assegnamento* $\mathbf{a} \in D_x$*, tale che* $\mathbf{U}(\mathbf{a}) \geq \mathbf{U}(\mathbf{a}^*)$*, con almeno un* $U^i(\mathbf{a}) > U^i(\mathbf{a}^*)$*. In caso contrario,* $\mathbf{U}(\mathbf{a}^*)$ *si dice dominato.*

Pertanto, poiché un problema multiobiettivo coinvolge l'ottimizzazione su assegnamenti parzialmente ordinati, la sua soluzione è un insieme di assegnamenti Pareto-ottimali.

3.8 Motivazioni dell'approccio multiobiettivo

In questa sezione si vogliono meglio argomentare i motivi che rendono necessario un approccio multiobiettivo per l'analisi del nostro problema, e mostrare come l'uso del framework DCOP standard non sia sufficiente a modellare il sistema e soprattutto a ricavarne l'intero insieme delle soluzioni Pareto-ottimali.
Ci si serve in questo ambito di un semplice esempio, rappresentato in Figura 3.2, che consiste di un sistema costituito da due agenti che controllano rispettivamente le variabili x_1 e x_2, e per ciascuno dei quali è definita una funzione obiettivo locale ($f_1(x_1)$ e $f_2(x_2)$). Tra gli agenti è posto un vincolo rappresentato dalla funzione $f_G(x_1, x_2)$.
Si considera il problema di massimizzazione simultanea dei tre obiettivi rappresentati da ciascuna funzione, con lo scopo di ottenere l'insieme di tutte le soluzioni Pareto-ottimali. Una singola soluzione generica consiste in un vettore delle funzioni obiettivo $[f_1, f_2, f_G]$. Dato un assegnamento delle variabili (x_1^*, x_2^*), la soluzione del problema corrispondente è data dal vettore dei valori $[f_1(x_1^*), f_2(x_2^*), f_G(x_1^*, x_2^*)]$.

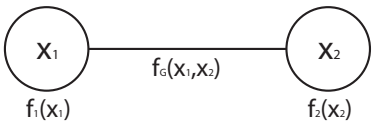

Figura 3.2: Grafo del sistema composto dalle variabili x_1 e x_2, con funzioni obiettivo locali $f_1(x_1)$ e $f_2(x_2)$, legate dal vincolo $f_G(x_1, x_2)$

Le funzioni locali degli agenti sono delle parabole con massimo in $x_i = 0$, scelte per semplicità, mentre la funzione relativa al vincolo f_G è una funzione lineare delle due variabili, nello specifico è la differenza tra le due.

$$\begin{cases} f_1(x_1) = -x_1^2 \\ f_2(x_2) = -x_2^2 + 1 \\ f_G(x_1, x_2) = x_1 - x_2 \end{cases}$$

Ipotizzando che questo modello rappresenti un sistema idrico nel quale gli agenti prelevano da un flusso d'acqua rispettivamente le quantità x_1 e x_2 traendovi profitto locale, si può interpretare il vincolo come la richiesta che la quantità prelevata x_1 sia maggiore di x_2; in questo caso, essendo il problema oggetto di massimizzazione, la funzione f_G restituirebbe un'utilità positiva, mentre nella situazione opposta si avrebbe un'utilità negativa. Nei termini dell'obiettivo globale f_G, si ha una soluzione migliore quanto più x_1 è maggiore di x_2.

3.8.1 Risoluzione DCOP standard e multiobiettivo

Per osservare le differenze tra i risultati ottenuti con un approccio DCOP standard e un approccio multiobiettivo, viene definito un semplice dominio discreto finito per entrambe le variabili: $D(x_1) = D(x_2) = \{-1, 0, +1\}$, e utilizzando questo dominio si costruiscono le tabelle delle possibili combinazioni di x_1 e x_2 con i relativi valori calcolati delle funzioni obiettivo (le relative soluzioni) con entrambi i metodi.
Nell'approccio DCOP standard le funzioni utilità vengono considerate come vincoli e aggregate in un'unica funzione globale F, calcolata come la somma di tutte le funzioni obiettivo:

$$F(x_1, x_2) = f_1(x_1) + f_2(x_2) + f_G(x_1, x_2)$$

Le soluzioni ottimali saranno quindi quelle corrispondenti al massimo valore di F.
Utilizzando invece l'approccio multiobiettivo, le funzioni non vengono aggregate, ma sono calcolate separatamente, ottenendo così dei vettori di soluzioni

$[f_1, f_2, f_G]$; le soluzioni ottimali vengono identificate confrontando i vari vettori ottenuti. I risultati ottenuti con i due approcci sono listati nella Tabella 3.2.

x_1	x_2	$f_1(x_1)$	$f_2(x_2)$	$f_G(x_1, x_2)$	$F(x_1, x_2)$
-1	-1	-1	0	0	-1
-1	0	-1	+1	-1	-1
-1	+1	-1	0	-2	-3
0	-1	0	0	1	+1
0	0	0	+1	0	+1
0	+1	0	0	-1	-1
+1	-1	-1	0	2	+1
+1	0	-1	+1	1	+1
+1	+1	-1	0	0	-1

Tabella 3.2: Soluzioni elaborate con gli approcci DCOP standard e multiobiettivo per il sistema rappresentato in Figura 3.2

Analizzando i dati rappresentati nella Tabella 3.2, si ottengono quattro soluzioni ottimali (x_1^*, x_2^*) con entrambi i metodi: $\{(0,-1),(0,0),(+1,-1),(+1,0)\}$. Con il metodo standard queste soluzioni corrispondono ad un valore $F = +1$, mentre con il metodo multiobiettivo corrispondono ai vettori $[f_1, f_2, f_G]$ $\{[0,0,1],[0,1,0],[-1,0,2],[-1,1,1]\}$. In questo caso i due approcci appaiono equivalenti, poiché generano lo stesso insieme di soluzioni ottimali.

3.8.2 Risoluzione analitica centralizzata

Si vogliono ora osservare le soluzioni calcolabili mediante un'analisi analitica delle funzioni obiettivo, abbandonando in questo caso il dominio discreto definito in precedenza e cercando le soluzioni solamente in base alla massimizzazione delle singole funzioni.
Essendo le funzioni locali dei due agenti $f_1(x_1)$ e $f_2(x_2)$ funzioni della sola variabile controllata dagli stessi agenti, è possibile massimizzarle indipendentemente. Analizzando il caso in cui gli agenti ottimizzano il proprio profitto locale senza badare agli interessi globali del sistema (rappresentati dalla funzione globale relativa al vincolo f_G), ogni agente sceglie per la propria variabile il valore che massimizza la propria funzione obiettivo, corrispondente al massimo della parabola; nell'esempio questa scelta si traduce nell'assegnamento $\mathcal{A}$:

$$\mathcal{A} = \begin{cases} \overline{x_1} = 0 \\ \overline{x_2} = 0 \end{cases}$$

Da cui si calcolano i valori delle funzioni obiettivo:

$$\begin{cases} f_1(\overline{x_1}) = -\overline{x_1}^2 = 0 \\ f_2(\overline{x_2}) = -\overline{x_2}^2 + 1 = 1 \\ f_G(\overline{x_1}, \overline{x_2}) = \overline{x_1} - \overline{x_2} = 0 \end{cases}$$

Ottenendo come soluzione il vettore $[0, 1, 0]$, che corrisponde ad uno dei vettori appartenenti agli insiemi calcolati nel paragrafo precedente, ed è pertanto ottimale.
Questa soluzione rispecchia la condizione in cui ogni agente ottimizza il proprio obiettivo senza curarsi del resto del sistema, ma è comunque basata su un approccio multiobiettivo poiché, nonostante per la massimizzazione non si sia considerata la funzione globale f_G, le funzioni non sono state aggregate ma sono state massimizzate indipendentemente, ottenendo come soluzione un vettore di valori per le funzioni obiettivo.
Volendo invece considerare anche il contributo della funzione globale f_G, non è più possibile ottimizzare ciascuna funzione obiettivo indipendentemente, ma occorre sommare i contributi di tutte le funzioni in un'unica funzione aggregata, ovvero la stessa funzione F calcolata con l'approccio DCOP standard nel paragrafo precedente. Calcolando poi le derivate parziali della funzione F, si ottengono i valori di x_1 e x_2 che ottimizzano la funzione aggregata, e da questi valori si può nuovamente calcolare il vettore soluzione corrispondente.

$$F(x_1, x_2) = f_1(x_1) + f_2(x_2) + f_G(x_1, x_2) = -x_1^2 + x_1 - x_2^2 - x_2 + 1$$

Le cui derivate parziali sono:

$$\begin{cases} \frac{\partial F}{\partial x_1} = -2x_1 + 1 \\ \frac{\partial F}{\partial x_2} = -2x_2 - 1 \end{cases}$$

Calcolando i valori che annullano le derivate parziali, e che cioè massimizzano la funzione F, si ottiene l'assegnamento $\mathcal{A}'$:

$$\mathcal{A}' = \begin{cases} \overline{x_1}' = \arg\max_{x_1} F(x_1, x_2) = 0.5 \\ \overline{x_2}' = \arg\max_{x_2} F(x_1, x_2) = -0.5 \end{cases}$$

Da cui si calcolano nuovamente i valori delle singole funzioni obiettivo:

$$\begin{cases} f_1(\overline{x_1}') = -\overline{x_1}'^2 = -0.25 \\ f_2(\overline{x_2}') = -\overline{x_2}'^2 + 1 = 0.75 \\ f_G(\overline{x_1}', \overline{x_2}') = \overline{x_1}' - \overline{x_2}' = 1 \end{cases}$$

Si ottiene quindi come nuova soluzione il vettore $v^* = [-0.25, 0.75, 1]$. Questa soluzione ovviamente non rientra tra quelle trovate nel paragrafo precedente, poiché qui le variabili x_1 e x_2 assumono valori non interi che non appartengono ai domini precedentemente definiti. Confrontando però questo vettore v^* con i quattro vettori ottimali estratti dalla Tabella 3.2 con il metodo DCOP multiobiettivo, si osserva che non è dominato dai vettori ottenuti in precedenza, pertanto è anche questa una soluzione ottimale.
Se invece calcoliamo il valore della funzione aggregata F per quest'ultimo assegnamento $\mathcal{A}'$, otteniamo:

$$F(\overline{x_1}', \overline{x_2}') = F^* = 1.5$$

Il valore della funzione aggregata F per le soluzioni appartenenti all'insieme calcolato in precedenza è $F = 1$; pertanto effettuando il confronto con un approccio DCOP tra queste soluzioni e quella appena trovata, si concluderebbe che vi è un'unica soluzione ottima $F^* = 1.5$, quindi le soluzioni trovate in precedenza sarebbero ritenute non ottimali e scartate, mentre si è invece dimostrato che sono Pareto-ottimali.

3.8.3 Verifica delle considerazioni sugli approcci DCOP standard e multiobiettivo

Come ulteriore argomentazione a queste considerazioni, si è scelto di riaffrontare l'esempio iniziale con gli approcci DCOP standard e multiobiettivo, modificando i domini delle variabili in modo da includere tra le soluzioni possibili anche quella trovata con il metodo delle derivate parziali. Il problema da risolvere rimane identico a quello iniziale, con le medesime variabili e funzioni obiettivo, mentre i domini delle variabili diventano: $D'(x_1) = \{0; 0.5; 1\}$ e $D'(x_2) = \{-1; -0.5; 0\}$.
La Tabella 3.3 contiene i risultati ottenuti con i metodi DCOP standard e multiobiettivo utilizzando i nuovi domini.

Dai dati contenuti nella Tabella 3.3, si può osservare che nessuno dei vettori $[f_1, f_2, f_G]$ calcolati con l'approccio multiobiettivo è dominato, pertanto si ottengono nove soluzioni ottimali. Considerando invece la funzione somma F, si ottiene un'unica soluzione ottima, corrispondente a F^*. È chiaro che con l'approccio DCOP standard vengono eliminate soluzioni che invece sono ottimali.

x_1	x_2	$f_1(x_1)$	$f_2(x_2)$	$f_G(x_1, x_2)$	$F(x_1, x_2)$
0	-1	0	0	1	1
0	-0.5	0	0.75	0.5	1.25
0	0	0	1	0	1
0.5	-1	-0.25	0	1.5	1.25
0.5	-0.5	-0.25	0.75	1	1.5
0.5	0	-0.25	1	0.5	1.25
1	-1	-1	0	2	1
1	-0.5	-1	0.75	1.5	1.25
1	0	-1	1	1	1

Tabella 3.3: Soluzioni elaborate con gli approcci DCOP standard e multiobiettivo per il sistema rappresentato in Figura 3.2 con domini modificati

3.8.4 Aggregazione delle funzioni obiettivo locali

Una formulazione alternativa potrebbe prevedere l'aggregazione di tutte le funzioni obiettivo locali degli agenti, che sono funzioni di una singola variabile e quindi sono teoricamente ottimizzabili indipendentemente. In questa formulazione si avrebbero quindi due obiettivi: l'obiettivo globale e l'obiettivo f_O, risultato dell'aggregazione di tutti gli obiettivi locali ($f_O = \sum_{i=1}^{N} f_i(x_i)$). I risultati ottenuti attraverso questa formulazione sono rappresentati nella Tabella 3.4.

x_1	x_2	$f_O(x_1, x_2)$	$f_G(x_1, x_2)$
0	-1	0	1
0	-0.5	0.75	0.5
0	0	1	0
0.5	-1	-0.25	1.5
0.5	-0.5	0.5	1
0.5	0	0.75	0.5
1	-1	-1	2
1	-0.5	-0.25	1.5
1	0	0	1

Tabella 3.4: Soluzioni elaborate con l'aggregazione degli obiettivi locali $[f_O, f_G]$ per il sistema rappresentato in Figura 3.2

Anche in questo caso si perdono però soluzioni ottimali, poiché sommando i contributi delle varie funzioni si perdono i contributi e gli interessi di ogni singolo agente. I vettori $[f_1, f_2]$ $[0, 0]$ e $[-1, 1]$, corrispondenti alla prima e all'ultima riga della Tabella 3.4, sono entrambi ottimali, come dimostrato in precedenza, ma con l'aggregazione delle funzioni obiettivo locali verrebbero scartati in quanto la somma dei loro valori è 0 e il vettore corrispondente $[f_O, f_G]$ $[0, 1]$ è sarebbe scartato perché peggiore del vettore [0.5,1].
In conclusione, per ottenere la frontiera di tutte le soluzioni effettivamente ottimali occorre mantenere distinti tutti i singoli obiettivi nella formulazione

del problema.

3.9 Multi-Objective COP

Un problema di ottimizzazione di vincoli (COP) consiste nella minimizzazione (o massimizzazione) di una funzione obiettivo soggetta ad un insieme di vincoli posti sui valori assegnabili ad un insieme di variabili decisionali indipendenti. Molti problemi del mondo reale coinvolgono molteplici obiettivi che devono essere ottimizzati simultaneamente. Contrariamente a quanto accade per una singola funzione obiettivo, l'ottimizzazione simultanea di diverse funzioni obiettivo, in alcuni casi conflittuali, non ammette in generale un'unica soluzione. Un problema di ottimizzazione di vincoli multiobiettivo (MO-COP) tende ad essere caratterizzato da un insieme di alternative da considerarsi equivalenti tra loro in assenza di informazioni riguardanti l'importanza di ogni obiettivo in relazione agli altri. Pertanto, la risoluzione di un MO-COP consiste nella ricerca di una frontiera efficiente, nominalmente l'insieme delle soluzioni con costi non-dominati tra le soluzioni possibili.
Gli algoritmi per la risoluzione di MO-COP rientrano tipicamente in due categorie. Gli algoritmi basati su *ricerca* (come branch-and-bound) scompongono il problema in più sottoproblemi risolti sequenzialmente applicando ricorsivamente le stesse regole; questi algoritmi non considerano la struttura del problema, hanno complessità temporale esponenziale con il numero di variabili, ma operano con complessità spaziale polinomiale. Gli algoritmi basati su *inferenza* (come eliminazione delle variabili) sfruttano al meglio invece le informazioni strutturali codificate nel problema. Questi metodi applicano una sequenza di trasformazioni che riducono la dimensione del problema, mantenendone inalterato lo spazio delle soluzioni; hanno complessità spaziale e temporale esponenziali relativamente all'ampiezza dell'albero (il numero massimo di figli di un nodo, sempre minore o uguale al numero di variabili), e per questo sono spesso impraticabili quando il valore del parametro è grande.
Lo spazio di ricerca AND/OR per COP [6] è un framework basato su pseudo-alberi che cattura le indipendenze condizionali nel problema, ed elabora un albero di ricerca esponenziale con la profondità dello pseudo-albero anzichè con il numero di variabili. L'AND/OR Branch-and-Bound [17] è un algoritmo di ricerca che incrocia ricerca in profondità e alberi di ricerca AND/OR associati ad una singola istanza monobiettivo COP. Questo framework viene esteso all'ottimizzazione multiobiettivo diventando MO-AOBB (Multi-Objective AND/OR Branch-and-Bound) [16], per la ricerca della frontiera efficiente di un'istanza MO-COP. L'efficienza di questo algoritmo dipende

dall'accuratezza della sua funzione euristica.
Questo studio riguarda però un problema centralizzato, in quanto MO-COP è un problema in cui le variabili non sono distribuite tra vari agenti e non ci si trova quindi in un sistema multiagente: sia la definizione del problema che gli algoritmi sono pertanto distanti in questo senso dall'approccio riguardante la modellazione di sistemi idrici che vogliamo realizzare in questo lavoro.

3.10 Multi-Objective DCOP

Diversi problemi del mondo reale implicano l'ottimizzazione di molteplici obiettivi, eventualmente conflittuali tra loro. In tale ambito, l'ottimizzazione indipendente di un obiettivo comporta un danno o una riduzione dell'utilità per gli altri obiettivi. Spesso i problemi multiobiettivo hanno una natura potenzialmente critica, pertanto un controllo centralizzato di questi sistemi non è auspicabile poiché genera un singolo punto di errore. Di conseguenza, è stato sostenuto l'uso di tecnologie multiagente al fine di raggiungere un risultato di controllo affidabile, robusto e scalabile di questi scenari. In particolare, sono stati proposti diversi algoritmi distribuiti per l'ottimizzazione di vincoli al fine di permettere a molteplici agenti di coordinare le proprie azioni nell'intento di raggiungere i propri obiettivi. Tuttavia, mentre sia l'ottimizzazione di vincoli multiobiettivo che il coordinamento distribuito hanno generato considerevole interesse, l'*ottimizzazione multiobiettivo distribuita* ha ricevuto scarsa attenzione. Da un lato, lo studio dell'ottimizzazione distribuita si è focalizzato esclusivamente su problemi che coinvolgono un unico obiettivo, che vengono spesso rappresentati come DCOP.
La generalizzazione di DCOP al caso multiobiettivo ha prodotto come risultato il framework Multi-Objective Distributed Constraint Optimization Problem (MO-DCOP [7]). In un MO-DCOP sono definiti diversi obiettivi: per ogni vincolo non viene definita un'unica funzione utilità, ma ne viene definita una per ogni obiettivo (pertanto ogni vincolo è di per sè una funzione multiobiettivo). Le utilità dei vari vincoli non vengono così aggregate globalmente in un unico valore; l'aggregazione delle utilità avviene distintamente per ciascun obiettivo. Una soluzione di un MO-DCOP pertanto non consiste in un unico valore, ma in un vettore di valori ciascuno corrispondente ad un singolo obiettivo e derivante dall'aggregazione dei contributi di ogni vincolo per ciascun obiettivo.

3.10.1 Formalizzazione MO-DCOP

La formalizzazione di un MO-DCOP avviene tramite un'estensione del framework DCOP al caso multiobiettivo. Questo nuovo formalismo coinvolge l'ottimizzazione simultanea di k DCOP (uno per ciascun obiettivo), si considera nello specifico il problema di massimizzazione del vettore delle funzioni obiettivo, definito come:

$$\mathbf{U}(\mathbf{x}) = \left[U^1(\mathbf{x}), \ldots, U^k(\mathbf{x})\right]^T$$

In cui $\mathbf{x}$ è l'insieme delle variabili del problema. Poiché ogni componente di $\mathbf{U}(\mathbf{x})$ è un DCOP, $\mathbf{U}(\mathbf{x})$ stesso può essere decomposto in N fattori, ciascuno dei quali rappresenta una funzione di vincolo multiobiettivo $\mathbf{U}_i$:

$$\mathbf{U}(\mathbf{x}) = \sum_{i=1}^{M} \mathbf{U}_i(\mathbf{x}_i)$$

dove $\mathbf{x}_i \subseteq \mathbf{x}$, e da cui segue la definizione di funzione di vincolo multiobiettivo $\mathbf{U}_i$:

$$\mathbf{U}_i(\mathbf{x}_i) = \left[U_i^1(\mathbf{x}_i), \ldots, U_i^k(\mathbf{x}_i)\right]^T$$

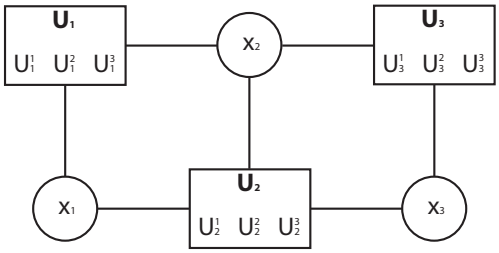

Figura 3.3: Grafo fattorizzato composto dai nodi-variabile x_1, x_2, x_3, nodi-funzione U_1, U_2, U_3 e obiettivi U^1, U^2 e U^3

La formulazione MO-DCOP viene infine codificata in un grafo fattorizzato (di cui un esempio è illustrato in Figura 3.3) composto da nodi-variabile che rappresentano le variabili x_i e nodi-funzione che rappresentano i vincoli (funzioni vettoriali multiobiettivo $\mathbf{U}_i$). Le soluzioni di un MO-DCOP sono caratterizzate attraverso i concetti di Pareto-ottimalità e non-dominanza. Seguendo queste definizioni, le soluzioni di un MO-DCOP sono caratterizzate come un insieme di assegnamenti ottimali PO corrispondenti ad un insieme di vettori utilità non-dominati ND. $\prod_{i=1}^{M} |D_i|$ è un limite superiore sul numero di soluzioni possibili, uguale al prodotto cartesiano dei domini di tutte le variabili.

3.10.2 B-MOMS

Il primo algoritmo risolutivo progettato per MO-DCOP è il Bounded Multi-Objective Max-Sum (B-MOMS) [7]. B-MOMS si compone di tre fasi:

- Fase *bounding* (B), che estende l'algoritmo Bounded Max-Sum introdotto nella Sezione 2.3.4, al fine di ottenere un sottografo aciclico del grafo fattorizzato multiobiettivo. L'algoritmo che genera lo spanning tree massimale al termine di questa fase viene generalizzato per l'analisi di vettori di pesi.
- Fase *max-sum* (MS), durante la quale gli agenti si coordinano per trovare l'insieme Pareto-ottimale di soluzioni al grafo fattorizzato privo di cicli computato nella prima fase. Questa fase richiede una ridefinizione delle due operazioni chiave richieste dall'algoritmo Max-Sum (somma e massimizzazione marginale) per il caso multiobiettivo.
- Fase *value-propagation* (VP), in cui gli agenti scelgono un assegnamento consistente alle variabili. Estende la fase VP standard al caso in cui esistono molteplici alternative incommensurabili.

Fase bounding

Questa fase è costruita sull'algoritmo Bounded Max-Sum estendendo i pesi d'arco w_{ij} a vettori di pesi multiobiettivo. A questo scopo, viene computato l'impatto di ogni variabile x_j nello scope di ogni funzone multiobiettivo locale $\mathbf{U}_i$ per tutti gli obiettivi:

$$\mathbf{w}_{ij} = \left[w_{ij}^1, \ldots, w_{ij}^k\right]^T$$

Inoltre, viene definito ogni peso scalare w_{ij}^o $(1 \leq o \leq k)$ come nel caso monobiettivo:

$$w_{ij}^o = \max_{\mathbf{x}_i \setminus x_j} \left[\max_{x_j} U_i^o(\mathbf{x}_i) - \min_{x_j} U_i^o(\mathbf{x}_i)\right]$$

Poiché la ricerca di uno spanning tree massimale è definita per istanze con pesi degli archi scalari, è necessario classificare i vettori di pesi. Questa procedura deve garantire che l'ordinamento risultante favorisca l'eliminazione dei vettori dominati rispetto ai non-dominati. Una via per fare questo è assegnare un peso scalare w_{ij} ad ogni vettore $\mathbf{w}_{ij}$, proporzionale al numero di pesi degli archi che domina. Formalmente:

$$w_{ij} = -|\{\mathbf{w}_{mn} | \mathbf{w}_{mn} > \mathbf{w}_{ij}, (i,j) \neq (m,n)\}|$$

Usando questa scalarizzazione, ai vettori di pesi non-dominati viene assegnato il valore 0, ai vettori dominati da un singolo elemento viene assegnato il valore -1, e così via. Attraverso questi pesi scalari, l'algoritmo GHS [10] può essere usato per computare uno spanning tree massimale.

Fase max-sum

La seconda fase esegue l'algoritmo Max-Sum sul grafo fattorizzato aciclico ottenuto nella fase precedente. Per applicare Max-Sum al problema multiobiettivo occorre generalizzare i messaggi scambiati tra funzioni e variabili passando da singoli valori a vettori di funzioni vincolo.
Se il grafo fattorizzato è aciclico, i messaggi r e q scambiati tra U_i e x_j rappresentano la massima utilità aggregata sui due componenti del grafo ottenuto rimuovendo la dipendenza tra U_i e x_j. Lo stesso vale nel caso multiobiettivo. Tuttavia, $q_{j\to i}(x_j)$ e $r_{i\to j}(x_j)$ non sono più funzioni scalari di x_j, ma questi messaggi mappano il dominio di x_j su un insieme di vettori utilità. Come verifica, si nota che nel dominio multiobiettivo l'utilità massima è definita in termini di relazione di dominanza; siccome questa relazione induce un ordinamento parziale, potrebbe esistere più di un massimo.
La computazione dei messaggi richiede la ridefinizione dei due operatori matematici chiave richiesti da Max-Sum: la somma di due funzioni vettoriali e la massimizzazione marginale rispetto ad una singola variabile. Questi operatori sono stati definiti in precedenza nel contesto dell'algoritmo Bucket Elimination Multiobiettivo [24], e vengono ridefiniti per l'algoritmo B-MOMS aggiungendo due funzioni f e g definite sullo scope $\mathbf{x}$:

$$(f+g)(\mathbf{x}) = ND(\{\mathbf{v}+\mathbf{w} | \mathbf{v} \in f(\mathbf{x}); \mathbf{w} \in g(\mathbf{x})\})$$

in cui la funzione $ND(A)$ filtra i vettori dominati dall'insieme A. L'uso dell'operatore + consente la computazione della somma dei messaggi r (che sono funzioni univariate di x_j) così come l'addizione della funzione multivariata $\mathbf{U}_i$ alla somma delle funzioni univariate di variabili x_k differenti.
Il secondo operatore, massimizzazione marginale $(\max_{\mathbf{x}_i \backslash x_j})$ riceve come input una funzione vettore multi-variata $(f(\mathbf{x}_i))$ e restituisce una funzione $f'(x_j)$:

$$f'(x_j = d) = ND\left(\bigcup_{\mathbf{d} \in D_{\mathbf{x}_i \backslash x_j}} f(\{\mathbf{x}_i \backslash x_j\} = \mathbf{d}, x_j = d)\right)$$

dove $D_{\mathbf{x}_i \backslash x_j}$ è il prodotto cartesiano dei domini delle variabili $\mathbf{x}_i \backslash x_j$.
Al termine della fase max-sum, ogni variabile x_j computa la propria funzione

marginale z_j per ottenere un insieme di assegnamenti Pareto-ottimali $\mathcal{A}_j^*$. Poiché potrebbero esserci molteplici assegnamenti ottimali, gli agenti devono raggiungere un accordo su quale assegnamento globale scegliere. Questo avviene nella fase successiva, in cui gli agenti scelgono insieme una soluzione Pareto-ottimale tra quelle computate nella fase MS.

Fase value-propagation

La terza ed ultima fase opera nuovamente sul grafo fattorizzato aciclico computato nella fase bounding. In questa fase, le variabili e i nodi funzione del grafo fattorizzato si coordinano per scegliere un assegnamento consistente alle variabili.
Al termine della fase MS, ogni variabile computa l'insieme $\mathcal{A}_j^*$ degli assegnamenti marginali Pareto-ottimali per x_j, ottenuto massimizzando la funzione marginale z_j:

$$\mathcal{A}_j^* = \arg\max_{x_j} z_j(x_j)$$

Se, per qualsiasi variabile x_j, $|\mathcal{A}_j^*| > 1$, allora esiste più di un singolo assegnamento globale ottimale: $|\widetilde{\mathbf{PO}}| > 1$. In questo caso, una variabile può scegliere un assegnamento $a_j^* \in \mathcal{A}_j^*$ casualmente, oppure un assegnamento che soddisfi qualche condizione logica C, ad esempio in relazione a un particolare obiettivo (per esempio massimizzare l'obiettivo i: $\max z_j(a_j^*)_i$). Per selezionare un assegnamento che soddisfi questa condizione, la fase value-propagation procede attraverso uno scambio di messaggi tra le variabili e le funzioni nel grafo fattorizzato aciclico, con un approccio completamente distribuito. La variabile x_r con l'indice più basso viene scelta come radice dell'albero, ed è responsabile per l'inizio della fase value-propagation selezionando un assegnamento Pareto-ottimale a_r^* che soddisfi C. La variabile invia quindi messaggi Value-Propagation ($x_r = a_r^*$) a tutti i nodi-funzione a cui è connessa. Il comportamento di tutti gli altri nodi nel grafo dipenderà dal loro tipo. Nello specifico:

Nodi-funzione: Alla ricezione di un messaggio ($x_j = a_j^*$) da una variabile x_j, la funzione di vincolo multiobiettivo $\mathbf{U}_i$ computa l'insieme $\mathcal{A}^*$ degli assegnamenti Pareto-ottimali locali per le variabili $\mathbf{x}_i \backslash x_j$, condizionato da $x_j = a_j^*$:

$$\mathcal{A}^* = \left\{ \mathbf{a} | \mathbf{a} \in D_{\mathbf{x}_i \backslash x_j}, g(\mathbf{a}) \in ND \left(\bigcup_{\mathbf{a} \in D_{\mathbf{x}_i \backslash x_j}} g(\mathbf{a}) \right) \right\}$$

in cui $g(\mathbf{a})$ è definita come:

$$g(\mathbf{a}) = \mathbf{U}_i\left(\{\mathbf{x}_i \backslash x_j\} = \mathbf{a}, x_j = a_j\right) + \sum_{k \in N(i) \backslash j} q_{k \to i}(a_k)$$

Una volta computato l'insieme $\mathcal{A}^*$, value-propagation sceglie un assegnamento Pareto-ottimale $\mathbf{a}^* \in \mathcal{A}^*$ che soddisfi la condizione C ed invia il messaggio $(x_k = a_k^*)$ ad ogni variabile $x_k (k \neq j)$, dove a_k^* è l'elemento di $\mathbf{a}^*$ corrispondente a x_k.

Nodi-variabile: Per ogni variabile non-radice x_j, una volta ricevuto un messaggio $(x_j = a_j^*)$ da una funzione $\mathbf{U}_i$, essa assegna al proprio valore il valore a_j^* e propaga il messaggio $(x_j = a_j^*)$ a tutti i nodi funzione $\mathbf{U}_k, k \neq i$.

Durante la fase VP, un singolo messaggio viene spedito attraverso ogni collegamento nel grafo fattorizzato. Pertanto l'algoritmo termina quando ogni nodo-variabile non-radice ha ricevuto un messaggio Value-Propagation.

Ottimalità della fase max-sum

Teorema 1. *La fase max-sum (MS) computa l'insieme completo delle soluzioni Pareto-ottimali* $\widetilde{\mathbf{PO}}$ *del grafo fattorizzato aciclico ottenuto potando i rami durante la fase bounding (B).*

Schema della dimostrazione. La dimostrazione si divide in due passi. L'algebra definita dagli operatori usati nella fase MS, assieme al co-dominio della funzione di vincolo multiobiettivo globale $\mathbf{U}$ è un semi-anello commutativo. Ogni algoritmo GDL risolve in modo ottimale problemi la cui algebra di valutazione è un semi-anello commutativo, quando il sottostante grafo dei vincoli è aciclico. Quindi essendo la fase MS un algoritmo GDL, il risultato vale. □

Mentre le soluzioni $\widetilde{\mathbf{PO}}$ sono Pareto-ottimali nel sottografo aciclico, esse non sono necessariamente Pareto-ottimali nel grafo fattorizzato originale. Usando il teorema di cui sopra si possono derivare limiti sulla qualità di queste soluzioni nel grafo originale.

Approssimazione delimitata

La derivazione dei limiti segue una procedura simile a quella dell'algoritmo Bounded Max-Sum descritto nella Sezione 2.3.4. Viene innanzitutto definito il vettore $\mathbf{W} = [W^1, \ldots, W^k]$ come somma dei vettori di pesi $\mathbf{w}_{ij}$ degli archi

tra U_j e x_i che sono stati potati nella fase bounding per ottenere un grafo aciclico. Inoltre, per caratterizzare il limite superiore computato dall'algoritmo, viene definito il concetto di *punto utopia*:

Definizione 10 (Punto utopia)**.** *Il punto utopia* $\mathbf{V}^*$ *di un problema di ottimizzazione multiobiettivo caratterizzato dalla funzione obiettivo* $\mathbf{U}(\mathbf{x}) = \left[U^1(\mathbf{x}), \ldots, U^k(\mathbf{x})\right]^T$ *è dato da:*

$$\mathbf{V}^* = \left[\max_{\mathbf{x}} U^1(\mathbf{x}), \ldots, \max_{\mathbf{x}} U^k(\mathbf{x})\right]$$

Il punto utopia è il vettore dei valori risultanti dall'ottimizzazione di ciascuno dei k DCOP indipendentemente. Chiaramente, per ogni assegnamento Pareto-ottimale $\mathbf{a}^* \in \mathbf{PO}$, $\mathbf{U}(\mathbf{a}^*) \leq \mathbf{V}^*$ è vero per qualsiasi MO-DCOP. Pertanto il punto utopia è un limite superiore del valore di una soluzione Pareto-ottimale di un MO-DCOP. Da questi concetti si può affermare quanto segue:

Teorema 2. *Dato un qualunque MO-DCOP, per qualsiasi assegnamento* $\tilde{\mathbf{a}} \in \widetilde{\mathbf{PO}}$ *computato da B-MOMS, vale il seguente limite:*

$$\mathbf{U}(\tilde{\mathbf{a}}) + \mathbf{W} \geq \mathbf{V}^*$$

Schema della dimostrazione. Il teorema segue direttamente l'estensione dell'algoritmo Bounded Max-Sum per DCOP standard al caso di problemi multiobiettivo. In dettaglio, per ogni obiettivo o $(1 \leq o \leq k)$ di un MO-DCOP, usando l'approccio definito in [23], è facile verificare che vale il limite:

$$U^o(\tilde{\mathbf{a}}) + W^o \geq \max_{\mathbf{x}} U^o(\mathbf{x})$$

che conclude quindi la dimostrazione. □

In maniera simile viene definito l'indice di approssimazione dipendente dal problema $\rho = [\rho_1, \ldots, \rho_k]$ delle soluzioni computate da B-MOMS, dove ogni ρ_i è dato dall'Equazione 3.2, i cui simboli sono definiti nella Sezione 2.3.4:

$$\rho = 1 + \left(\tilde{V}_m + W - \tilde{V}\right)/\tilde{V} \tag{3.2}$$

3.10.3 Multi-Objective Adopt

L'algoritmo B-MOMS [7] è il risultato della generalizzazione dell'algoritmo Max-Sum per problemi multiobiettivo. Un'alternativa a questo algoritmo è la progettazione di un risolutore per MO-DCOP che utilizzi una strategia

di ricerca basata su pseudo-alberi; si tratta di una generalizzazione dell'algoritmo Adopt al caso multiobiettivo. Mentre le rappresentazioni con gli algoritmi GDL si dimostrano efficaci garantendo complessità computazionali accettabili, per la ricerca basata su pseudo-alberi occorre cercare diverse tecniche di potatura dei rami dell'albero di ricerca. Il metodo di ricerca su pseudo-alberi può teoricamente funzionare anche in reti di vincoli contenenti cicli. Per generalizzare gli algoritmi DCOP convenzionali che impiegano potatura dei grafi, è essenzialmente necessario porre dei limiti ai vettori dei costi.
Per affrontare la risoluzione di un MO-DCOP attraverso un risolutore basato su pseudo-alberi, occorre introdurre dei limiti per i vettori multiobiettivo, per poter confrontare anche vettori corrispondenti a soluzioni parziali che non contengono i costi di tutti gli obiettivi.
Il risultato della generalizzazione dell'algoritmo Adopt al caso multiobiettivo è Multi-Objective Adopt [18]. MO-Adopt effettua ricerca su albero e programmazione dinamica parziale. Le computazioni bottom-up della programmazione dinamica parziale e top-down della ricerca su albero utilizzano entrambe dei vettori limitati per le funzioni multiobiettivo. Per operare nel caso multiobiettivo, occorre estendere diverse operazioni comprendenti aggregazione, decomposizione e comparazione delle funzioni obiettivo a vettori limitati.

Limiti inferiori e superiori dei vettori costo

Un assegnamento parziale $\mathcal{A}^-$ non può essere valutato da alcuna funzione costo $f_{i,j}$ tale che $\{(x_i, d_i), (x_j, d_j)\} \not\subset \mathcal{A}^-$ perché diversi valori di variabili sono ignoti. In questo caso, vengono definiti dei valori limite superiore e inferiore per generalizzare valori di costo non definiti. Per un problema DCOP classico di minimizzazione si può definire il limite inferiore come 0 e il limite superiore come $+\infty$. I valori limite generano un confine inferiore e superiore per i valori costo. Usando la notazione $lb(v)$ e $ub(v)$ per rappresentare i confini inferiore e superiore del valore costo v, è ovvio che $lb(v)$ non supera mai $ub(v)$; se v è un valore esatto, allora $lb(v)$ e $ub(v)$ assumono lo stesso valore. La somma di due valori costo è generalizzata considerando i confini; $v + v'$ viene estesa alla coppia di somme $lb(v) + lb(v')$ e $ub(v) + ub(v')$.
I confini sono estesi in modo naturale per i vettori di valori costo. Per ogni valore costo v^k in un vettore, i limiti sono definiti come sopra specificato. I vettori a cui vengono applicati questi confini vengono chiamati *vettori limitati*. Poiché i problemi multiobiettivo hanno più assegnamenti Pareto-ottimali, occorre gestire un insieme di vettori limitati di valori costo.

Confronto di vettori limitati

Il concetto di dominanza tra vettori dev'essere generalizzato per i vettori limitati. Si definiscono le notazioni $v \prec v'$ e $v \preceq v'$ rispettivamente come $ub(v) < lb(v')$ e $ub(v) \leq lb(v')$. Queste notazioni vengono inoltre estese per applicare la comparazione di vettori limitati. La notazione appena introdotta consente la definizione di non-dominanza di vettori limitati.

Definizione 11 (Non-dominanza)**.** $\mathbf{v}^+$ *è non dominato se e solo se non esiste un altro vettore* $\mathbf{v}$ *tale che* $\mathbf{v} \preceq \mathbf{v}^+$*, e* $v^k \prec v^{+k}$ *per almeno una funzione costo. Altrimenti il vettore* $\mathbf{v}^+$ *è dominato.*

Aggregazione di insiemi di vettori limitati

I vettori di valori costo vengono aggregati durante la computazione della soluzione. Poiché vi sono molteplici assegnamenti non-dominati e vettori costo per ogni problema parziale, l'aggregazione è definita per insiemi di vettori limitati. Essa consiste di due operazioni. Nella prima, viene computato un insieme di vettori che contiene somme per ogni combinazione di vettori e per ogni insieme di vettori.

Definizione 12 (Somma)**.** *La somma degli insiemi di vettori* V *e* V' *viene denotata con* $V \oplus V'$*. La somma di* $V_0 \oplus \cdots \oplus V_h$ *è definita come segue:*

$$V_0 \oplus \cdots \oplus V_h = \bigcup_{T \in V_0 \times \cdots \times V_h} \left\{ \sum_{\mathbf{v} contenuto\ in\ T} \mathbf{v} \right\}$$

Il risultato della somma di cui sopra potrebbe contenere vettori ridondanti o dominati. Nella seconda operazione, viene applicato un filtro per conservare i soli vettori non-dominati.

Definizione 13 (Filtraggio)**.** *Il filtro* $Flt(V)$ *elimina un vettore* $\mathbf{v}$ *da un insieme* V *di vettori per evitare i casi seguenti:*

1. $\exists \mathbf{v}' \in V, \forall v' \in \mathbf{v}', \forall v \in \mathbf{v}, lb(v') \leq lb(v) \wedge ub(v) \leq ub(v')$
2. $\exists \mathbf{v}' \in V, \mathbf{v}' \preceq \mathbf{v}$

Dove $\mathbf{v}'$ *rappresenta uno degli altri vettori in* V*.*

Risoluzione basata su pseudo-alberi

Con l'assunzione che tutti i vincoli siano binari, alla base di questo metodo, la costruzione dello pseudo-albero definisce una struttura che implica un ordinamento parziale sulle variabili. Una variabile x_i ha un singolo nodo padre

pr_i e un insieme di nodi figli Ch_i. La computazione dell'agente i è basata sulla soluzione parziale s_i (chiamata *contesto*) degli antenati Ast_i della variabile x_i nello pseudo-albero. L'insieme Nb_i^u denota l'insieme parziale degli antenati di x_i che sono direttamente legati ad x_i da un vincolo.
Il costo ottimale per l'assegnamento parziale $g_i^*(s_i)$ è definito ricorsivamente dalle seguenti regole:

$$g_i^*(s_i) = \min_{d \in D_i} g_i(s_i \cup \{(x_i, d)\})$$

$$g_i(s_i \cup \{(x_i, d)\}) = \delta_i(s_i \cup \{(x_i, d)\}) + \sum_{j \in Ch_i} g_j^*(s_j) \text{ t.c. } s_j \subseteq (s_i \cup \{(x_i, d)\})$$

$$\delta_i(s_i \cup \{(x_i, d)\}) = \sum_{(x_j, d_j) \in s_i, j \in Nb_i^u} f_{i,j}(d, d_j)$$

In una determinata computazione, alcuni costi potrebbero non essere ancora ricevuti; in questo caso vengono usati i valori dei limiti superiore e inferiore come valori di default.
Per computare i costi, un agente i memorizza le seguenti informazioni:

- s_i: contesto corrente ricevuto dal padre di i.
- d_i: assegnamento corrente della propria variabile. $\{(x_i, d_i)\}$ è parte del contesto dei figli di i.
- $g_{j,d}^*$: valore di costo per $s_i \cup \{(x_i, d)\}$ ricevuto dal figlio j.

Nel processo di ricerca gli agenti scambiano due tipi di messaggi:

- $(\text{CONTEXT}_{i \to j}, s_i \cup \{(x_i, d_i)\})$: trasferisce il contesto corrente e le relative informazioni dal nodo i al figlio j.
- $(\text{COST}_{i \leftarrow j}, s_j, g_j^*(s_j))$: trasferisce il costo $g_j^*(s_j)$ per il contesto corrente s_j dal nodo j al padre i.

In una determinata computazione, il costo $g_j^*(s_j)$ è rappresentato dai limiti inferiore $lb(g_j^*(s_j))$ e superiore $lb(g_j^*(s_j))$. Pertanto la notazione $g_j^*(s_j)$ rappresenta implicitamente due limiti di un costo.
Nella ricerca su albero ogni nodo non-foglia i propaga il suo contesto inviando messaggi CONTEXT con un approccio top-down. Ogni nodo non-radice j invia il proprio costo tramite messaggi COST al padre. I limiti superiore e inferiore dei costi per ogni nodo vengono aggiornati durante il processo

di ricerca; quando questi coincidono, l'agente conclude che non esiste una soluzione migliore per l'assegnamento corrente. L'algoritmo termina quando il nodo radice conclude che il proprio assegnamento $\{(x_r, d)\}$ è globalmente ottimale; il nodo radice invia quindi l'assegnamento ottimale ai figli con messaggi CONTEXT attivando un flag specifico. Gli altri agenti terminano di conseguenza con uno stile top-down.
Per problemi multiobiettivo, la computazione dei costi viene generalizzata per il calcolo di vettori di costi. Il vettore dei costi ottimale per un contesto s_i viene calcolato ricorsivamente come nel caso monobiettivo:

$$G_i^*(s_i) = Flt\left(\bigcup_{d \in D_i} G_i(s_i \cup \{(x_i, d)\})\right)$$

$$G_i(s_i \cup \{(x_i, d)\}) = Flt\left(\delta_i(s_i \cup \{(x_i, d)\}) \oplus \bigoplus_{j \in Ch_i} G_j^*(s_j)\right) \text{ t.c. } s_j \subseteq (s_i \cup \{(x_i, d)\})$$

$$\delta_i(s_i \cup \{(x_i, d)\}) = \sum_{(x_j, d_j) \in s_i, j \in Nb_i^u} \mathbf{f}_{i,j}(d, d_j)$$

L'algoritmo MO-Adopt raggiunge sempre la terminazione in quanto la propagazione del contesto dei vari nodi dello pseudo-albero porta al soddisfacimento della condizione di terminazione nel nodo radice.

Capitolo 4

Modellazione e risoluzione di sistemi idrici attraverso MO-DCOP

4.1 Formulazione MO-DCOP del problema

L'analisi effettuata nel capitolo precedente ha portato alla conclusione che il framework DCOP non è sufficientemente espressivo per rappresentare le caratteristiche di un modello di sistema idrico distribuito, che è a tutti gli effetti un problema di ottimizzazione multiobiettivo.
Il framework DCOP standard, originariamente progettato per problemi monobiettivo, è stato generalizzato al caso multiobiettivo nello studio [7], in cui viene proposta una formulazione generale di un problema di ottimizzazione di vincoli distribuiti multiobiettivo MO-DCOP, ed il primo algoritmo in grado di risolverlo. Lo studio [18] adotta la formulazione MO-DCOP e presenta un algoritmo risolutivo risultato della generalizzazione dell'algoritmo Adopt al caso multiobiettivo.
I sistemi idrici oggetto di questo studio sono quindi modellabili attraverso MO-DCOP, essendo problemi distribuiti di ottimizzazione multiobiettivo. Un generico problema di ottimizzazione multiobiettivo è composto da una serie di variabili legate da alcuni vincoli i cui costi sono rappresentati da diverse funzioni obiettivo; nel caso dei sistemi idrici, la formulazione degli obiettivi è strettamente legata agli agenti (e quindi alle variabili) appartenenti al sistema. In un sistema idrico, ogni agente persegue un proprio obiettivo locale, da cui deriva la necessità di formulare una funzione obiettivo per ciascun agente; inoltre, la (eventuale) violazione dei vincoli si traduce in un costo globale per il sistema, pertanto è necessario definire un'ulteriore

funzione obiettivo (globale).
Il modello di un sistema idrico si compone quindi di:

- n agenti $\{1, 2, \ldots, n\}$.
- n variabili $\{x_1, x_2, \ldots, x_n\}$, dove l'agente i controlla la variabile x_i.
- k vincoli di tipo hard C_H o di tipo soft C_S, $C = C_H \cup C_S$.
- n funzioni obiettivo locali $F_{Oi}(x_i)$, una per ciascun agente i.
- una funzione obiettivo globale $F_G = \sum C_j$, risultato dell'aggregazione dei costi di ogni vincolo ($C_j \in C$).

Un sistema composto da n agenti dev'essere modellato come un problema che coinvolge $n + 1$ obiettivi.
Poiché in un DCOP (e di conseguenza anche in un MO-DCOP) è possibile rappresentare solamente delle funzioni costo relative ai vincoli, non è possibile formulare esplicitamente le funzioni obiettivo di ogni agente come tali, ma occorre rappresentarle attraverso funzioni costo di qualche vincolo. A questo scopo è sufficiente introdurre nella modellazione un vincolo unario per ogni variabile x_i la cui funzione costo rappresenta la funzione obiettivo locale dell'agente i. Nella formulazione MO-DCOP si avranno pertanto alcuni vincoli la cui funzione costo rappresenta l'obiettivo di un singolo agente, e altri vincoli la cui funzione costo rappresenta effettivamente il costo da pagare globalmente per la violazione degli stessi.
Si deduce che ogni vincolo in questa formulazione rappresenta un costo relativo ad un unico obiettivo (locale di un agente o globale). In un generico MO-DCOP, ogni vincolo rappresenta una funzione di costo multiobiettivo, in quanto la specifica prevede che ogni vincolo possa fornire un contributo per più di un obiettivo, ed è rappresentato quindi da un vettore di funzioni costo. Nel caso di studio di sistemi idrici non occorre definire un vettore di funzioni costo per ogni vincolo, poiché questo vettore sarebbe composto da n valori pari a 0 e da un unico valore significativo per l'obiettivo a cui il vincolo corrisponde. È quindi sufficiente rappresentare la funzione costo del vincolo con un unico valore (significativo), specificando l'obiettivo corrispondente. Nel processo di risoluzione del problema i valori di ogni singola funzione verranno inseriti in vettori di funzioni costo (multiobiettivo) nella posizione dell'obiettivo corrispondente, permettendo così il confronto tra i diversi vettori e mantenendo separati i contributi relativi ad obiettivi diversi.

La formalizzazione del modello MO-DCOP di un sistema idrico segue la formalizzazione generale di un MO-DCOP introdotta nella Sezione 3.10.1, e consiste nell'ottimizzazione del vettore delle funzioni obiettivo:

$$\mathbf{F}(\mathbf{x}) = \left[F^1(\mathbf{x}), \ldots, F^n(\mathbf{x}), F^{n+1}(\mathbf{x})\right]^T$$

In cui $\mathbf{x} = \{x_1, x_2, \ldots, x_n\}$ è l'insieme delle variabili; il numero degli obiettivi è pari a $n + 1$. L'obiettivo n+1-esimo è denominato obiettivo G. Una funzione di vincolo multiobiettivo è definita come:

$$\mathbf{F}_i(\mathbf{x}_i) = \left[F_i^1(\mathbf{x}_i), \ldots, F_i^{n+1}(\mathbf{x}_i)\right]^T$$

Come detto in precedenza, ogni vincolo in un sistema idrico rappresenta un unico costo, relativo all'obiettivo globale (G), mentre le funzioni obiettivo di ogni agente vengono modellate attraverso vincoli unari, anch'essi rappresentanti un'unico costo corrispondente all'obiettivo dell'agente. Pertanto il vettore dei costi di ogni funzione di vincolo multiobiettivo $\mathbf{F}_i$ è composto da n valori nulli ed un solo valore significativo. La funzione obiettivo F_{Oi} dell'i-esimo agente è rappresentata dalla funzione di vincolo multiobiettivo:

$$\mathbf{F}_{Oi}(\mathbf{x}_i) = \left[0, \ldots, F_{Oi}^i(x_i), \ldots, 0\right]^T$$

Mentre la funzione di vincolo multiobiettivo relativa ad un vincolo è:

$$\mathbf{F}_i(\mathbf{x}_i) = \left[0, \ldots, 0, F_i^{n+1}(\mathbf{x}_i)\right]^T$$

4.1.1 Formulazione MO-DCOP del modello di riferimento

Viene qui proposta una formulazione MO-DCOP del modello di riferimento introdotto in [26] e descritto nella Sezione 3.2.1, per il quale vengono specificate delle funzioni utilità anzichè funzioni costo, trattandosi di un problema di massimizzazione. Gli obiettivi sono 7, uno per ciascun agente più l'obiettivo globale G. Nella Tabella 4.1 vengono elencate le funzioni utilità dei vincoli unari che rappresentano le funzioni obiettivo dei singoli agenti, la cui formulazione è banale.

Per la rappresentazione dei vincoli "reali" vengono adottate le linee guida introdotte nella descrizione di vincoli soft e hard presentate nel capitolo precedente; per ogni vincolo soft è definito un costo C_k (nello specifico un'utilità negativa, $C_k \leq 0$) da pagare in caso di violazione del vincolo. In questo modello i vincoli rappresentati dalle funzioni F_4 e F_6 riportati nella Tabella 4.2 sono ridondanti e hanno lo stesso significato dei vincoli F_9 e F_{11} per via delle uguaglianze imposte dai vincoli fisici F_8 e F_{10}. Nella Tabella 4.2

Funzione obiettivo locale	Funzione costo
$\max_{x_1} f_1(x_1) = a_1x_1^2 + b_1x_1 + c_1$	$F_{O1}(x_1) = a_1x_1^2 + b_1x_1 + c_1$
$\max_{x_2} f_2(x_2) = a_2x_2^2 + b_2x_2 + c_2$	$F_{O2}(x_2) = a_2x_2^2 + b_2x_2 + c_2$
$\max_{x_3} f_3(x_3) = a_3x_3^2 + b_3x_3 + c_3$	$F_{O3}(x_3) = a_3x_3^2 + b_3x_3 + c_3$
$\max_{x_4} f_4(x_4) = a_4x_4^2 + b_4x_4 + c_4$	$F_{O4}(x_4) = a_4x_4^2 + b_4x_4 + c_4$
$\max_{x_5} f_5(x_5) = a_5x_5^2 + b_5x_5 + c_5$	$F_{O5}(x_5) = a_5x_5^2 + b_5x_5 + c_5$
$\max_{x_6} f_6(x_6) = a_6x_6^2 + b_6x_6 + c_6$	$F_{O6}(x_6) = a_6x_6^2 + b_6x_6 + c_6$

Tabella 4.1: Formulazione delle funzioni costo dei vincoli unari rappresentanti le funzioni obiettivo degli agenti nel sistema descritto nella Sezione 3.2.1

sono riportati tutti i vincoli per coerenza con la formulazione originale del problema.

Disequazione	Tipo vincolo	Funzione costo
$\alpha_1 - x_1 \leq 0$	Soft	$F_1(x_1) = \begin{cases} 0 & \alpha_1 - x_1 \leq 0 \\ C_1 & \text{altrimenti} \end{cases}$
$\alpha_2 - Q_1 - x_1 \leq 0$	Soft	$F_2(x_1) = \begin{cases} 0 & \alpha_2 - Q_1 - x_1 \leq 0 \\ C_2 & \text{altrimenti} \end{cases}$
$\alpha_3 - x_4 \leq 0$	Soft	$F_3(x_4) = \begin{cases} 0 & \alpha_3 - x_4 \leq 0 \\ C_3 & \text{altrimenti} \end{cases}$
$\alpha_4 - Q_2 + x_4 \leq 0$	Soft	$F_4(x_4) = \begin{cases} 0 & \alpha_4 - Q_2 + x_4 \leq 0 \\ C_4 & \text{altrimenti} \end{cases}$
$\alpha_5 - x_6 \leq 0$	Soft	$F_5(x_6) = \begin{cases} 0 & \alpha_5 - x_6 \leq 0 \\ C_5 & \text{altrimenti} \end{cases}$
$\alpha_6 - x_2 - x_3 + x_6 \leq 0$	Soft	$F_6(x_2, x_3, x_6) = \begin{cases} 0 & \alpha_6 - x_2 - x_3 + x_6 \leq 0 \\ C_6 & \text{altrimenti} \end{cases}$
$x_2 - S - Q_1 + x_1 \leq 0$	Hard	$F_7(x_1, x_2) = \begin{cases} 0 & x_2 - S - Q_1 + x_1 \leq 0 \\ -\infty & \text{altrimenti} \end{cases}$
$x_3 = Q_2 - x_4$	Hard	$F_8(x_3, x_4) = \begin{cases} 0 & x_3 = Q_2 - x_4 \\ -\infty & \text{altrimenti} \end{cases}$
$\alpha_4 - x_3 \leq 0$	Soft	$F_9(x_3) = \begin{cases} \alpha_4 - x_3 \leq 0 \\ C_9 \quad \text{altrimenti} \end{cases}$
$x_5 = x_2 + x_3 - x_6$	Hard	$F_{10}(x_2, x_3, x_5, x_6) = \begin{cases} 0 & x_5 = x_2 + x_3 - x_6 \\ -\infty & \text{altrimenti} \end{cases}$
$\alpha_6 - x_5 \leq 0$	Soft	$F_{11}(x_5) = \begin{cases} 0 & \alpha_6 - x_5 \leq 0 \\ C_{11} & \text{altrimenti} \end{cases}$

Tabella 4.2: Formulazione delle funzioni costo dei vincoli del sistema descritto nella Sezione 3.2.1

Nella Figura 4.1 è rappresentato il grafo fattorizzato del modello MO-DCOP appena descritto, nel quale non sono stati riportati i vincoli ridondanti F_4 e F_6, e per ogni vincolo viene omessa la classica rappresentazione delle funzioni costo relative a ciascun obiettivo (delle quali solo una è signi-

ficativa), indicando solamente il nome del vincolo.
L'obiettivo relativo all'unica utilità non nulla dei vincoli F_{Oi}, con $i \in [1, 6]$, è l'obiettivo i, mentre per i vincoli F_k, con $k \in [1, 11]$, l'unica utilità non nulla corrisponde all'obiettivo G.

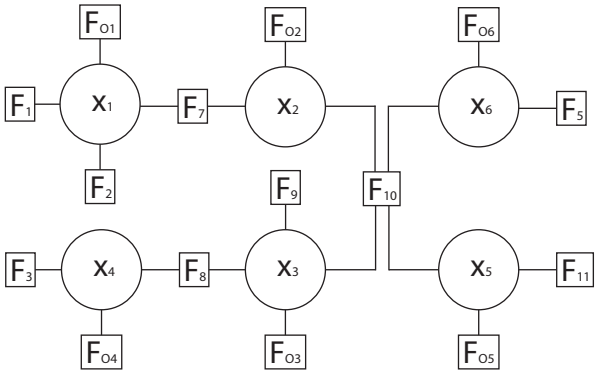

Figura 4.1: Grafo fattorizzato della formulazione MO-DCOP del modello di riferimento

4.2 Risoluzione del modello MO-DCOP

Avendo formulato il problema attraverso il formalismo MO-DCOP, la risoluzione dello stesso può essere affidata a uno degli algoritmi precedentemente introdotti nelle Sezioni 3.10.2 e 3.10.3, progettati appositamente per un MO-DCOP: B-MOMS o MO-Adopt. Entrambi gli algoritmi, tuttavia, forniscono come risultato un'unica soluzione ottimale al problema, in base alle specifiche implementative dell'algoritmo. Nel caso di studio di sistemi idrici, l'interesse principale si focalizza sull'ottenimento della frontiera di Pareto di tutte le soluzioni ottimali, per poterle confrontare e soprattutto per valutare come le decisioni dei diversi agenti indirizzano la scelta della soluzione verso un particolare punto sulla frontiera (corrispondente ad una particolare soluzione ottimale) piuttosto che altri punti.
Occorre pertanto entrare in dettaglio nell'implementazione degli algoritmi disponibili, e studiare una via per ottenere l'intera frontiera delle soluzioni Pareto-ottimali anzichè una singola soluzione.
L'idea alla base dell'algoritmo B-MOMS, descritto nella Sezione 3.10.2, e nello specifico dell'algoritmo Max-Sum che generalizza, è di elaborare le informazioni distribuite nel modello al fine di identificare le condizioni che corrispondono ad una situazione ottimale. Questo processo consiste nello scambio di messaggi tra i nodi-funzione e i nodi-variabile del grafo fattorizzato in cui viene codificato il problema. Tali messaggi rappresentano le

massime utilità aggregate tra i componenti del grafo ottenute rimuovendo un determinato ramo di dipendenza tra un nodo-funzione e un nodo-variabile. Il processo di scambio iterativo di questi messaggi tra i nodi, con la relativa propagazione dei contenuti dei messaggi nell'intero grafo, corrisponde ad una procedura in cui i vari nodi elaborano le informazioni al fine di scartare le soluzioni non ottimali e propagare solo quelle ottimali. Per ogni variabile del grafo viene calcolata una funzione marginale, il cui contenuto rappresenta le utilità (per ciascun obiettivo) migliori ottenibili per ogni valore del dominio della variabile stessa. Alla fine di questo processo, ogni agente possiede una mappa (la funzione marginale) delle soluzioni migliori ottenibili per ogni diverso assegnamento locale possibile. Nell'algoritmo B-MOMS, le informazioni contenute nelle funzioni marginali vengono utilizzate nella fase value-propagation per scegliere una soluzione ottimale sulla base di una determinata funzione sociale. Se anzichè effettuare la fase value-propagation si usassero le informazioni contenute nelle marginali senza operare scelte specifiche di assegnamento delle variabili, si potrebbe estrarre dalle marginali stesse la frontiera di Pareto di tutte le soluzioni ottimali. Riassumendo, è possibile modificare l'algoritmo B-MOMS per ottenere la frontiera di Pareto delle soluzioni ottimali sostituendo la fase value-propagation con una fase di "assemblaggio" della frontiera.
Diversamente dall'approccio dell'algoritmo B-MOMS, l'algoritmo MO-Adopt, descritto nella Sezione 3.10.3, non segue una strategia in cui si cercano le soluzioni ottimali tra le quali sceglierne una specifica, ma opera componendo delle soluzioni parziali, in cui solo alcuni agenti hanno assegnato un valore alla propria variabile, fino a trovare un assegnamento globale a tutte le variabili corrispondente ad una soluzione ottimale. È pertanto complesso ideare un meccanismo per elaborare tutte le soluzioni ottimali attraverso MO-Adopt; l'idea più immediata sarebbe l'esecuzione iterativa dell'algoritmo tenendo conto delle soluzioni ottimali trovate in precedenza (inserendole ad esempio nella forma di vincoli su tutte le variabili e rendendole quindi inammissibili nella iterazione corrente) fino al raggiungimento di un'iterazione in cui l'algoritmo non trova più alcuna soluzione e pertanto dovrebbe aver elaborato l'intera frontiera di Pareto. Questa via è sconveniente oltre che computazionalmente onerosa; dato che non è possibile ottenere un limite a priori sulla dimensione della frontiera di Pareto di un problema, ad eccezione del prodotto cartesiano dei domini di tutte le variabili, non si saprebbe quante volte sarebbe necessario reiterare l'esecuzione di MO-Adopt.
Si è pertanto deciso di implementare una versione modificata dell'algoritmo B-MOMS, sostituendo la fase value-propagation con una fase di costruzione (estrazione) della frontiera di Pareto. In questa implementazione non è stata

inclusa la fase iniziale di bounding, che riduce una formulazione MO-DCOP il cui grafo dei vincoli è ciclico ad una formulazione approssimata con grafo aciclico, in quanto l'attenzione di questo lavoro si focalizza su modellazione dei sistemi ed elaborazione della frontiera delle soluzioni ottimali; inoltre nel contesto dei sistemi idrici la presenza di cicli (nella rete dei vincoli) è un'eventualità piuttosto rara.

4.3 Considerazioni sulla funzione marginale

Nell'algoritmo Max-Sum, ogni nodo-variabile x_j computa ad ogni iterazione una funzione marginale $z_j(x_j)$ come somma dei messaggi r ricevuti da tutti i nodi funzione collegati. L'informazione contenuta nella funzione marginale rappresenta l'insieme delle soluzioni migliori ottenibili per ogni possibile assegnamento della variabile x_j, come mostrato nell'Equazione 4.1. La funzione marginale è costruita come una tabella in cui ogni riga corrisponde ad un determinato assegnamento della variabile x_j, e per ciascuna riga sono elencate le soluzioni (localmente) ottimali corrispondenti.

$$z_j(x_j) = \sum_{i \in M(j)} r_{i \to j}(x_j) = \arg\max_{\mathbf{x} \setminus x_j} \sum_{i=1}^{N} U_i(\mathbf{x}_i) \tag{4.1}$$

Si dimostra il seguente teorema, che verrà usato per l'elaborazione della frontiera di Pareto nella fase implementativa.

Teorema 3. *Una soluzione è Pareto-ottimale se e solo se è contenuta in tutte le funzioni marginali, quando queste hanno raggiunto una situazione invariante e non cambiano contenuto in successive iterazioni dell'algoritmo Max-Sum per un grafo dei vincoli aciclico.*

Dimostrazione. Dalla definizione di Pareto-ottimalità riportata nella Sezione 3.7.1 è noto che una soluzione è Pareto-ottimale se e solo se il relativo vettore dei costi non è dominato da nessun altro vettore.
Un nodo-variabile computa la propria funzione marginale sommando i messaggi r ricevuti in una data iterazione; la somma dei vettori contenuti nei messaggi r genera dei vettori-soluzione. Poiché sui vettori contenuti in un messaggio r viene operata una massimizzazione da parte del nodo-funzione che lo computa, ogni messaggio r contiene solo vettori non-dominati. Ogni messaggio r, quindi ogni vettore non-dominato, viene peraltro propagato entro un numero finito di iterazioni verso tutti i nodi-variabile del grafo attraverso i messaggi q.

Il vettore dei costi di una soluzione Pareto-ottimale corrisponde alla somma di vettori non-dominati, contenuti in messaggi r che vengono ricevuti da tutti i nodi-variabile, pertanto tale vettore viene computato da tutti i nodi-variabile ed inserito in tutte le funzioni marginali.
Ogni nodo-variabile computa ed inserisce nella propria funzione marginale i vettori-soluzione non-dominati per ogni assegnamento locale della variabile. Se un vettore è contenuto in tutte le marginali, significa che esiste un assegnamento globale (in cui a ogni variabile è assegnato un valore) il cui vettore dei costi è non-dominato, pertanto questo vettore corrisponde ad una soluzione Pareto-ottimale. □

Corollario 1. *La frontiera di Pareto delle soluzioni ottimali di un MO-DCOP con grafo dei vincoli aciclico è data dall'intersezione tra gli insiemi delle soluzioni contenute in ogni singola marginale.*

In condizioni particolari, i vettori contenuti nella riga della marginale $z_j(x_j)$ corrispondente al valore $\overline{x_j}$ potrebbero essere dominati dai vettori contenuti in tutte le altre righe della stessa marginale. Questo significa che con l'assegnamento $x_j = \overline{x_j}$ si ottengono soluzioni sub-ottimali rispetto alle soluzioni ottenibili con un assegnamento diverso. Si potrebbe quindi pensare di eliminare la riga in questione dalla marginale per non considerarla durante l'elaborazione della frontiera. Tuttavia, l'identificazione ed eliminazione di queste soluzioni sub-ottimali (con conseguente consumo computazionale) non è necessaria, in quanto la propagazione dei messaggi tra i nodi del grafo durante le varie iterazioni dell'algoritmo Max-Sum influenza direttamente il calcolo delle marginali, e nello specifico la massimizzazione operata nel calcolo dei messaggi r comporta automaticamente l'eliminazione delle soluzioni non-ottimali da parte degli altri nodi; queste soluzioni possono tuttalpiù permanere nella funzione marginale del nodo-variabile in cui sono state originate. Si deduce che le soluzioni sub-ottimali contenute in una riga di una marginale non sono contenute nelle altre marginali, pertanto non appartengono all'intersezione tra le soluzioni di tutte le marginali, e non verranno quindi inserite nella frontiera di Pareto.

4.4 Implementazione dell'algoritmo risolutivo MO-Max-Sum

L'algoritmo Max-Sum, generalizzato al caso multiobiettivo, produce come risultato delle funzioni marginali, che contengono tutte le informazioni sufficienti per l'elaborazione della frontiera di Pareto delle soluzioni ottimali di

un dato problema. Pertanto si è deciso di focalizzare la fase implementativa sulla realizzazione dell'algoritmo MO-Max-Sum, ovvero dell'estensione multiobiettivo dell'algoritmo Max-Sum. L'implementazione segue le linee semantiche dell'algoritmo Max-Sum dettagliate in [23], adattando le strutture dati e le informazioni elaborate e propagate attraverso i vari messaggi al caso multiobiettivo, passando quindi da funzioni costo scalari a vettori di funzioni costo. L'esecuzione distribuita dell'algoritmo si basa sull'assunzione che ogni agente controlli una singola variabile.
Le fasi principali dell'algoritmo MO-Max-Sum implementato possono essere identificate come:

- Lettura della specifica del problema e costruzione del grafo fattorizzato.
- Elaborazione e scambio iterativo dei messaggi tra i vari nodi, con calcolo iterativo delle funzioni marginali.
- Raccolta delle informazioni dalle marginali ed elaborazione della frontiera di Pareto.

4.4.1 Specifica e codifica del problema

Poiché l'algoritmo MO-Max-Sum lavora sulla base di un grafo bipartito fattorizzato, occorre fornire all'applicazione una formulazione "leggibile" del problema, che possa essere facilmente interpretata e codificata in un grafo fattorizzato. Questa specifica consiste in un elenco di variabili e funzioni.
Per ogni variabile viene specificato:

- un identificativo univoco (x_j),
- un dominio discreto e finito, espresso come elenco di valori.

Per ogni funzione viene specificato:

- un identificativo univoco (F_i),
- lo scope, ovvero l'insieme delle variabili da cui dipende la funzione,
- l'obiettivo per il quale la funzione è significativa,
- l'elenco dei valori assunti in funzione di ogni possibile tupla delle variabili dello scope.

Come argomentato nel capitolo precedente, nel caso di studio di sistemi idrici, ogni vincolo (e pertanto ogni funzione costo) genera un contributo relativo ad un singolo obiettivo; pertanto nella formulazione delle funzioni

non occorre specificare per ogni tupla delle variabili dello scope un vettore di costi, ma è sufficiente specificare l'unico valore significativo e l'obiettivo corrispondente.
Il grafo fattorizzato è costituito da un insieme di nodi-variabile e nodi-funzione, che vengono istanziati in base ai dati della specifica del problema. Gli archi che collegano i vari nodi vengono creati in riferimento allo scope di ciascuna funzione definita: se la variabile x_j appartiene allo scope della funzione F_i allora nel grafo fattorizzato esiste l'arco x_j–F_i. La struttura del grafo fattorizzato è implicitamente codificata nella specifica di ogni nodo, che possiede una lista dei nodi vicini.

4.4.2 Algoritmo MO-Max-Sum

Una volta codificato il problema attraverso un grafo fattorizzato, può iniziare l'esecuzione vera e propria dell'algoritmo MO-Max-Sum. La computazione dell'algoritmo è strettamente legata alla struttura del grafo fattorizzato, in quanto l'algoritmo si basa sullo scambio iterativo di messaggi tra nodi-funzione e nodi-variabile. I nodi-variabile computano ed inviano messaggi q, mentre i nodi-funzione elaborano e inviano messaggi r.
Nella prima iterazione tutti i messaggi q sono inizializzati a zero, pertanto i messaggi r inviati da una funzione F_i a una variabile x_j sono una massimizzazione della funzione stessa rispetto a tutte le variabili nello scope di F_i tranne x_j. Dall'iterazione successiva ogni messaggio $q_{j\to i}$ viene calcolato sommando i messaggi r ricevuti nell'iterazione precedente da tutti i nodi-funzione collegati alla variabile x_j tranne il nodo-funzione F_i, al quale si sta inviando il messaggio q, interpretando l'Equazione 4.2. Ad ogni iterazione, i nodi-funzione attendono la ricezione dei messaggi q nell'iterazione corrente, il cui contenuto è necessario per l'elaborazione dei messaggi r come definito nell'Equazione 4.3. Un messaggio inviato nell'iterazione t viene denotato con r^t oppure q^t.
L'elaborazione del messaggio q^t da parte di un nodo-variabile richiede l'avvenuta ricezione dei messaggi r^{t-1} dell'iterazione precedente, mentre l'elaborazione dei messaggi r^t da parte di un nodo-funzione richiede l'avvenuta ricezione dei messaggi q^t dell'iterazione corrente. L'ordine con cui vengono inviati e ricevuti i messaggi non è predefinito, se un nodo non ha ancora ricevuto i messaggi necessari per l'elaborazione del proprio messaggio, si pone in attesa fino all'avvenuta ricezione.

L'elaborazione del contenuto dei messaggi $q^t_{j\to i}(x_j)$ e $r^t_{i\to j}(x_j)$ tra un generico nodo-variabile x_j e un generico nodo-funzione F_i, rappresentati nella

Figura 4.2a), avviene seguendo le equazioni 4.2 e 4.3, in cui $M(j)$ e $N(i)$ denotano rispettivamente i vicini di un nodo-variabile e di un nodo-funzione.

$$q^t_{j\to i}(x_j) = \sum_{k\in M(j)\setminus i} r^{t-1}_{k\to j}(x_j) \tag{4.2}$$

$$r^t_{i\to j}(x_j) = \max_{\mathbf{x}_i\setminus\, x_j} \left(F_i(\mathbf{x}_i) + \sum_{k\in N(i)\setminus j} q^t_{k\to i}(x_k) \right) \tag{4.3}$$

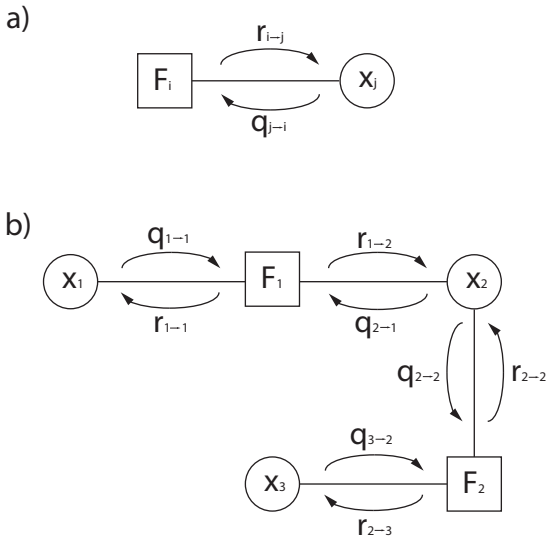

Figura 4.2: Scambio di messaggi tra nodi nel grafo fattorizzato

Nel grafo fattorizzato in Figura 4.2b), il nodo-funzione F_1, per elaborare ed inviare il messaggio $r^t_{1\to 2}$, deve attendere la ricezione del messaggio $q^t_{1\to 1}$, poiché il messaggio $r^t_{1\to 2}$ è calcolato come:

$$r^t_{1\to 2}(x_2) = \max_{x_1} \left(F_1(x_1, x_2) + q^t_{1\to 1}(x_1) \right)$$

Il nodo-variabile x_2, per elaborare il messaggio $q^t_{2\to 2}$, deve attendere la ricezione del messaggio $r^{t-1}_{1\to 2}$ dell'iterazione precedente in quanto il messaggio $q^t_{2\to 2}$ è calcolato come:

$$q^t_{2\to 2}(x_2) = r^{t-1}_{1\to 2}(x_2)$$

Lo stesso discorso vale per tutte le altre dipendenze nel grafo dei vincoli. In generale, un nodo-funzione con arità a, per poter computare un messaggio r, necessita di $a-1$ messaggi q, dei quali deve attendere la ricezione. Si intuisce che un nodo-funzione unario non considera alcun messaggio q, in

quanto riceve un unico messaggio q proprio dalla variabile a cui invia l'unico messaggio r (nell'Equazione 4.3, $N(i)\backslash j = \{\}$, pertanto la sommatoria non considera alcun elemento q). Ne consegue che il messaggio r inviato da un nodo-funzione unario rappresenta sempre la massimizzazione della funzione rispetto all'unica variabile dello scope, significato che permane anche nelle iterazioni successive; pertanto il contenuto del messaggio r di un nodo-funzione unario è invariante e può essere computato un'unica volta. L'implementazione dell'algoritmo MO-Max-Sum proposta in questa tesi identifica i nodi-funzione unari e ne calcola i messaggi r un'unica volta, durante la prima iterazione.
L'algoritmo è in grado di valutare distintamente vincoli hard e vincoli soft. È possibile identificare le soluzioni inammissibili, ovvero che violano un vincolo hard, definendo una soglia S_H sufficientemente negativa (teoricamente $-\infty$). Quando un nodo-funzione elabora il contenuto di un messaggio r destinato ad una variabile x_j, computa per ogni riga del messaggio una serie di vettori composti dai costi (utilità) relativi a ciascun obiettivo, dove ad ogni riga corrisponde un determinato valore del dominio della variabile x_j. Un nodo-funzione corrispondente ad un vincolo hard valuta che questi costi non siano inferiori alla soglia S_H, e scarta i vettori che non soddisfano questa condizione. L'identificazione delle soluzioni inammissibili avviene pertanto già nella prima iterazione dell'algoritmo, in cui ogni nodo-funzione che rappresenta un vincolo hard scarta i vettori contenenti costi inferiori alla soglia S_H.
Quando un nodo-variabile ha ricevuto nell'iterazione t tutti i messaggi r dai nodi-funzione a cui è collegato, l'iterazione t termina ed il nodo-variabile calcola la propria marginale. Se il contenuto della marginale calcolata nell'iterazione t è identico al contenuto della stessa marginale calcolata nelle k iterazioni precedenti (con $k \in \mathbb{N}$ preimpostato), la marginale viene considerata invariante, e il nodo-variabile viene disattivato, in quanto, essendo giunto ad una situazione stazionaria, il contenuto dei messaggi che riceve ed invia nelle iterazioni successive rimarrà identico a quello dell'iterazione appena terminata. I nodi-funzione collegati ad un nodo-variabile "disattivato" non attendono più la ricezione di messaggi da quel nodo, ma ricopiano il contenuto dell'ultimo messaggio ricevuto dal nodo-variabile prima che si disattivasse. Questo meccanismo permette di eliminare computazioni inutili in cui alcuni nodi perderebbero tempo ad elaborare messaggi il cui contenuto è noto poiché identico a quello già precedentemente elaborato.
L'algoritmo MO-Max-Sum termina quando tutti i nodi-variabile sono disattivati, e di conseguenza tutte le funzioni marginali sono invarianti. Questa condizione corrisponde all'avvenuta propagazione dei messaggi nell'intero grafo,

per la quale il numero massimo di iterazioni dell'algoritmo richiesto è pari alla distanza massima nel cammino tra due nodi nel grafo, che corrisponde al doppio della profondità dell'albero con profondità minima costruito partendo dal grafo (un grafo aciclico come quello considerato in questo lavoro è già di per sè un albero). Nel caso di un grafo lineare, che assume la forma di una catena di nodi-variabile e nodi-funzione, il numero di iterazioni equivale alla lunghezza della catena. L'implementazione di MO-Max-Sum prevede la definizione esterna di un numero massimo di iterazioni al termine del quale l'algoritmo, se non è terminato in un'iterazione precedente, viene interrotto.

4.4.3 Elaborazione della frontiera di Pareto

Quando l'algoritmo MO-Max-Sum termina, in seguito al raggiungimento del numero massimo di iterazioni o alla disattivazione di tutti i nodi-variabile, vengono prelevate le marginali calcolate nell'ultima iterazione di ogni nodo-variabile. L'implementazione della procedura di elaborazione della frontiera di Pareto applica l'enunciato del Corollario 1: l'insieme delle soluzioni ottimali viene elaborato come l'intersezione tra gli insiemi delle soluzioni contenute in ciascuna marginale. Nella Figura 4.3 viene riportato lo pseudo-codice dell'algoritmo MO-Max-Sum, che riassume le operazioni dei nodi-variabile e dei nodi-funzione, e le operazioni per l'elaborazione della frontiera di Pareto.

4.5 Implementazione Java

L'algoritmo MO-Max-Sum è stato implementato come applicativo Java seguendo la specifica della sezione precedente. Il modello del problema da risolvere viene fornito all'applicazione mediante un documento XML contenente tutte le informazioni relative a funzioni e variabili necessarie per la costruzione del grafo fattorizzato. In Figura 4.4 è riportato un esempio di specifica XML del problema descritto nella Sezione 3.8.1.
L'esecuzione dell'algoritmo avviene in modo distribuito, attraverso l'uso di diversi thread. Per ogni nodo del grafo fattorizzato viene creato un thread la cui computazione procede asincronamente rispetto agli altri nodi. Se un nodo non può procedere nella computazione perché non ha ancora ricevuto dai vicini i messaggi il cui contenuto è necessario per l'elaborazione dei messaggi da inviare, il thread corrispondente viene messo in attesa. Ogni thread immagazzina i messaggi ricevuti nell'oggetto corrispondente al proprio nodo. Viene creato inoltre un thread speciale che raccoglie le funzioni marginali definitive di ogni nodo-variabile e, quando le possiede tutte, avvia la procedura di elaborazione della frontiera di Pareto.

Algorithm 4.4.1: MO-Max-Sum($variabili, funzioni$)

procedure Nodo-Variabile($vicini(x_j), iterazioni$)
 for each $F_i \in vicini(x_j)$
  do invia messaggio $q^1_{x_j \to F_i}$ inizializzato a 0
 for $t \leftarrow 2$ **to** $iterazioni$
  do
   - **for each** $F_i \in vicini(x_j)$
     - **do**
       - **while** messaggi r^{t-1} non ricevuti
         - **do** attendi messaggi r^{t-1}
       - elabora ed invia messaggio $q^t_{x_j \to F_i}$
   - $z_j(x_j) \leftarrow$ computa marginale
   - **if** marginale invariante
     - **then** interrompi iterazioni
 disattiva x_j
 return $(z_j(x_j))$

procedure Nodo-Funzione($vicini(F_i), iterazioni$)
 for $t \leftarrow 1$ **to** $iterazioni$
  do
   - **for each** $x_j \in vicini(F_i)$
     - **do**
       - **if** x_j disattivato
         - **then** $q^t_{x_j \to F_i} \leftarrow q^{t-1}_{x_j \to F_i}$
   - **for each** $x_j \in vicini(F_i)$
     - **do**
       - **while** messaggi q^t non ricevuti
         - **do** attendi messaggi q^t
       - elabora ed invia messaggio $r^t_{F_i \to x_j}$
   - **if** tutti i vicini sono disattivati
     - **then** interrompi iterazioni

procedure ElaboraFrontiera($marginali$)
 $frontiera \leftarrow z_1(x_1), z_1(x_1) \in marginali$
 $marginali \leftarrow marginali \backslash z_1(x_1)$
 for each $z_j(x_j) \in marginali$
  do $frontiera \leftarrow frontiera \cap z_j(x_j)$
 return $(frontiera)$

main
 $iterazioni \leftarrow$ iterazioni massime consentite
 $marginali \leftarrow \emptyset$
 for each $x_j \in variabili$
  do $marginali \leftarrow marginali \cup$ Nodo-Variabile($vicini(x_j), iterazioni$)
 for each $F_i \in funzioni$
  do Nodo-Funzione($vicini(F_i), iterazioni$)
 while marginali non ancora computate
  do attendi computazione marginali
 $frontiera \leftarrow$ ElaboraFrontiera($marginali$)

Figura 4.3: Pseudo-codice dell'algoritmo MO-Max-Sum

```
<graph objectives="3">
  <variables>
    <variable name="x1" position="1">
      <domain>
        <value>-1</value>
        <value>0</value>
        <value>1</value>
      </domain>
    </variable>
    <variable name="x2" position="2">
      <domain>
        <value>-1</value>
        <value>0</value>
        <value>1</value>
      </domain>
    </variable>
  </variables>
  <functions>
    <function name="f1" objective="1" arity="1">
      <scope>
        <variable>x1</variable>
      </scope>
      <tuples>
       <tuple f="-1">-1</tuple>
       <tuple f="0">0</tuple>
       <tuple f="-1">1</tuple>
      </tuples>
    </function>
    <function name="f2" objective="2" arity="1">
      <scope>
        <variable>x2</variable>
      </scope>
      <tuples>
       <tuple f="0">-1</tuple>
       <tuple f="1">0</tuple>
       <tuple f="0">1</tuple>
      </tuples>
    </function>
    <function name="fG" objective="1" arity="2">
      <scope>
        <variable>x1</variable>
        <variable>x2</variable>
      </scope>
      <tuples>
       <tuple f="0">-1 -1</tuple>
       <tuple f="-1">-1 0</tuple>
       <tuple f="-2">-1 1</tuple>
       <tuple f="1">0 -1</tuple>
       <tuple f="0">0 0</tuple>
       <tuple f="-1">0 1</tuple>
       <tuple f="2">1 -1</tuple>
       <tuple f="1">1 0</tuple>
       <tuple f="0">1 1</tuple>
      </tuples>
    </function>
  </functions>
</graph>
```

Figura 4.4: Specifica XML del problema descritto nella Sezione 3.8.1

Capitolo 5

Realizzazioni sperimentali e valutazioni

In questo capitolo vengono presentati i risultati sperimentali ottenuti attraverso l'esecuzione di diversi test sull'algoritmo MO-Max-Sum descritto nel capitolo precedente, a sostegno della soluzione progettata, correlati da un'analisi di qualità e significato delle soluzioni ottenute e della scalabilità dell'algoritmo.

5.1 Risultati sperimentali per il modello di riferimento

In questa sezione vengono illustrati e discussi i risultati della risoluzione attraverso l'algoritmo MO-Max-Sum del modello MO-DCOP di riferimento descritto nella Sezione 4.1.1. In questo test sono stati assegnati ai parametri del sistema gli stessi valori utilizzati nello studio [26], ad eccezione di alcune approssimazioni effettuate per ottenere dei valori interi. Il sistema ottenuto con questi assegnamenti viene denominato sistema $\mathcal{S}$. Nella Tabella 5.1 sono listati i valori assegnati ai parametri nel sistema $\mathcal{S}$.

i	a_i	b_i	c_i	α_i
1	-0.2	6	-5	12
2	-0.06	2.52	0	10
3	-0.29	6.38	-3	8
4	-0.13	5.98	-6	6
5	-0.055	3.63	-23	15
6	-0.15	7.5	-15	10

Tabella 5.1: Valori assegnati ai parametri caratteristici del modello di riferimento nel sistema $\mathcal{S}$

Nella Tabella 5.2 sono listati, per ogni variabile (e quindi per ogni agente), il valore $x_{i_{MAX}} = \arg\max_{x_i} F_{Oi}(x_i)$ che massimizza la funzione obiettivo locale dell'agente $F_{Oi}(x_i)$, ed il corrispondente valore massimo della funzione obiettivo $F_{Oi_{MAX}} = \max F_{Oi}(x_i) = F_{Oi}(x_{i_{MAX}})$. Le funzioni obiettivo degli agenti hanno una forma parabolica, le coppie di valori nella Tabella 5.2 indicano, per ogni agente, l'ascissa del massimo della rispettiva parabola, con il relativo valore della funzione obiettivo.

i	$x_{i_{MAX}}$	$F_{Oi_{MAX}}$
1	15	40.00
2	21	26.46
3	11	32.09
4	23	62.77
5	33	36.90
6	25	78.75

Tabella 5.2: Ottimi locali per le funzioni obiettivo di ciascun agente nel sistema $\mathcal{S}$

Il sistema è stato quindi analizzato nelle condizioni corrispondenti agli scenari proposti nello studio originale (identificati dai valori dei parametri Q_1, Q_2 e S), che rispecchiano le condizioni di portata dei flussi alta, media o bassa. Nella Tabella 5.3 vengono rappresentati i valori assegnati ai parametri relativi alle portate dei flussi che attraversano il sistema, per ciascuno scenario.

Scenario	Q_1	Q_2	S
Portata alta $\mathcal{S}_H$	80	35	10
Portata media $\mathcal{S}_M$	40	20	8
Portata bassa $\mathcal{S}_L$	15	8	3

Tabella 5.3: Possibili scenari per il sistema $\mathcal{S}$

Sono stati quindi definiti per ciascuna variabile dei domini discreti, composti da 5 valori, per rendere possibile un'analisi manuale di ogni scenario.

x_i	$D(x_i)$
x_1	$\{0, 3, 7, 12, 15\}$
x_2	$\{15, 18, 21, 22, 25\}$
x_3	$\{2, 7, 9, 11, 12\}$
x_4	$\{1, 6, 11, 23, 24\}$
x_5	$\{7, 10, 16, 19, 33\}$
x_6	$\{4, 5, 10, 17, 25\}$

Tabella 5.4: Domini delle variabili nel sistema $\mathcal{S}$

In questi test, i costi C_k per la violazione dei vincoli soft sono stati uniformemente fissati al valore 10 (formalmente attraverso un'utilità negativa pari a -10) per ogni vincolo soft.

5.1.1 Risultati degli scenari

Avendo definito i domini delle variabili ed i parametri del sistema per tutti gli scenari, si è cercato di risolvere manualmente ogni scenario attraverso deduzioni e ragionamenti sui valori delle variabili che potessero soddisfare i vincoli hard, scartando di conseguenza tutte le combinazioni di valori delle variabili che avrebbero reso la corrispondente soluzione insoddisfacibile. Nella risoluzione manuale si sono cercate quindi le soluzioni ammissibili per ogni scenario effettuando una restrizione dei domini delle variabili definiti nella Tabella 5.4, eliminando i valori che non soddisfano i vincoli hard F_7, F_8 e F_{10} definiti nella Tabella 4.2. Una volta trovato l'insieme degli assegnamenti che generano soluzioni ammissibili, si è elaborata la frontiera di Pareto scartando da questo insieme gli assegnamenti che generano vettori di soluzioni dominati.
Per ogni scenario, la frontiera di Pareto ottenuta attraverso la risoluzione teorica manuale del problema coincide con la frontiera elaborata dall'algoritmo MO-Max-Sum. Dal confronto con i risultati dello studio [26] in cui è stato proposto il modello di riferimento, è emerso che le soluzioni trovate attraverso il metodo proposto nello studio originale, basato su penalità e fattori di interesse locale, appartengono alle frontiere di Pareto elaborate dall'algoritmo MO-Max-Sum. Inoltre, la risoluzione operata dall'algoritmo MO-Max-Sum, a differenza del metodo proposto in [26], non dipende da particolari parametri quali i fattori di interesse locale, che condizionano le scelte degli agenti influenzando di conseguenza le soluzioni, ma elabora tutte le soluzioni ottimali senza introdurre ulteriori restrizioni nel sistema.
Questo risultato conferma la validità della modellazione di sistemi idrici mediante MO-DCOP e dimostra, unitamente ad altre prove nelle quali sono stati utilizzati modelli più semplici, ottenendo sempre gli insiemi delle soluzioni ottimali corretti, che l'algoritmo elabora correttamente la frontiera di Pareto del problema che risolve. Nelle Tabelle 5.5, 5.6 e 5.7 sono rappresentate le frontiere di Pareto dei tre possibili scenari del sistema $\mathcal{S}$.

Nelle Figure 5.1, 5.2 e 5.3 sono rappresentate graficamente le frontiere di Pareto degli scenari $\mathcal{S}_H$, $\mathcal{S}_M$ e $\mathcal{S}_L$. Ogni vettore soluzione è rappresentato da una spezzata che collega 7 punti, ciascuno corrispondente ad una funzione obiettivo. Graficamente, è possibile valutare l'ottimalità di tutte le soluzioni appartenenti ad una frontiera di Pareto osservando che nessuna spezzata

x_1	x_2	x_3	x_4	x_5	x_6	F_{O1}	F_{O2}	F_{O3}	F_{O4}	F_{O5}	F_{O6}	F_G
15	18	11	24	19	10	40.00	25.92	32.09	62.64	26.12	45.00	−10
15	21	11	24	7	25	40.00	26.46	32.09	62.64	−0.29	78.75	−10
15	21	12	23	16	17	40.00	26.46	31.80	62.77	21.00	69.15	0
15	22	11	24	16	17	40.00	26.40	32.09	62.64	21.00	69.15	0
15	25	11	24	19	17	40.00	25.50	32.09	62.64	26.12	69.15	0
15	25	12	23	33	4	40.00	25.50	31.80	62.77	36.90	12.60	−10

Tabella 5.5: Assegnamenti ottimali con i relativi vettori-soluzione per lo scenario con portata alta $\mathcal{S}_H$

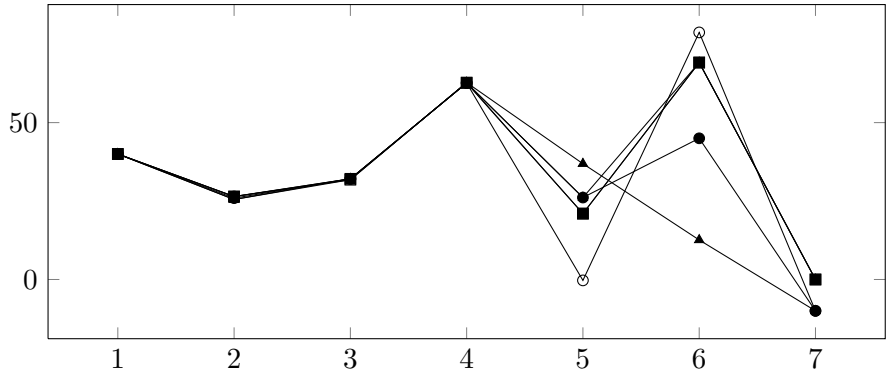

Figura 5.1: Frontiera di Pareto del sistema $\mathcal{S}_H$

x_1	x_2	x_3	x_4	x_5	x_6	F_{O1}	F_{O2}	F_{O3}	F_{O4}	F_{O5}	F_{O6}	F_G
15	15	9	11	19	5	40.00	24.30	30.93	44.05	26.12	18.75	−10
15	18	9	11	10	17	40.00	25.92	30.93	44.05	7.80	69.15	0

Tabella 5.6: Assegnamenti ottimali con i relativi vettori-soluzione per lo scenario con portata media $\mathcal{S}_M$

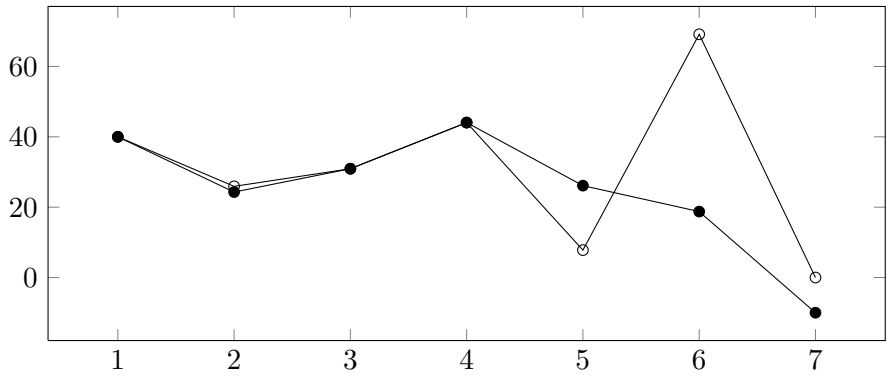

Figura 5.2: Frontiera di Pareto del sistema $\mathcal{S}_M$

x_1	x_2	x_3	x_4	x_5	x_6	F_{O1}	F_{O2}	F_{O3}	F_{O4}	F_{O5}	F_{O6}	F_G
0	18	2	6	10	10	−5.00	25.92	8.60	25.20	7.80	45.00	−40
0	21	2	6	16	4	−5.00	26.46	8.60	25.20	21.00	12.60	−40
3	21	2	6	7	10	11.20	24.30	8.60	25.20	−0.29	45.00	−50

Tabella 5.7: Assegnamenti ottimali con i relativi vettori-soluzione per lo scenario con portata bassa $\mathcal{S}_H$

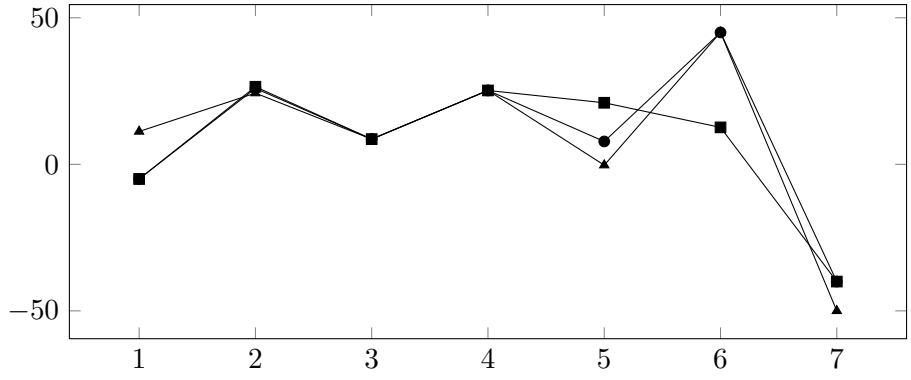

Figura 5.3: Frontiera di Pareto del sistema $\mathcal{S}_L$

è completamente sottostante ad un'altra; se una spezzata fosse completamente sottostante ad un'altra, il suo vettore delle funzioni obiettivo sarebbe dominato.

5.2 Analisi della complessità dell'algoritmo MO-Max-Sum

Le complessità computazionale dell'algoritmo MO-Max-Sum dipende strettamente dalla dimensione dei domini delle variabili e dall'arità delle funzioni del modello MO-DCOP. Il punto critico per le prestazioni è la computazione dei messaggi r, durante la quale un nodo-funzione deve confrontare tutte le possibili combinazioni di valori prelevati dai domini delle variabili dello scope.
Per una nodo-funzione F_i di arità a, ipotizzando che i domini delle variabili dello scope abbiano tutti la stessa dimensione d, la computazione di un messaggio $r_{i\to j}(x_j)$ secondo l'Equazione 4.3 ha complessità temporale pari a $O(d^{a+2})$, in quanto occorre iterare in cicli annidati:

- Il dominio della variabile x_j, per comporre ogni riga del messaggio, con complessità $O(d)$.

- Ogni possibile combinazione delle variabili dello scope di F_i, per operare la massimizzazione, con complessità $O(d^a)$.

- Il dominio di ogni variabile x_k dal cui nodo viene prelevato il contenuto dell'ultimo messaggio $q_{k \to i}(x_k)$, con complessità $O(d)$.

La computazione dei messaggi q e delle funzioni marginali z da parte di un nodo-variabile x_j richiede una complessità temporale decisamente inferiore, pari a $O(d)$, con andamento lineare rispetto alla dimensione del dominio della variabile ($|D(x_j)| = d$).
Nella risoluzione di un generico grafo fattorizzato, il punto critico per le prestazioni dell'algoritmo è individuato dal nodo-funzione con la massima arità. Per il modello di riferimento definito nella Sezione 4.1.1, il punto critico per le prestazioni coincide con il nodo-funzione F_{10} del grafo fattorizzato rappresentato in Figura 4.1.
Sotto il profilo della complessità spaziale, il punto critico è nuovamente il nodo-funzione con la massima arità, in quanto lo spazio richiesto per la rappresentazione di tutte le combinazioni delle variabili dello scope è pari a $O(d^a)$. Relativamente al modello descritto nella Sezione 4.1.1, la dimensione massima dei domini per cui è stato possibile eseguire l'algoritmo è 25; per dimensioni superiori, l'algoritmo non può essere eseguito in quanto le librerie per la gestione dei dati XML non sono più in grado di elaborare le specifiche del sistema e generano un errore della memoria.
Nella Tabella 5.8 vengono listati i tempi medi di esecuzione dell'algoritmo MO-Max-Sum per la risoluzione dei diversi scenari del sistema $\mathcal{S}$ definito nella Sezione 5.1.

Scenario	Tempo medio di esecuzione [ms]
$\mathcal{S}_H$	247.95
$\mathcal{S}_M$	243.12
$\mathcal{S}_L$	229.23

Tabella 5.8: Tempi di esecuzione dell'algoritmo MO-Max-Sum per la risoluzione degli scenari del sistema $\mathcal{S}$

Nella Tabella 5.9 vengono listati i tempi medi di esecuzione dell'algoritmo MO-Max-Sum per la risoluzione dello scenario $\mathcal{S}_H$ per diverse dimensioni dei domini delle variabili.

Si è infine analizzata la scalabilità dell'algoritmo MO-Max-Sum per un sistema più semplice, per poter aumentare maggiormente la dimensione dei domini delle variabili. Per questa analisi è quindi stato adottato il sistema rappresentato in Figura 3.2, composto dai nodi-variabile x_1 e x_2 e da tre

Dimensione domini	Tempo medio di esecuzione [ms]
5	247.95
8	529.08
10	974.71
12	1805.03
20	16496.90
25	78640.28

Tabella 5.9: Tempi di esecuzione dell'algoritmo MO-Max-Sum per la risoluzione del problema definito nella Sezione 4.1.1

nodi-funzione f_1, f_2 e f_G. Nella Tabella 5.10 vengono listati i tempi medi di esecuzione dell'algoritmo per la risoluzione di questo sistema per diverse dimensione dei domini delle variabili.

Dimensione domini	Tempo medio di esecuzione [ms]
100	3898.47
200	27520.75
400	920569.65
500	13088491.90

Tabella 5.10: Tempi di esecuzione dell'algoritmo MO-Max-Sum per la risoluzione del sistema rappresentato in Figura 3.2

Come considerazione finale sull'analisi della scalabilità dell'algoritmo MO-Max-Sum, si sottolinea che la complessità dell'algoritmo cresce esponenzialmente solo con il numero di variabili da cui ogni funzione dipende (l'arità della funzione). Gli algoritmi centralizzati e gli algoritmi distribuiti completi come MO-Adopt scalano invece esponenzialmente con il numero di variabili del sistema, pertanto con l'algoritmo MO-Max-Sum è possibile calcolare la frontiera di Pareto di problemi che gli algoritmi centralizzati e gli algoritmi come MO-Adopt non sono in grado di elaborare.

Capitolo 6

Conclusioni

In questo lavoro di tesi, il problema di ottimizzazione degli obiettivi in un sistema idrico è stato identificato come un caso particolare di problema di ottimizzazione multiobiettivo in ambito multiagente, contraddistinto dagli obiettivi di ogni singolo agente e da un obiettivo che rappresenta lo stato globale del sistema. Per la modellazione di questi problemi, è stato adottato il formalismo MO-DCOP, che dispone dell'espressività necessaria a rappresentare un generico problema di ottimizzazione multiobiettivo di vincoli distribuiti. Per un sistema idrico, le singole funzioni obiettivo e le relazioni tra gli agenti sono rappresentabili attraverso delle funzioni costo (o utilità), e quindi attraverso dei vincoli.
In questo lavoro sono state delineate le regole per la formalizzazione MO-DCOP di un generico sistema idrico, utilizzando come sistema di riferimento il caso particolare illustrato in [26], per il quale l'approccio risolutivo originale ha mostrato dei limiti nella gestione dell'ottimalità dei vincoli.
L'analisi degli algoritmi risolutivi per MO-DCOP ha focalizzato l'interesse di questo lavoro sull'algoritmo B-MOMS, e in particolare sull'algoritmo Max-Sum che ne costituisce la base, con particolare attenzione al ruolo delle funzioni marginali. Dal prodotto delle fasi progettuale ed implementativa si è ottenuto l'algoritmo MO-Max-Sum, un algoritmo che elabora la specifica MO-DCOP di un sistema idrico producendo come risultato finale la frontiera di Pareto di tutte le soluzioni ottimali del sistema. L'algoritmo MO-Max-Sum mostra particolare efficienza rispetto ai metodi centralizzati e ad algoritmi distribuiti come MO-Adopt, in quanto la complessità computazionale scala esponenzialmente solo con il numero di variabili da cui ogni funzione dipende, tipicamente di molto inferiore al numero totale di variabili (agenti) nel sistema.
Tra gli sviluppi futuri di questo lavoro di tesi vi sono l'inserimento di una fase

di bounding nell'algoritmo MO-Max-Sum, rendendolo così utilizzabile anche per problemi con grafi dei vincoli ciclici. Un interessante spunto considera uno studio approfondito dell'efficienza dell'algoritmo, con l'introduzione di alcune varianti sulle modalità di rappresentazione dei dati del problema da risolvere; si potrebbe ad esempio modificare la rappresentazione delle funzioni costo dei vincoli sostituendo l'elenco esplicito delle combinazioni possibili ed il relativo costo con una formulazione implicita dalla quale i nodi-funzione possano calcolare i valori dei costi durante l'elaborazione dei messaggi, confrontando l'efficienza delle diverse implementazioni. Un ulteriore spunto riguarda l'analisi di sistemi idrici di grandi dimensioni, con un numero elevato di variabili, per i quali è interessante lo studio di un metodo per la suddivisione efficiente in sottoproblemi da ottimizzare singolarmente e il coordinamento di questi sottoproblemi per l'elaborazione di soluzioni globalmente ottimali.

Bibliografia

[1] S. M. Aji and R. J. McEliece. The generalized distributive law. *IEEE Transactions on Information Theory*, 46(2):325–343, 2000.

[2] K. Apt. *Principles of Constraint Programming*. Cambridge, 2003.

[3] I. N. Athanasiadis. A review of agent-based systems applied in environmental informatics. In *Proceedings of the International Congress on Modelling and Simulation 2005*, pages 1574–1580, 2005.

[4] F. Bousquet and C. Le Page. Multi-agent simulations and ecosystem management: a review. *Ecological Modelling*, 176(3-4):313–332, 2004.

[5] E. Bowring, M. Tambe, and M. Yokoo. Multiply-constrained distributed constrained optimization. In *Proceedings of the 5th International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 1413–1420, 2006.

[6] R. Dechter and R. Mateescu. AND/OR search spaces for graphical models. *Artificial Intelligence*, 171(2-3):73–106, 2007.

[7] F. M. Delle Fave, R. Stranders, A. Rogers, and N. R. Jennings. Bounded decentralised coordination over multiple objectives. In *Proceedings of the 10th International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 371–378, 2011.

[8] A. Fedoruk and J. Denzinger. A general framework for multi-agent search with individual and global goals: Stakeholder search. *International Transactions on Systems Science and Applications*, 1(4):357–362, 2006.

[9] S. Fitzpatrick and L. Meertens. Distributed coordination through anarchic optimization. In Victor Lesser, Charles L. Ortiz, Milind Tambe, and Gerhard Weiss, editors, *Distributed Sensor Networks*, volume 9 of

Multiagent Systems, Artificial Societies, and Simulated Organizations, pages 257–295. Springer US, 2003.

[10] R. G. Gallager, P. A. Humblet, and P. M. Spira. A distributed algorithm for minimum-weight spanning trees. *ACM Transactions on Programming Languages and Systems*, 5(1):66–77, 1983.

[11] M. Giuliani, A. Castelletti, F. Amigoni, and X. Cai. Multi-agent systems optimization for distributed watershed management. In *Proceedings of the International Congress on Environmental Modelling and Software (iEMSs2012)*, 2012.

[12] K. Hirayama and M. Yokoo. Distributed partial constraint satisfaction problem. In *Proceedings of the 3rd International Conference on Principles and Practice of Constraint Programming*, pages 222–236, 1997.

[13] K. Hirayama and M. Yokoo. An approach to over-constrained distributed constraint satisfaction problems: Distributed hierarchical constraint satisfaction. In *Proceedings of the 4th International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 135–142, 2000.

[14] Q. B. Le, R. Seidl, and R. W. Scholz. Feedback loops and types of adaptation in the modelling of land-use decisions in an agent-based simulation. *Environmental Modelling and Software*, (27-28):83–96, 2012.

[15] R. T. Maheswaran, J. P. Pearce, and M. Tambe. A family of graphical-game-based algorithms for distributed constraint optimization problems. In *Proceedings of the Coordination of Large-Scale Multiagent Systems 2005*, pages 127–146, 2005.

[16] R. Marinescu. Exploiting problem decomposition in multi-objective constraint optimization. In *Proceedings of the 15th international Conference on Principles and Practice of Constraint Programming*, pages 592–607, 2009.

[17] R. Marinescu and R. Dechter. AND/OR branch-and-bound for graphical models. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, pages 224–229, 2005.

[18] T. Matsui, M. Silaghi, K. Hirayama, M. Yokoo, and H. Matsuo. Distributed search method with bounded cost vectors on multiple objectives DCOPs. In *Proceedings of the 15th International Conference on Principles and Practice of Multi-Agent Systems*, pages 137–152, 2012.

[19] P. Modi, W. Shen, M. Tambe, and M. Yokoo. ADOPT: Asynchronous distributed constraint optimization with quality guarantees. *Artificial Intelligence*, 161(1-2):149–180, 2005.

[20] T. L. NG, J. W. Eheart, X. Cai, and J. B. Braden. An agent-based model of farmer decision-making and water quality impacts at the watershed scale under markets for carbon allowances and a second-generation biofuel crop. *Water Resources Research*, 47(9), 2011.

[21] J. P. Pearce and M. Tambe. Quality guarantees on k-optimal solutions for distributed constraint optimization problems. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pages 1446–1451, 2007.

[22] A. Petcu and B. Faltings. DPOP: A scalable method for multiagent constraint optimization. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, pages 266–271, 2005.

[23] A. Rogers, A. Farinelli, R. Stranders, and N. R. Jennings. Bounded approximate decentralised coordination via the max-sum algorithm. *Artificial Intelligence*, 175(2):730–759, 2011.

[24] E. Rollon. *Multi-Objective Optimization in Graphical Models*. PhD thesis, Universitat Politecnica de Catalunya, 2008.

[25] S. J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Inc., 1996.

[26] Y. C. E. Yang, X. Cai, and D.M. Stipanovič. A decentralized optimization algorithm for multiagent system-based watershed management. *Water Resources Research*, 45:1–18, 2009.

[27] M. Yokoo, E. H. Durfee, T. Ishida, and K. Kuwabara. The distributed constraint satisfaction problem: Formalization and algorithms. *IEEE Transactions on Knowledge and Data Engineering*, 10(5):673–685, 1998.

[28] M. Yokoo and K. Hirayama. Distributed breakout algorithm for solving distributed constraint satisfaction problems. In *Proceedings of the 2nd International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 401–408, 1996.

[29] Y. Zhengyu, C. Kiekintveld, A. Kumar, and M. Tambe. Local optimal solutions for DCOP: New criteria, bound, and algorithm. In *Second International Workshop on Optimisation in Multi-Agent Systems*, 2009.