POLITECNICO DI MILANO
Corso di Laurea Magistrale in Ingegneria Informatica
Dipartimento di Elettronica, Informazione e Bioingegneria

VALUTAZIONE SIMULATA DI UN SISTEMA MULTIAGENTE BASATO SU MO-DCOP PER LA GESTIONE DI BACINI IDRICI

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

Tesi di Laurea Magistrale di:
Stefano SUARDI
Matricola 782915

Anno Accademico 2012-2013

Sommario

Nel corso degli ultimi anni vi è stato un crescente interesse nei confronti delle tematiche ambientali anche da parte dell'opinione pubblica che si è resa conto della scarsità delle risorse presenti sul nostro pianeta. Tra queste risorse ce n'è una di vitale importanza, cioè l'*acqua*, o meglio l'acqua dolce che rappresenta solo il 3% di quella presente sulla Terra, e più dei due terzi di questa si trova nei ghiacciai in forma solida. Di conseguenza la gestione di questa risorsa è di estrema importanza sia per l'uomo che per la vita in generale. Lo scopo di questa tesi è di mostrare le potenzialità di un particolare algoritmo di ottimizzazione distribuita multiagente multiobbiettivo per la gestione ottimale di un bacino idrico. Tale algoritmo è stato testato su vari problemi a crescente complessità. Attraverso la generazione automatica di scenari più o meno complessi, alla loro traduzione in modelli direttamente elaborabili dall'algoritmo, e alla loro successiva risoluzione abbiamo evidenziato un'ottima scalabilità dell'algoritmo al crescere della complessità del modello e il principale difetto legato al fatto che l'algoritmo non è completo e quindi in alcune condizioni non siamo in grado di garantire una certa qualità delle soluzioni.

Abstract

Over the last few years there has been a growing interest in environmental issues also from the perspective of the public opinion, who realized the scarcity of resources on our planet. Among these resources one is of vital importance, namely *water*, or rather the fresh water that is only 3% of water on Earth, and more than two-thirds of this is located in glaciers in solid form. Consequently, the management of this resource is very important both for humans and for life in general. The purpose of this thesis is to evaluate a particular distributed multi-agent multi-objective optimization algorithm for optimal watershed management. Such an algorithm has been tested on various problems at growing complexity. Through the automatic generation of more or less complex scenarios, to their translation into models that can be processed by the algorithm, and their subsequent resolution we show that the algorithm has excellent scalability with respect to the increasing of complexity of the model and that its main drawback is related to its incompleteness and therefore in some conditions we are not able to guarantee a certain quality of the solutions.

Ringraziamenti

Queste poche righe mi danno la possibilità di esprimere gratitudine a tutte le persone che mi sono state affianco nel raggiungere questo traguardo importante. Gratitudine che non sono mai riuscito dire a voce, forse a causa del mio carattere o forse perché non mi ero ancora mai soffermato a riflettere su questi ultimi anni di sacrifici e perché no, anche di qualche piccola soddisfazione.

Intendo innanzitutto ringraziare i miei genitori Emanuela e Giovanni e mio fratello Andrea per avermi sempre supportato e sopportato. Ringrazio poi tutti i miei parenti, in particolare mia nonna, gli zii e le zie.

Un grazie speciale va a Valentina che mi è rimasta affianco più di tutti in questi anni, condividendo gioe e dolori.

Grazie anche a tutti i miei amici, quelli del Mc e non solo. Grazie anche ai colleghi nonostante negli ultimi anni ci siamo frequentati troppo poco.

Un ringraziamento ulteriore va a Francesco Amigoni e Matteo Giuliani che sono sempre stati disponibili a chiarire i miei dubbi e hanno saputo indirizzarmi al meglio di fronte agli ostacoli incontrati.

Stefano

Indice

Elenco delle figure

ELENCO DELLE FIGURE

Elenco delle tabelle

Capitolo 1

Introduzione

Questa tesi si propone di combinare tematiche relative a due campi, il primo è l'ottimizzazione basata su tecniche multiagente, il secondo è legato all'ingegneria ambientale, nella fattispecie alla gestione di un bacino idrico. La tesi pone l'accento sulla tematica di ottimizzazione dell'uso dell'acqua lungo il corso di un fiume adottando il punto di vista di un'ipotetica autorità di bacino, che deve quindi considerare tutti gli utenti presenti all'interno del sistema.

Gli obbiettivi di una ricerca in questa direzione sono molteplici, tra i principali individuiamo la preservazione di alcune aree ecologiche, e il miglioramento della fruizione dell'acqua per tutte le entità coinvolte, anche per coloro che essendo a valle sono più penalizzate. Un altro punto cruciale è legato alla gestione sinergica degli impianti di produzione di energia idroelettrica. La supervisione di tale scenario, però, non può essere effettuata direttamente in modo centralizzato [9, 14, 16], ma è svolta indirettamente mediante la stipulazione di apposite norme rappresentate da vincoli nel modello considerato [7, 17].

Per perseguire gli obbietti di ottimizzazione adoperiamo un algoritmo sviluppato nel mondo degli agenti autonomi, in particolar modo l'algoritmo *B-MOMS* [4] facente riferimento al formalismo *MO-DCOP*, Quindi usiamo un'ottimizzazione multiobbiettivo che ci consente di tener conto degli interessi di ciascun attore. Di conseguenza la nostra soluzione è rappresentata da una frontiera di Pareto, cioè dall'insieme delle utilità relative agli assegnamenti Pareto

efficienti, cioè quegli assegnamenti che non possono migliorare la condizione di un soggetto senza peggiorare la condizione di un altro. L'algoritmo in grado di risolvere questo tipo di problema è già presente in letteratura [4], ed esiste già un'implementazione funzionante [3] su un particolare caso di studio [17].

Obbiettivo di questa tesi è quindi quello di generare alcuni scenari idrici verosimili per verificare la validità del sopra citato algoritmo. Quindi il nostro scopo è quello evidenziarne i punti di forza e i difetti, proponendo delle migliorie.

La simulazione è composta da tre fasi, la prima prevede la generazione di scenari più o meno complessi su cui si basa la seconda fase che consiste nell'applicazione dell'algoritmo. Infine, nella terza fase analizziamo i dati raccolti dalle simulazioni cercando di evidenziare limiti e pregi di tale approccio.

La tesi è strutturata nel modo seguente.

Nel Capitolo 2 mostriamo lo stato dell'arte relativamente alla parte dei sistemi multiagente.

Nel Capitolo 3 introduciamo gli aspetti di carattere ambientale e descriviamo l'algoritmo B-MOMS.

Nel Capitolo 4 mostriamo il progetto dal punto di vista logico includendo le problematiche riscontrate.

Nel Capitolo 5 descriviamo l'implementazione del sistema suddivisa nei moduli principali.

Nel Capitolo 6 eseguiamo alcuni test di cui mostriamo i dati raccolti.

Nel Capitolo 7 traiamo le conclusioni evidenziando sia i risultati positivi che i limiti e di conseguenza alcune linee guida sui possibili sviluppi futuri utili al miglioramento dell'approccio esposto.

Nell'appendice A riportiamo gli schemi dei file XML usati.

Capitolo 2

Stato dell'arte

In questo capitolo mostriamo uno spaccato dell'intelligenza artificiale partendo dalla definizione di agente fino alla presentazione di formalismi basati sull'ottimizzazione distribuita tra più agenti in un'ottica multiobbiettivo.

2.1 Sistemi intelligenti

La base su cui si fonda l'intelligenza artificiale è la definizione di agente razionale che descriviamo nei prossimi paragrafi.

2.1.1 Agente razionale

Un *agente* [13] è tutto ciò che può essere visto come un sistema in grado di percepire l'*ambiente* che lo circonda mediante l'uso di *sensori* e agisce su quest'ultimo attraverso degli *attuatori* (Figura 2.1).

Un tale agente per essere considerato *razionale* deve poter scegliere, tra le varie azioni possibili, quella che gli permetta di raggiungere il massimo grado di successo, quindi, per fare ciò, è necessario definire una misura prestazionale. Questa metrica è solitamente oggettiva, cioè non dipende dal giudizio dell'agente stesso, e valuta i risultati ottenuti piuttosto che il comportamento che ci si aspetta dall'agente.

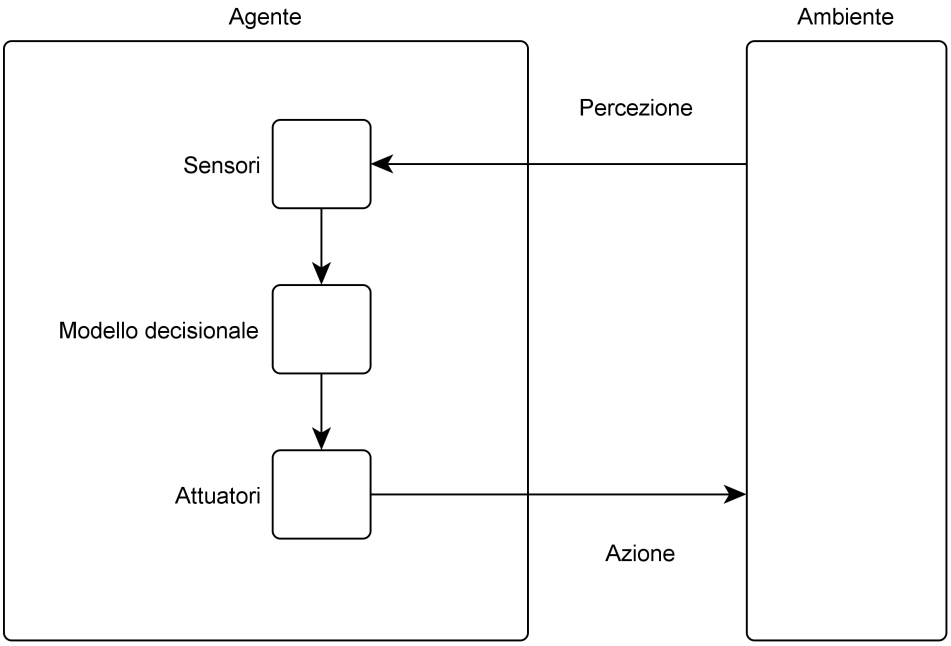


Figura 2.1: Agente

La razionalità di un agente dipende inoltre da altri tre fattori: la conoscenza dell'ambiente che lo circonda, le azioni che può effettuare e la sequenza di azioni che lo hanno portato allo stato attuale.

Quindi si può definire un *agente razionale* [13] come quell'agente che per ogni sequenza di percezioni scelga un'azione che massimizzi la sua misura prestazionale date le informazioni a sua disposizione, cioè quelle ottenute durante la sequenza percettiva insieme a cioè che l'agente conosce.

Ciascun ambiente può essere classificato mediante alcune proprietà, tra le quali troviamo:

- *Completamente osservabile/parzialmente osservabile*: se i sensori sono in grado di misurare in ogni istante l'ambiente nei suoi aspetti rilevanti allora quest'ultimo è completamente osservabile, altrimenti solo parzialmente.

- *Deterministico/stocastico*: a secondo che lo stato successivo sia completamente determinato dallo stato corrente e dall'azione eseguita o

meno.

- *Episodico/sequenziale*: in un ambiente episodico la scelta dell'azione dipende solo dallo stato corrente, mentre in uno sequenziale dipende dalla loro storia.
- *Statico/dinamico*: un ambiente è dinamico quando cambia mentre l'agente sta decidendo l'azione da prendere, statico altrimenti.
- *Discreto/continuo*: in base al modo in cui sono gestiti il tempo, le percezioni e le azioni.
- *Agente singolo/multiagente*: a seconda del numero di agenti che si sceglie di considerare.

Un agente può inoltre essere caratterizzato dalla complessità del suo modulo decisionale, troviamo infatti:

- *Agenti reattivi semplici*: questi agenti si basano esclusivamente sulla percezione dello stato attuale, prendono quindi le decisioni in base a delle regole che dipendono solo dalla percezione corrente.
- *Agenti reattivi basati su modello*: questa tipologia prevede l'aggiunta di un modello in grado di tener traccia della storia delle percezioni, così da poter in qualche modo risolvere la mancanza di informazioni dovuta ad un ambiente non completamente osservabile.
- *Agenti basati su obbiettivi*: l'introduzione del concetto di obbiettivo da perseguire consente all'agente di pianificare le azioni future, è però necessario poter prevedere il prossimo stato al verificarsi di una data azione.
- *Agenti basati su utilità*: un modello decisionale di questo tipo introduce una funzione di utilità che dipende dallo stato, in questo modo l'agente può preferire una particolare azione piuttosto che un'altra, quindi è in grado di ordinare le possibili soluzioni.

- *Agenti in grado di apprendere*: questi agenti sono in grado di prendere le decisioni in base alla loro esperienza, cioè sono in grado di migliorare le proprie prestazioni in base all'apprendimento.

2.1.2 Problemi di soddisfacimento di vincoli

Un *problema di soddisfacimento di vincoli* o *CSP (Constraint Satisfaction Problem)*, è definito da un insieme finito di variabili $X = \{x_1, \ldots, x_n\}$, e da un insieme di vincoli $C = \{c_1, \ldots, c_m\}$. A ciascuna variabile x_i è assegnato un valore v_i che appartiene al dominio corrispondente D_i. È necessario introdurre il concetto di assegnamento, cioè un insieme, anche vuoto, di coppie ordinate $\langle x_i, v_i \rangle$ dove $x_i \in X$ e $v_i \in D_i$. Un assegnamento che non viola alcun vincolo è chiamato consistente, mentre è invece completo se menziona tutte le variabili. Una soluzione di un CSP è quindi un assegnamento consistente e completo.

Il difetto principale di una tale formalizzazione è quello di considerare tutte le soluzioni sullo stesso piano di importanza, è quindi possibile generalizzare il modello proposto aggiungendo una funzione obbiettivo mediante la quale si possono ordinare le soluzioni, si parla quindi di *problemi vincolati di ottimizzazione* o più semplicemente di *COP (Constraint Optimization Problem)*.

2.2 Sistemi multiagente

Un *sistema multiagente* o *MAS (Multi-Agent System)* è costituito da un insieme di agenti che interagiscono tra di loro e con l'ambiente esterno, quindi si tratta di un'estensione di quanto visto fin'ora.

2.2.1 Problemi distribuiti di soddisfacimento di vincoli

Un *problema distribuito di soddisfacimento di vincoli* o *DCSP (Distributed Constraint Satisfaction Problem)* [18] è l'analogo di CSP definito per sistemi composti da più agenti. In questa formalizzazione le variabili e i vincoli sono

distribuiti su più agenti che comunicano tra di loro sotto le seguenti assunzioni: lo scambio di messaggi avviene solo tra agenti che conoscono l'indirizzo del destinatario e per ipotesi i messaggi vengono ricevuti nell'ordine in cui sono stati inviati.

Formalmente un DCSP è composto da m agenti e da n variabili, ciascuna variabile x_j appartiene ad un agente i e si scrive $appartiene(x_j, i)$. Inoltre si formalizza il fatto che l'agente a conosce il vincolo c_k e si scrive $conosce(c_k, a)$. La soluzione di un DCSP sia ha se e sole se valgono le seguenti condizioni:

- $\forall i, \forall x_j$ tali che $appartiene(x_j, i)$, a x_j è assegnato un valore d_j.
- $\forall a, \forall c_k$ tale che $conosce(c_k, a)$, il vincolo c_k è soddisfatto per l'assegnamento $x_j = d_j$.

2.2.2 Problemi distribuiti di ottimizzazione vincolata

Un problema distribuito di ottimizzazione vincolata, o più comunemente un *DCOP (Distributed Constraint Optimization Problem)* [11] consiste in n variabili $V = \{x_1, x_2, \dots x_n\}$, ciascuna associata ad un agente, in cui, il valore delle variabili è preso dai rispettivi domini discreti e finiti $D_1, D_2, \dots D_n$. Una variabile può essere controllata solo dall'agente a cui appartiene e che conosce i valori del suo dominio. Quindi l'obbiettivo per gli agenti è la scelta dei valori che, se assegnati alle variabili, rendano massima la funzione obbiettivo globale $U(\mathbf{x})$, quest'ultima è data come la somma delle varie funzioni obbiettivo e i vincoli presenti, anch'essi trasformati in funzioni. Ciascuna di queste ultime può essere definita come: $U_i(\mathbf{x}_i) : \mathbf{D}_i \to \mathbb{N}$ dove $\mathbf{x}_i \subseteq V$ e $\mathbf{D}_i$ equivale al prodotto cartesiano dei domini delle variabili da cui U_i dipende. Quindi formalmente l'obbiettivo è quello di massimizzare la seguente funzione obbiettivo:

$$\max_{\mathbf{x}} \sum_{i}^{l} U_i(\mathbf{x}_i)$$

in cui U_i indica l'i-esima funzione utilità (o vincolo) e l il numero di funzioni utilità (o vincoli) del sistema.

Bounded Max-Sum

Un algoritmo in grado di risolvere un DCOP è l'algoritmo *BMS (Bounded Max-Sum)* [12] che rappresenta una versione migliorata dell'algoritmo *Max-Sum* presente nella classe di modelli *GDL (Generalised Distributive Law)* [1]. Quest'ultimo algoritmo ha il difetto di funzionare bene su grafi aciclici, ma non vi è alcuna garanzia di convergenza alla soluzione ottima in presenza di cicli. Per questo motivo è stato sviluppato BMS in grado di garantire su qualsiasi grafo la convergenza alla soluzione ottima e di fornire un limite superiore dell'errore di approssimazione.

Per poter applicare l'algoritmo in questione è necessario costruire il cosiddetto *grafo fattorizzato* di un problema DCOP i cui nodi sono le variabili e i vincoli, mentre gli archi rappresentano le relazioni tra questi ultimi.

Il funzionamento prevede lo scambio di due tipi di messaggi, il primo dalla variabile x_j alla funzione U_i:

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

dove $M(j)$ indica l'insieme degli indici delle funzioni connesse alla variabile x_j.

Il secondo dalla funzione U_i alla 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 $N(i)$ indica l'insieme degli indici delle variabili connesse alla funzione U_i.

Considerando un grafo aciclico possiamo calcolare la funzione marginale di ciascuna variabile mediante la seguente formula:

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

da cui ricaviamo l'assegnamento ottimale di x_j:

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

Per il funzionamento anche su grafi ciclici è necessario renderli aciclici mediante il calcolo dell'albero di copertura minimo ottenuto pesando

opportunamente gli archi con il seguente peso:

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

Quindi è possibile applicare l'algoritmo Max-Sum garantendo un limite superiore dell'errore di approssimazione che equivale a:

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

dove W è pari alla somma dei pesi degli archi tagliati e $\tilde{V}_m = \sum_i \min_{\mathbf{x}_i^c} U_i(\tilde{\mathbf{a}}_i)$ in cui $\mathbf{x}_i^c$ corrisponde all'insieme delle variabili eliminate dalla funzione U_i, mentre $\tilde{\mathbf{a}}_i$ è l'assegnamento ottimo trovato nel modello approssimato.

2.3 Ottimizzazione multiobiettivo

Questo problema è definito come la massimizzazione simultanea di k funzioni obbiettivo definite su un insieme $\mathbf{x} = \{x_1, \ldots, x_n\}$ di n variabili in cui ciascuna x_j appartiene a un dominio discreto $D_{x_j} = \{d_j^1, \ldots, d_j^{|D_{x_j}|}\}$. Una soluzione è quindi un assegnamento $\mathbf{a}^* = \{(x_1 = d_1), \ldots, (x_n = d_n)\}$ tale che

$$\mathbf{a}^* = \arg\max_{\mathbf{a} \in D_{\mathbf{x}}} \mathbf{U}(\mathbf{x}) = [U^1(\mathbf{x}_1), \ldots, U^k(\mathbf{x}_k)]^T$$

dove $\mathbf{x}_i \subseteq \mathbf{x}$ e $D_{\mathbf{x}} = \bigtimes_{j=1}^{n} D_{x_j}$.

2.3.1 MO-DCOP

Si può formalizzare un MO-DCOP come estensione di un DCOP in ottica multiobbiettivo [4], quindi risolvere un MO-DCOP equivale a risolvere k DCOP simultaneamente. Formalmente si deve massimizzare il seguente vettore obbiettivo:

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

Dato che è formato da più DCOP può essere scomposto in N fattori:

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

dove $\mathbf{x}_i \subseteq \mathbf{x}$ rappresenta il vettore delle variabili da cui ciascuna U_i dipende, quest'ultima può essere così definita:

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

B-MOMS

Uno dei possibili algoritmi risolutori di un MO-DCOP è l'algoritmo *B-MOMS (Bounded Multi-Objective Max-Sum)* [4] che rappresenta l'estensione al caso multiobbiettivo dell'algoritmo BMS mostrato nel Paragrafo 2.2.2. È composto da tre fasi principali descritte di seguito:

- *Bounding*, in cui si ottiene l'albero di copertura minimo del grafo fattorizzato contente cicli.
- *Max-sum*, nella quale gli agenti si coordinano per costruire la frontiera di Pareto.
- *Value-propagation*, in cui gli agenti selezionano un assegnamento consistente tra quelli che formano la frontiera.

Capitolo 3

Impostazione del problema di ricerca

3.1 Lavori precedenti di ottimizzazione di sistemi idrici

La gestione dei bacini idrici tradizionalmente è risolta mediante l'uso di tecniche centralizzate le quali portano a dei risultati in generale molto efficienti [2] ma che spesso non sono praticabili su scenari reali [9, 14, 16]. Data l'impraticabilità di un approccio centralizzato, solitamente troviamo situazioni reali non coordinate in cui ciascuna parte tende a massimizzare il proprio profitto locale, questo comportamento tende a far raggiungere bassi livelli di efficienza a livello di sistema [10]. A causa della bassa efficienza ottenuta a livello di sistema si è cercato di raggiungere un compromesso tra questi due estremi mediante un meccanismo di coordinamento [15]. Tale approccio però manca di solide basi teoriche che vengono introdotte modellizzando il sistema basandosi sui MAS [17]. Questa modellizzazione viene affinata adottando i formalismi DCOP e DCSP in [7] e ulteriormente approfondita in [8], in cui si evidenza la superiorità del formalismo DCOP rispetto a DCSP.

Su quest'ultimo lavoro è stata sviluppata la tesi di Enrico Bontempi [3],

in cui viene formalizzato un sistema idrico mediante MO-DCOP e viene implementato l'algoritmo B-MOMS relativamente al caso di studio proposto da Yang et al. [17].

Questo è dunque il punto di partenza della presente tesi, il cui obbiettivo è quello di verificare la validità dell'approccio fin'ora sviluppato in altri contesti realistici. Per fare ciò dobbiamo procedere in tre fasi, la prima consiste nella generazione di sistemi idrici plausibili e del relativo modello. La seconda è la simulazione vera e propria dell'algoritmo precedentemente implementato, mentre l'ultima verte sulla valutazione dei dati ottenuti.

3.2 Assunzioni e semplificazioni usate nella modellizzazione dei sistemi idrici

Eseguiamo lo studio della gestione ottimale di un bacino idrico basandoci sul modello proposto da Yang et al., quindi utilizziamo un modello statico, cioè che è indipendente dal tempo. L'unica grandezza fisica a cui ci riferiamo è la quantità d'acqua disponibile nel sistema espressa in m^3, mentre la funzione obbiettivo è adimensionale in quanto non siamo in un contesto reale. Inoltre non consideriamo l'incertezza del modello, per esempio dovuta a piogge.

3.3 Caso di studio

Mostriamo un caso di studio originariamente proposto da Yang et al. in cui si utilizza un approccio basato su agenti per modellare un sistema idrico. Quest'ultimo è formato da un fiume principale e da uno secondario. In Figura 3.1 mostriamo gli agenti che si trovano nel sistema idrico, ciascuno di essi ha un interesse differente, ciò è espresso mediante la funzione obbiettivo relativa ad ogni singolo agente, quindi individuiamo:

- A_1: una città, deve far fronte alla domanda di acqua da parte dei suoi cittadini.

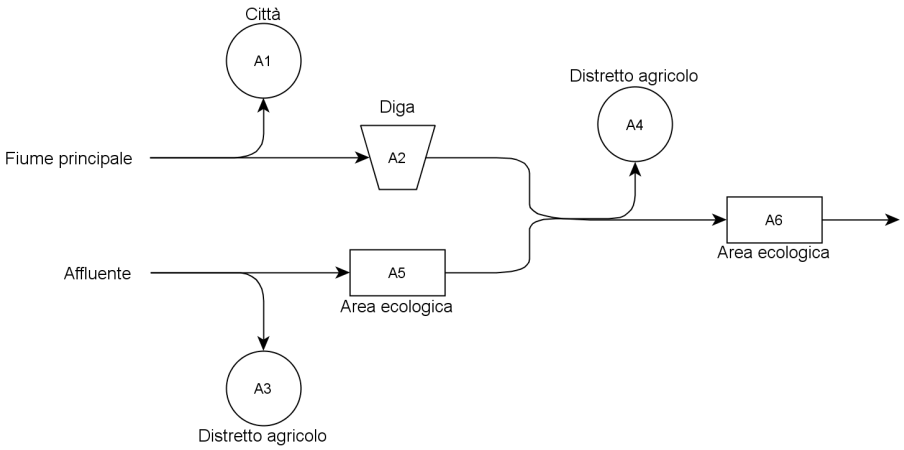


Figura 3.1: Sistema idrico proposto nel caso di studio

- A_2: la diga posta sul fiume principale, il suo obbiettivo è la produzione di energia idroelettrica.
- A_3: il distretto agricolo posto sul corso secondario, utilizza l'acqua per soddisfare le proprie esigenze irrigue.
- A_4: il distretto agricolo che troviamo lungo il fiume principale, necessita di acqua per l'irrigazione.
- A_5: l'area ecologica da preservare posta lungo l'affluente.
- A_6: l'area ecologica da preservare del corso d'acqua principale.

Il sistema descritto è formato da un insieme di vincoli mostrati nella Tabella 3.1. Sono presenti inoltre le varie funzioni obbiettivo relative agli agenti tutte nella forma proposta di seguito e riferita all'agente i-esimo:

$$\max_{x_i} f_i(x_i)$$

3.4 Aderenza al formalismo MO-DCOP

Di seguito mostriamo in che modo aderiamo alla formalizzazione MO-DCOP e le semplificazioni fatte. Utilizziamo un agente per ogni variabile decisionale

Agenti	Vincoli	Descrizione
A_1	$\alpha_1 - x_1 \leq 0$ $\alpha_2 - x_1 \leq Q_1$	x_1: prelievo di acqua della città α_1: fabbisogno minimo della città α_2: flusso minimo richiesto dalla diga Q_1: portata iniziale del fiume principale
A_2	$x_2 \leq S + Q_1 - x_1$	x_2: acqua rilasciata dalla diga S: livello d'acqua della diga
A_3	$\alpha_3 - x_3 \leq 0$ $x_3 \leq Q_2$	x_3: prelievo di acqua della fattoria 3 α_3: fabbisogno minimo dell'agente A_3 Q_2: portata iniziale del fiume secondario
A_4	$\alpha_4 - x_4 \leq 0$ $x_4 \leq x_2 + Q_2 - x_3$	x_4: prelievo di acqua dell'agente 4 α_4: fabbisogno minimo dell'agente 4
A_5	$x_5 = Q_2 - x_3$ $\alpha_5 - Q_2 + x_3 \leq 0$	x_5: acqua che fluisce a valle di A_5 α_5: portata necessaria all'area ecologica
A_6	$x_6 = x_2 + x_5$ $-x_2 + k + x_3 + x_4 \leq 0$ $k = \alpha_6 - Q_2$	x_6: acqua che fluisce a valle di A_6 α_6: portata necessaria all'area ecologica

Tabella 3.1: Formulazione proposta da Yang et al.

per un totale di n agenti e n variabili. Il vettore delle funzioni obbiettivo è formato da una componente per ciascun agente, più una relativa all'intero sistema, in totale è composto da $n+1$ elementi. Le prime n componenti quindi non dipendono dall'intero vettore delle variabili $\mathbf{x} = \{x_1, \ldots, x_n\}$ ma bensì dalla sola variabile a cui si riferiscono, quindi otteniamo la seguente funzione obbiettivo:

$$\mathbf{U}(\mathbf{x}) = [U^1(x_1), \ldots, U^n(x_n), U^{n+1}(\mathbf{x})]^T$$

Come abbiamo già detto ogni agente massimizza il proprio profitto mediante una funzione obbiettivo $f_i(x_i)$ che coincide proprio con $U^i(x_i)$.

Consideriamo che il sistema sia composto da m vincoli, in cui al vincolo j-esimo associamo una funzione utilità pari a $U_j^{n+1}(\mathbf{x}_j)$ e che assume un valore in base alla violazione o meno del vincolo a cui è riferita. Quest'ultima funzione dipende dalle variabili a cui è collegato il vincolo cioè $\mathbf{x}_j \subseteq \mathbf{x}$. Per esempio consideriamo il vincolo j-esimo $x_4 \leq x_2 + Q_2 - x_3$ relativo all'agente A_4 di Tabella 3.1, se esso è soddisfatto allora $U_j^{n+1}(x_3, x_4) = 0$ altrimenti $U_j^{n+1}(x_3, x_4) = -\infty$, che scritto in modo compatto risulta equivalente a:

$$C_j(x_3, x_4) = \begin{cases} 0 & x_4 \leq x_2 + Q_2 - x_3 \\ -\infty & x_4 > x_2 + Q_2 - x_3 \end{cases}$$

Quindi l'ultima componente è definita come la somma di tutte le utilità relative ai vincoli del sistema:

$$U^{n+1}(\mathbf{x}) = \sum_{j=0}^{m} U_j^{n+1}(\mathbf{x}_j)$$

Consideriamo inoltre, per ciascuna variabile, un dominio finito e discreto pari a D_i e quindi il vettore dei domini è rappresentato dal prodotto cartesiano dei singoli domini $\mathbf{D} = \bigtimes_i D_i$.

3.5 Analisi dell'algoritmo B-MOMS

La realizzazione dell'algoritmo B-MOMS (sviluppata in [3]) che usiamo in questa tesi si basa sulle formalizzazioni proposte nelle Sezioni 2.2.2 e 2.3.1. L'implementazione in questione, però, è progettata per funzionare esclusivamente

su grafi fattorizzati aciclici quindi consta solamente di due fasi che mostriamo di seguito.

La prima fase si occupa dell'esecuzione dell'algoritmo Max-Sum in modo distribuito, quindi prevede la costruzione del grafo fattorizzato partendo dalla definizione del problema. Questo grafo è ottenuto inserendo un nodo-funzione per ogni funzione prevista dal problema, quindi sia le funzioni utilità che i vincoli tradotti in funzioni, e un nodo-variabile per ciascuna variabile del sistema, cioè per ogni agente presente. I vari nodi sono poi collegati inserendo un arco tra le funzioni e le variabili da cui esse dipendono. Ogni nodo creato è quindi in grado di inviare e ricevere messaggi dai nodi a cui è collegato, nella fattispecie distinguiamo i messaggi inviati dai nodi-funzione, che chiamiamo messaggi r, e i quelli inviati dai nodi-variabile che chiamiamo invece messaggi q. Richiamiamo quindi le formule previste per il calcolo di tali messaggi indicando con t l'istante di tempo al quale si riferiscono:

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

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

dove $M(j)$ e $N(i)$ indicano rispettivamente l'insieme degli indici dei nodi connessi alla variabile x_j e quelli connessi alla funzione U_i. Dalla forma evinciamo che i messaggi di tipo q dipendono da quelli di tipo r riferiti all'istante precedente, mentre i messaggi r dipendono da quelli q riferiti allo stesso istante temporale. Quindi possiamo avviare l'algoritmo inizializzando tutti i messaggi q al valore zero.

Mostriamo in Figura 3.2 un grafo fattorizzato su cui illustriamo un semplice esempio per chiarirne il funzionamento. All'istante iniziale ciascun nodo-variabile invia il messaggio q^0 inizializzato a zero, non appena il nodo U_1 riceve almeno due messaggi, supponiamo $q^0_{1\to 1}$ e $q^0_{3\to 1}$, allora esso può inviare il messaggio $r^0_{1\to 2}$ calcolato mediante l'equazione:

$$r^0_{i\to j}(x_2) = \max_{x_1,x_3}\Big(U_1(x_1,x_2,x_3) + q^0_{1\to 1}(x_1) + q^0_{3\to 1}(x_3)\Big)$$

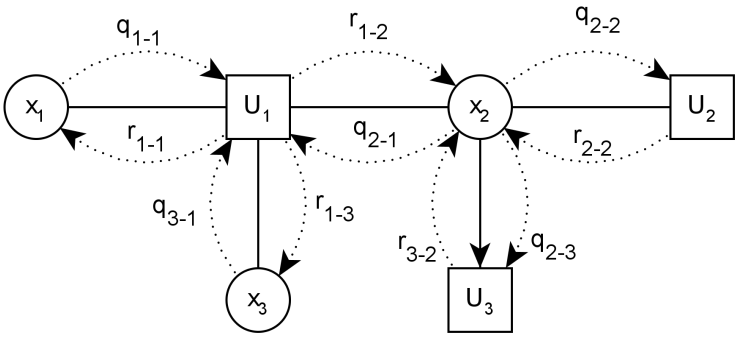


Figura 3.2: Grafo fattorizzato d'esempio

Allo stesso modo vengono calcolati tutti i messaggi di tipo r anche dagli altri nodi-funzione. Passiamo quindi all'analisi del passo successivo in cui vengono calcolati i messaggi q. Per esempio consideriamo il nodo-variabile x_2, esso, per poter inviare il messaggio $q^1_{2\to1}$, deve aver ricevuto sia $r^0_{2\to2}$ che $r^0_{3\to2}$, quindi otteniamo:

$$q^1_{2\to1} = r^0_{2\to2} + r^0_{3\to2}$$

I nodi-variabile, inoltre, ogni volta che nell'iterazione t ricevono tutti i messaggi dai propri vicini calcolano la funzione marginale mediante la formula:

$$z^t_j(x_j) = \sum_{i\in M(j)} r^t_{i\to j}(x_j)$$

Quando tale funzione marginale risulta la stessa per due iterazioni successive i nodi che l'hanno calcolata si disattivano, quindi i nodi ad essi collegati assumono come valido l'ultimo valore ricevuto. L'algoritmo ha termine quando tutti i nodi-variabile risultano disattivati, cioè in altre parole quando tutte le funzioni marginali sono invarianti.

La seconda fase prevede solamente il calcolo della frontiera di Pareto determinata come l'intersezione tra gli insiemi delle soluzioni contenute in ciascuna funzione marginale.

Capitolo 4

Progetto logico della soluzione del problema

In questo capitolo descriviamo i concetti fondamentali usati nel prosieguo della trattazione suddividendoli, per chiarezza espositiva, nelle tre fasi previste, cioè generazione degli scenari, simulazione e valutazione delle soluzioni.

4.1 Generazione scenari

La prima fase consiste nel generare alcuni scenari plausibili su cui eseguire le simulazioni. Per scenario intendiamo un insieme di corsi d'acqua che affluiscono in un unico fiume principale lungo cui sono presenti gli agenti che distinguiamo in due categorie principali. Individuiamo gli *agenti passivi* i quali non hanno potere decisionale e per questo vengono trascurati nella trattazione della tesi. Esistono poi gli *agenti attivi* che, invece, prendono una decisione in grado variare la quantità d'acqua disponibile a valle di quest'ultimi. Dobbiamo inoltre fare un'ulteriore distinzione tra gli agenti che prelevano acqua dal corso del fiume da quelli che invece controllano direttamente la quantità d'acqua in uscita.

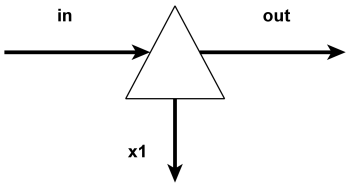


Figura 4.1: Schematizzazione di un agente di tipo città

4.1.1 Tipologie di agenti

Agenti passivi

In uno scenario idrico gli *agenti passivi* sono rappresentati dalle aree ecologiche da preservare, cioè quei siti in cui è necessario avere una portata d'acqua entro certi valori, pensiamo per esempio ai danni che potrebbe causare alla fauna un fiume in secca. Questi agenti, non potendo controllare alcuna variabile, non vengono inseriti nel modello da ottimizzare.

Agenti che prelevano acqua - Città

Per agente prelevatore intendiamo tutti quelli che necessitano acqua per il loro fabbisogno, per esempio possono essere città, distretti agricoli, aree industriali e così via. Per semplificare la nomenclatura indichiamo questi agenti usando il termine *città* e ci riferiamo ad essi mediante lo schema mostrato in Figura 4.1. Quest'ultimo prevede una portata d'acqua in ingresso denominata *in*, una quantità che l'agente preleva coincidente con la variabile di controllo ad esso riferita quindi a x_1 e una portata in uscita che equivale a $out = in - x_1$.

Agenti che controllano la portata d'acqua in uscita - Dighe

Gli agenti che invece hanno controllo sulla quantità d'acqua in uscita sono tutti quelli che hanno a disposizione un serbatoio, considerato per semplicità infinito, in cui immagazzinare l'acqua a monte. Per snellire la nomenclatura usiamo il termine *diga* per riferirci a questo tipo di agente. Come mostrato dalla Figura 4.2 essi dispongono di una portata in ingresso detta *in*, di un livello

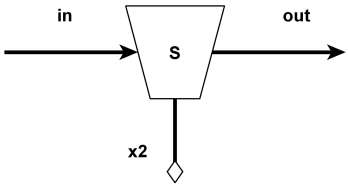


Figura 4.2: Schematizzazione di un agente di tipo diga

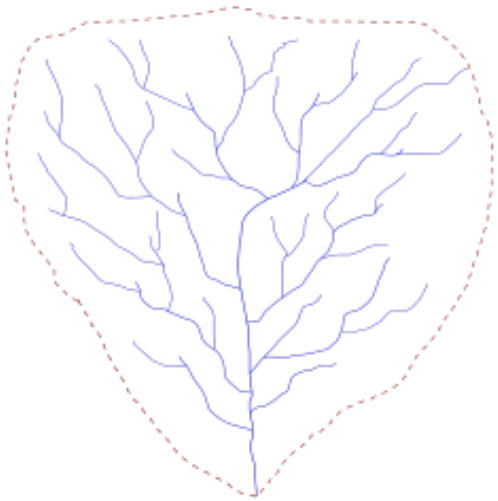


Figura 4.3: Bacino idrografico

chiamato S che indica l'acqua presente nel serbatoio e di una variabile x_2 che controlla la portata in uscita *out*, quindi vale la seguente relazione $out = x_2$.

4.1.2 Topologia dei bacini idrografici

Rendiamo più realistico possibile uno scenario aggiungendo la possibilità di avere alcune ramificazioni lungo il corso del fiume, abbiamo scelto di limitarci al caso in cui più corsi d'acqua affluiscono in un unico fiume e non il viceversa, cioè non è possibile che un fiume si divida, tale situazione, infatti, è abbastanza rara in natura e più complicata da modellizzare. Quindi in generale ciò che otteniamo è un bacino idrografico analogo a quello mostrato in Figura 4.3, in cui lo scorrimento è dall'alto verso il basso.

4.1.3 Vincoli

Rappresentiamo un tale problema utilizzando il formalismo MO-DCOP che ci consente di esprimere sotto forma di vincoli sia i vincoli veri e propri, espressi mediante una funzione di utilità, che le funzioni obbiettivo dei singoli agenti.

Tale formalismo permette inoltre di esplicitare due tipologie differenti di vincoli. I vincoli di tipo fisico o anche detti *hard*, che non possono mai essere violati, sono realizzati impostando a $-\infty$ l'utilità associata al vincolo in caso di violazione. I vincoli di tipo normativo denominati *soft*, che invece possono essere violati, sono attuati mediante un'utilità negativa in caso di violazione. Per raggiungere il nostro scopo dobbiamo costruire il modello formato dai vincoli partendo dallo scenario generato. Il tutto deve essere fatto in modo automatico e per farlo è necessario associare a ciascun agente i vincoli corrispondenti come mostrato nei prossimi paragrafi.

Schema vincolo funzione obbiettivo

Ogni funzione obbiettivo è trasformata immediatamente in un vincolo nel modo seguente: sia $f(x_i)$ la funzione obbiettivo dell'agente $i-$esimo che controlla la variabile x_i, allora il vincolo corrispondente sarà:

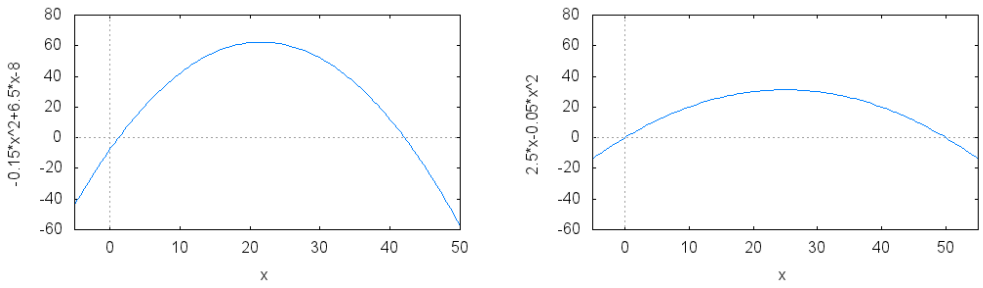
$$C_0(x_i) = f(x_i)$$

Per aderire a quanto mostrato nel Paragrafo 3.4 ciò equivale alla formula seguente:

$$U^i(x_i) = C_0(x_i)$$

Per meglio rappresentare uno scenario reale e per evidenziare la validità dell'algoritmo assumiamo che le funzioni obbiettivo abbiano una forma non lineare. Nello specifico scegliamo funzioni a forma di parabola con concavità verso il basso quindi nella forma dell'Equazione 4.1 con un grafico relativo agli agenti città (Figura 4.4(a)) e diga (Figura 4.4(b)).

$$f_(x) = ax^2 + bx - c \tag{4.1}$$



(a) Parabola delle città (b) Parabola delle dighe

Figura 4.4: Forma delle parabole

Schema vincoli agenti di tipo città

In riferimento alla Figura 4.1 definiamo i vincoli che la presenza di un agente di tipo città comporta, facendo una distinzione tra i vincoli fisici (hard) da quelli normativi (soft).

Trattandosi di un agente che preleva acqua, il vincolo fisico deve impedire che la quantità d'acqua prelevata sia maggiore di quella disponibile, quindi assegniamo un'utilità pari a $-\infty$ nel caso in cui $in - x_1 \leq 0$. Per quanto riguarda i vincoli normativi ne abbiamo due, il primo ha lo scopo di far fluire a valle del nodo una quantità sufficiente ai fabbisogni futuri pari a α_1, quindi stabiliamo un costo in caso di violazione espresso come utilità negativa. Questo costo è direttamente proporzionale alla violazione come mostrato nell'Equazione 4.2.

$$C_1(\mathbf{x}) = \begin{cases} 0 & in(\mathbf{x}) - x_1 \geq \alpha_1 \\ u_1(\mathbf{x}) & \alpha_1 \geq in(\mathbf{x}) - x_1 \geq 0 \\ -\infty & in(\mathbf{x}) - x_1 < 0 \end{cases} \quad u_1(\mathbf{x}) = in(\mathbf{x}) - x_1 - \alpha_1 \tag{4.2}$$

Quest'ultimo vincolo sintetizza quanto esposto fin'ora e si tratta di un vincolo misto, cioè sia hard che soft. L'accorpamento in un unico vincolo è dovuto al fatto che è in funzione delle stesse variabili e se trattato separatamente creerebbe un ciclo sul grafo dei vincoli, il che è conveniente da evitare come spiegheremo in seguito.

Il secondo vincolo normativo è a sè stante ed è interamente rappresentato dall'Equazione 4.3 in cui è presente la quantità dettata dalla norma α_2, cioè il

fabbisogno minimo di ciascun agente di tipo città. Come per il primo vincolo, in caso di violazione, il costo da pagare è direttamente proporzionale alla violazione stessa.

$$C_2(x_1) = \begin{cases} 0 & x_1 \geq \alpha_2 \\ u_2(x_1) & x_1 < \alpha_2 \end{cases} \quad dove \quad u_2(x_1) = x_1 - \alpha_2 \tag{4.3}$$

Schema vincoli agente di tipo diga

Anche per le dighe sono previsti due vincoli, per descriverli faremo riferimento alla Figura 4.2. Il primo, esposto nell'Equazione 4.4, è un vincolo fisico e ha lo scopo di impedire che la quantità di acqua in uscita, che equivale alla variabile di controllo, sia maggiore di quella effettivamente disponibile a monte, cioè deve valere $S + in \geq x_2$. L'utilità prevista in caso di violazione è ancora una volta $-\infty$, zero altrimenti.

$$C_3(\mathbf{x}) = \begin{cases} 0 & x_2 - in(\mathbf{x}) - S \leq 0 \\ -\infty & x_2 - in(\mathbf{x}) - S > 0 \end{cases} \tag{4.4}$$

Il secondo vincolo è invece di tipo normativo, analogamente al caso precedente, serve per garantire che una quantità d'acqua pari a α_3 continui a fluire a valle del nodo, ed è espresso dall'Equazione 4.5. Il costo in caso di violazione è sempre direttamente proporzionale alla violazione stessa.

$$C_4(x_2) = \begin{cases} 0 & x_2 \geq \alpha_3 \\ u_3(x_2) & x_2 < \alpha_3 \end{cases} \quad dove \quad u_3(x_2) = x_2 - \alpha_3 \tag{4.5}$$

4.1.4 Scenario generato

A titolo d'esempio mostriamo alcuni scenari generati mediante il metodo sopra esposto (Figure 4.5, 4.6 e 4.7). Ogni agente è rappresentato da un nodo a forma di rettangolo che può essere di tipo città, in bianco, o di tipo diga, in grigio chiaro. Troviamo, inoltre, i nodi iniziali rappresentati da un quadrato grigio scuro con indicata la portata d'acqua iniziale e i nodi unione rappresentati da un cerchio e che indicano la confluenza di due rami del corso d'acqua. I

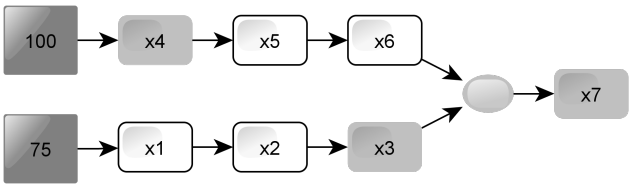


Figura 4.5: Fiume 1

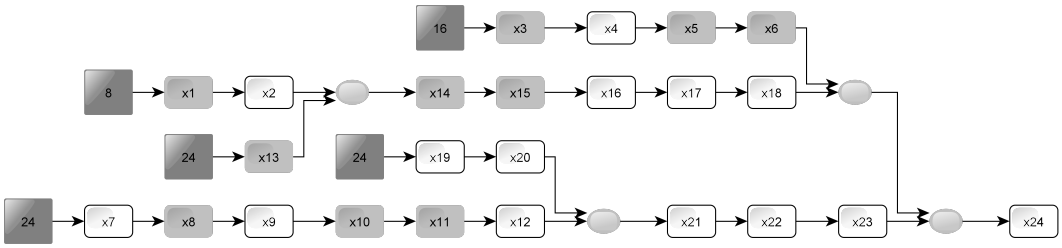


Figura 4.6: Fiume 2

nodi sono tra loro collegati da archi diretti, il verso della freccia indica il fluire dell'acqua, quindi il nodo da cui parte l'arco si trova a monte, mentre il nodo in cui termina la freccia è l'agente a valle.

4.2 Simulazione

Una volta costruiti gli scenari inizia la seconda fase di simulazione che prevede la costruzione del grafo dei vincoli e solo successivamente la simulazione vera e propria.

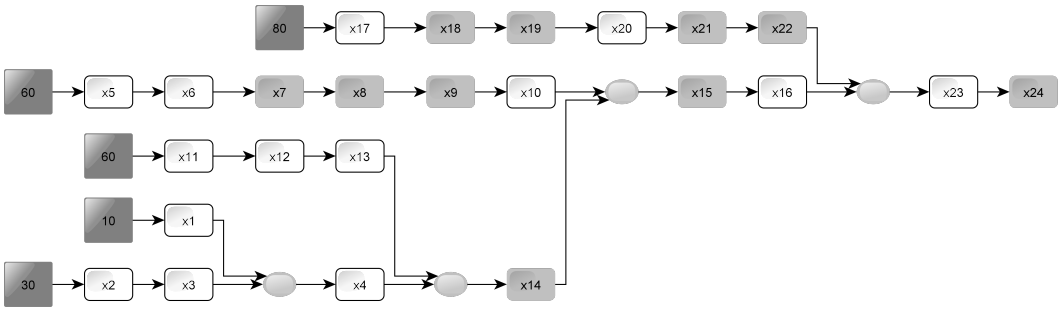


Figura 4.7: Fiume 3

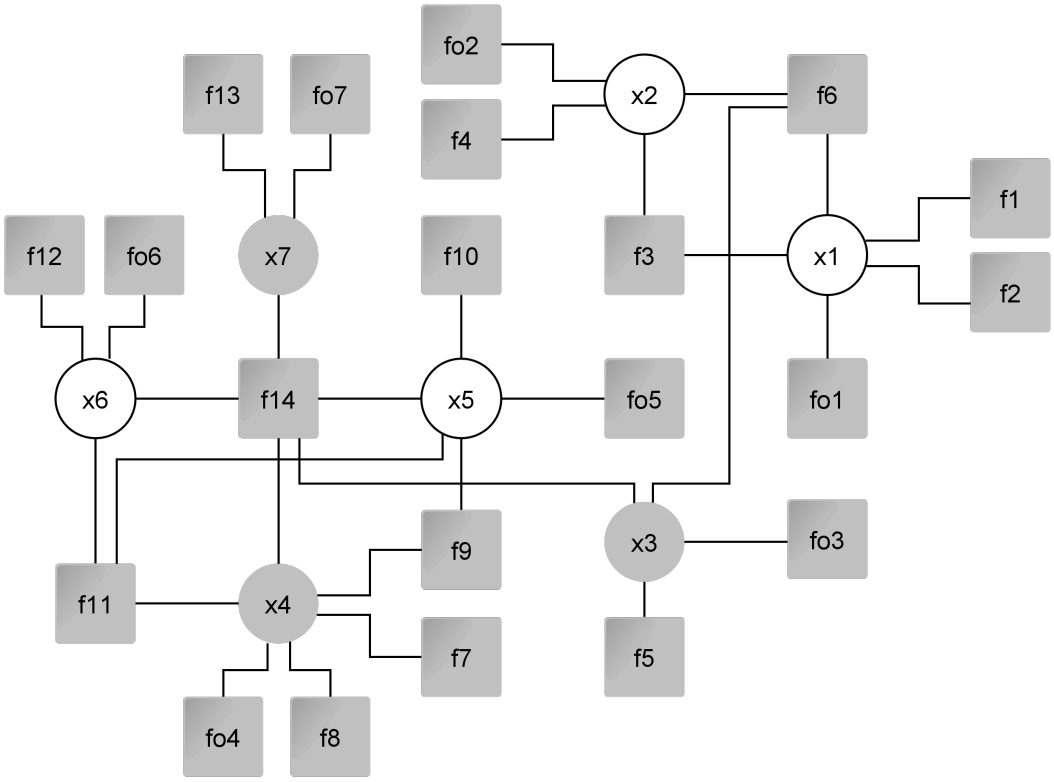


Figura 4.8: Grafo fattorizzato

4.2.1 Grafo fattorizzato

Partendo dal modello formato dai vincoli costruiamo il grafo fattorizzato (o grafo dei vincoli), cioè il grafo che mette in relazioni le variabili, quindi gli agenti, con le funzioni che le vincolano. In Figura 4.8 mostriamo un esempio di questo grafo (riferito al fiume mostrato in Figura 4.5), in cui abbiamo scelto la convenzione di racchiudere le variabili dentro un cerchio e le funzioni di vincolo in un quadrato. I grafi dei vincoli degli scenari ottenuti non sono sempre connessi e sono in generale ciclici. Questi due aspetti costituiscono altrettanti problemi poiché l'algoritmo utilizzato funziona esclusivamente su grafi aciclici e connessi.

Per quanto riguarda la connessione del grafo fattorizzato, la risolviamo escludendo dalla simulazione tutti gli scenari non connessi in quanto poco rilevanti ai fini della valutazione dell'algoritmo. Infatti risolvere un problema di

ottimizzazione vincolata su un grafo composto da n grafi tra loro non connessi equivale a risolvere n problemi di ottimizzazione più semplici, ciò equivale a ridurre di molto la complessità del problema e quindi abbiamo preferito scartare questi scenari per ottenere dei dati più significativi.

Il secondo problema, riguardante la presenza di cicli, è molto più complesso da affrontare, in questo caso però, non è possibile scartare a priori tutti gli scenari che hanno un grafo fattorizzato ciclico poiché altrimenti ci concentreremmo solo su casi poco significativi. Quindi l'unica soluzione attuabile senza cambiare algoritmo di ottimizzazione è quello di rimuovere i cicli tagliando alcuni archi.

4.2.2 Rimozione cicli

Tagliare l'arco che collega la variabile x_i con la funzione di vincolo $f(\mathbf{x_k}, x_i)$ significa eliminare dall'espressione di f la variabile x_i ottenendo la funzione $f'(\mathbf{x_k})$, la funzione ottenuta è diversa da quella di partenza, quindi applicando un taglio si genera un modello simile al precedente ma di fatto diverso. Per come è costruito l'algoritmo B-MOMS dovremmo essere in grado di ottenere un limite superiore dell'errore che commettiamo andando a modificare la struttura del grafo, questo limite dipende però dal valore peggiore che può assumere l'utilità. Se analizzassimo i vincoli presentati nelle Sezioni 4.1.3 e 4.1.3 ci accorgeremmo che solo due di questi possono generare cicli in quanto, nel caso generale, dipendono dall'intero vettore delle variabili nel sistema. Questi due vincoli sono di tipo hard, quindi purtroppo non siamo in grado di fornire un limite superiore per l'errore in quanto l'utilità peggiore equivale sempre a $-\infty$. Ciò implica che le soluzioni ottenute potrebbero essere non fattibili, nel senso che si potrebbero generare delle soluzioni che violano dei vincoli fisici, ad esempio prelevando dal fiume più acqua di quanta ne sia disponibile. Approfondiremo questa problematica nel corso della trattazione, per ora ci limitiamo a illustrare come ottenere un grafo aciclico.

Tale grafo può essere ricavato sia, come suggerito in letteratura [4], usando un approccio distribuito applicando l'algoritmo *GHS* [6], sia mediante il

classico algoritmo *MST (Minimum Spanning Tree)* che però è centralizzato. Abbiamo scelto quest'ultimo per semplicità dato che, nella fase attuale, l'implementazione dell'algoritmo B-MOMS è ancora immatura e non adatta ad essere utilizzata in uno scenario realmente distribuito. Nel caso in cui quest'ultimo raggiungesse maturità tale da essere utilizzabile in applicazioni reali, andrebbe sostituita l'attuale implementazione con una realizzazione dell'algoritmo GHS.

Prima di poter applicare l'algoritmo MST dobbiamo associare ad ogni arco un peso opportuno, mostriamo di seguito alcune tecniche implementate.

Uniform (U)

Questa tecnica è quella più semplice, è relativa al già citato lavoro di Delle Fave et al. [4] in cui si suggerisce di favorire l'eliminazione dei vettori dominati rispetto a quelli non dominati. In presenza di vincoli fisici ciò non è possibile in quanto gli archi di tali vincoli sono tutti importanti allo stesso modo, quindi è sufficiente associare a ciascun arco lo stesso peso, per esempio pari a 1. In Figura 4.9 mostriamo un possibile grafo aciclico ottenuto applicando MST sul grafo ciclico di Figura 4.8. Mentre in Figura 4.10 forniamo il corrispondente grafo comprensivo dei pesi in cui, per chiarezza, sono stati omessi i vincoli unari e vengono rappresentati mediante una linea tratteggiata gli archi tagliati dall'algoritmo MST.

Ranking Nodes (RN)

In quest'altra tecnica poniamo il peso dell'arco che collega il nodo *x1* al nodo *f1* equivalente alla somma del numero di archi uscenti da *x1* e da *f1*, quindi otteniamo il grafo con i rispettivi pesi, come evidenziato in Figura 4.11. Per esempio il nodo variabile x_1 è collegato a 5 nodi funzione, tra cui f_6, quest'ultimo è invece collegato a 3 nodi variabile. Il peso dell'arco $< x_1 : f_6 >$ è dunque 8. Mediante il solito algoritmo di taglio possiamo ottenere il grafo di Figura 4.12.

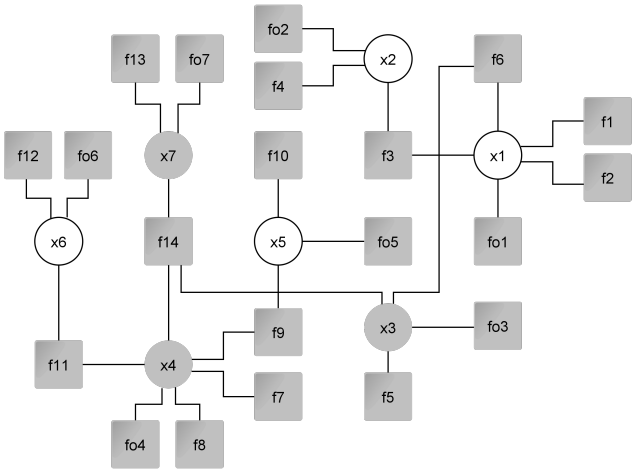


Figura 4.9: Grafo fattorizzato aciclico - Uniform, Ranking Upstream e Constraint Less

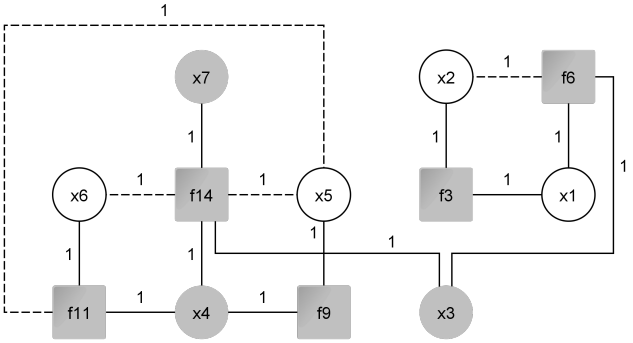


Figura 4.10: Grafo fattorizzato con pesi - Uniform

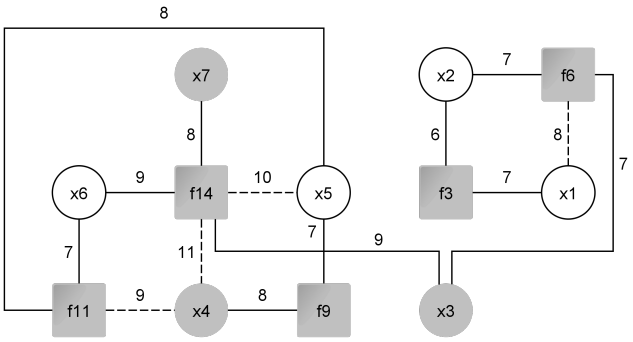


Figura 4.11: Grafo fattorizzato con pesi - Ranking Nodes

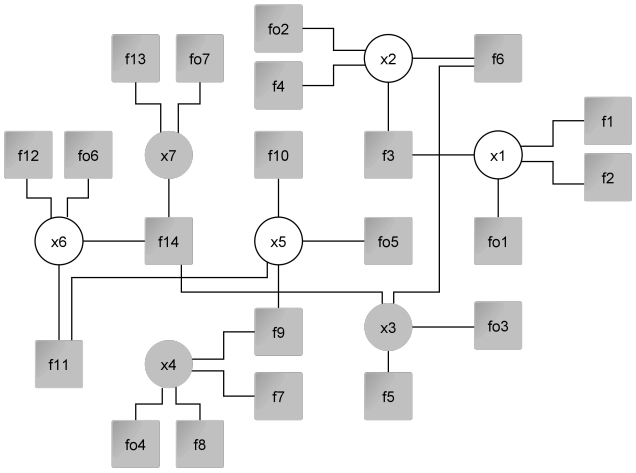


Figura 4.12: Grafo fattorizzato aciclico - Ranking Nodes

Ranking Upstream (RU)

Nella terza metodologia associamo un peso basso agli archi collegati alle variabili poste a monte, e via via più alto spostandoci a valle. Quindi, dato che l'algoritmo è di minimizzazione, i primi saranno più importanti, un esempio di grafo pesato in questo modo è mostrato in Figura 4.13. Alla base di questa euristica c'è il tentativo di ottenere degli assegnamenti relativi ai nodi a monte meno soggetti a errori rispetto di quelli a valle, questo perché i vincoli generati dai nodi a valle dipendono dalle variabili a monte e non viceversa. In riferimento allo scenario denominato *Fiume 1*, della Figura 4.5, i nodi x_1 e x_4 si trovano a monte quindi avranno entrambi un punteggio pari a 1, i nodi x_2 e x_5, invece, un punteggio di 3, mentre x_3 e x_6 pari a 5, infine il nodo più a valle, cioè x_7, il punteggio massimo pari a 7. Pesiamo quindi tutti gli archi uscenti dal nodo variabile x_i con il punteggio di quest'ultimo. L'applicazione di MST ci consente di ottenere il grafo mostrato in Figura 4.9.

Constraint More (CM)

L'idea generale dietro a questa tecnica è quella di far sì che il grafo risultante sia una sovrastima del modello originale. Spieghiamo brevemente il funzionamento considerando i vincoli che possono generare cicli, cioè C_1 e C_3 esposti

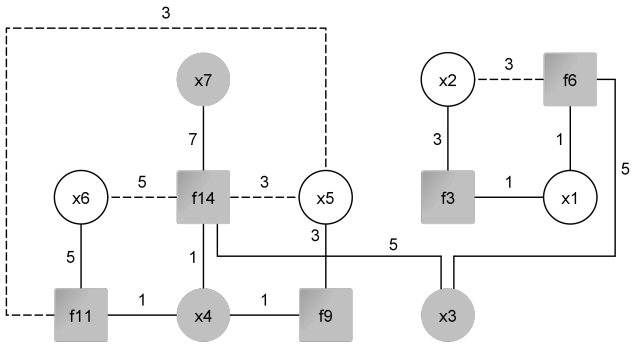


Figura 4.13: Grafo fattorizzato con pesi - Ranking Upstream

rispettivamente nelle Sezioni 4.1.3 e 4.1.3. Questi vincoli sono rispettati se valgono le seguenti disequazioni:

$$\sum_i x_i - \sum_j x_j + \sum_l q_l - x_1 \geq \alpha_1 \quad \textit{con } \alpha_1 \textit{ e } q_l \textit{ costanti } \forall_l$$

nel caso di C_1 e:

$$x_2 - \sum_i x_i + \sum_j x_j - \sum_l q_l - S \leq 0 \quad \textit{con } S \textit{ e } q_l \textit{ costanti } \forall_l$$

per C_3. Quindi possiamo scriverle entrambe nella forma:

$$\sum_i x_i - \sum_j x_j + \sum_l k_l \geq 0 \quad \textit{con } k_l \textit{ costanti } \forall_l$$

Dato che le variabili appartengono tutte a un dominio positivo o nullo allora tutte le variabili x_i sono positive, mentre quelle contrassegnate con x_j sono negative. Tagliare una variabile positiva significa rendere il vincolo più difficile da soddisfare, nel senso che ci sono meno combinazioni di assegnamenti che soddisfano il vincolo. Viceversa se tagliamo una variabile di segno negativo rendiamo il vincolo più semplice da soddisfare. Quindi in questa tecnica, in cui preferiamo il taglio di variabili che vincolano di più un dato vincolo, assegniamo un peso alto alle variabili di segno positivo, basso altrimenti. Ricordiamo infatti che l'algoritmo MST cerca l'albero di copertura che minimizza il peso degli archi.

In Figura 4.14 mostriamo il grafo fattorizzato in cui sono presenti i pesi calcolati con questa tecnica. A titolo d'esempio applichiamo quanto fin'ora

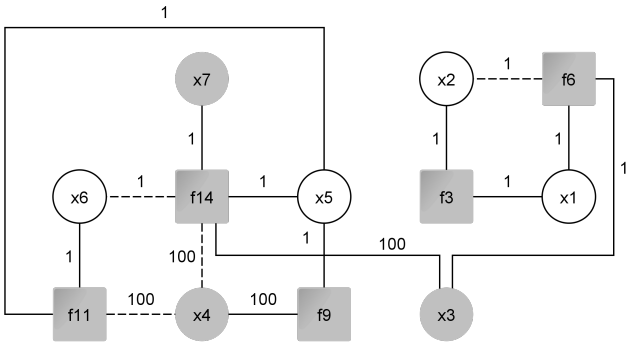


Figura 4.14: Grafo fattorizzato con pesi - Constraint More

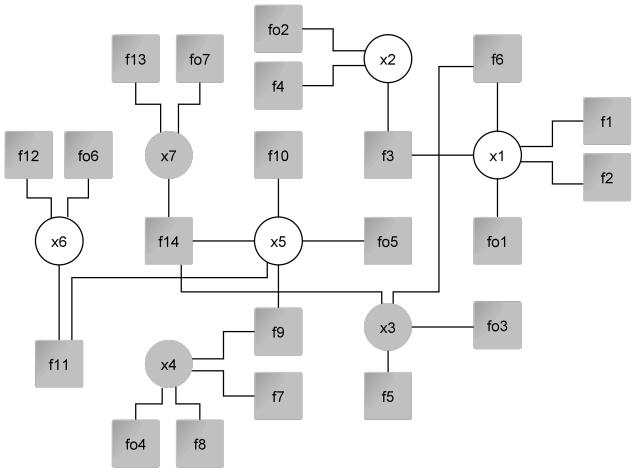


Figura 4.15: Grafo fattorizzato aciclico - Constraint More

esposto al vincolo descritto nell'Equazione 4.6, in cui $u = x_4 - x_5 - x_6 - 9.0$ rappresenta l'utilità negativa assegnata in caso di violazione. Quindi, dato che x_4 è di segno positivo, associamo un peso alto, per esempio 100, all'arco che lo collega a f_{11}, mentre a x_5 e x_6 assegniamo un peso basso, per esempio 1 in corrispondenza degli archi che li collegano a f_{11}.

$$f_{11}(x_4, x_5, x_6) = \begin{cases} 0 & x_4 - x_5 - x_6 \geq 9 \\ u(x_4, x_5, x_6) & 9 \geq x_4 - x_5 - x_6 \geq 0 \\ -\infty & altrimenti \end{cases} \tag{4.6}$$

Il grafo aciclico completo è invece mostrato in Figura 4.15.

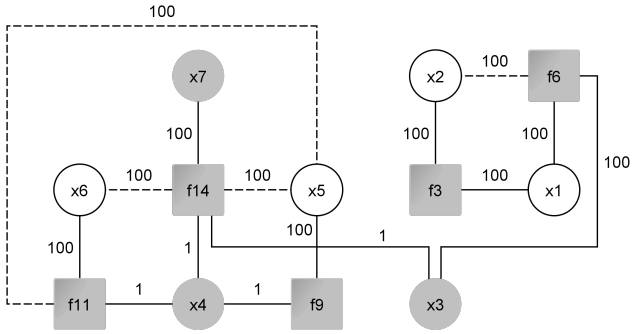


Figura 4.16: Grafo fattorizzato con pesi - Constraint Less

Constraint Less (CL)

Questa tecnica è la duale della precedente, infatti, mira a ottenere un grafo che tende ad essere una sottostima del modello originale, quindi vale tutto ciò che abbiamo esposto nel Paragrafo 4.2.2 ad eccezione per i pesi che ovviamente sono assegnati in modo opposto. Mostriamo in Figura 4.16 il grafo fattorizzato semplificato mentre il grafo aciclico completo è in Figura 4.9.

4.3 Valutazione delle soluzioni

Ottenuto il grafo aciclico possiamo procedere alla simulazione vera e propria eseguendo la versione modificata dell'algoritmo B-MOMS. Ciò è svolto per ogni tecnica di assegnamento dei pesi prevista, quindi per ciascuna di esse viene generata una frontiera di Pareto, di conseguenza dobbiamo procedere a una valutazione dei risultati confrontando tali frontiere di Pareto.

4.3.1 Fattibilità delle soluzioni

A causa della modifica dei vincoli fisici per rendere il grafo fattorizzato aciclico è possibile che alcune soluzioni trovate non siano fattibili, quindi dobbiamo, in fase di valutazione, determinare quali soluzioni sono fattibili e le indicheremo con $\mathcal{S}_F$ e quali no, indicate con $\mathcal{S}_U$, mentre denotiamo l'insieme di tutte le

soluzioni con $\mathcal{S} = \mathcal{S}_F \cup \mathcal{S}_U$. Ciò è svolto grazie a un controllo di fattibilità effettuato sul modello originale.

4.3.2 Frontiera di riferimento

Nelle varie misure prestazionali utilizzeremo il concetto di frontiera di riferimento, con la quale intendiamo quella frontiera di Pareto ottenuta mediante un algoritmo di forza bruta che valuta ciascun vincolo per ogni valore del dominio di ogni variabile e la indicheremo con $\mathcal{S}^*$. Si tratta quindi di un algoritmo molto oneroso computazionalmente, ipotizzando uno scenario composto da a agenti, ciascuno con d elementi nel dominio e f variabili di vincolo, ha una complessità temporale asintotica pari a $\mathcal{O}(f \cdot d^a)$.

Questo riferimento dovrebbe dominare la frontiera con cui si sta confrontando, in generale ciò però non accade sempre a causa della modifica ai vincoli fisici derivante dai tagli effettuati sul grafo fattorizzato per renderlo aciclico. La conseguenza è che le misure prestazionali potrebbero non avere la stessa efficacia di quella che avrebbero se confrontassero problemi equivalenti.

4.3.3 Generational Distance

La metrica *generational distance* fornisce una prima misura prestazionale di una frontiera di Pareto. Ha senso solo se la frontiera da misurare ha più di 2 elementi, nel nostro caso se $|\mathcal{S}_F| > 2$. Per ciascun punto di $\mathcal{S}_F$ calcoliamo la distanza con il più vicino dei punti di $\mathcal{S}^*$. La metrica equivale quindi alla media di queste distanze. Questo tipo di valutazione ha il difetto di non essere in grado di catturare il concetto di copertura della frontiera di riferimento.

Mostriamo in Figura 4.17 un esempio relativo a un caso semplice in due dimensioni in cui i punti neri fanno parte di $\mathcal{S}^*$, mentre i punti grigi rappresentano $\mathcal{S}_F$, infine i segmenti indicano le distanze precedentemente descritte.

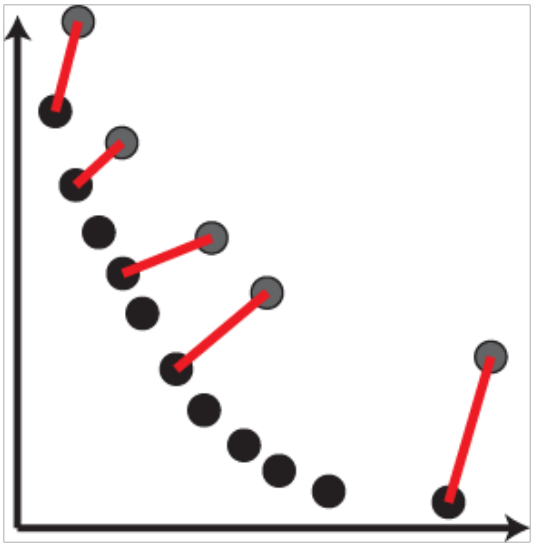


Figura 4.17: Generational Distance

4.3.4 Hypervolume

Utilizziamo la metrica Hypervolume [19] per valutare ciascuna frontiera di Pareto. Questo indice sintetizza in un unico valore sia la vicinanza di una frontiera di Pareto a una frontiera di riferimento sia il grado di copertura dello spazio delle soluzioni. Infatti esso misura il volume della porzione dominata nello spazio degli obbiettivi, questo risultato è normalizzato all'interno degli estremi della frontiera di riferimento quindi assume un valore compreso nell'intervallo $[0, 1]$, più è vicino a 1 migliore è la soluzione misurata. Ha senso calcolarla solo su $\mathcal{S}_F$ quando $|\mathcal{S}_F| > 2$.

Per esempio se consideriamo un problema di minimizzazione in due dimensioni, allora otteniamo le aree descritte in Figura 4.18, quella più scura indica l'area riferita alla miglior soluzione possibile, mentre quella più chiara l'area della soluzione approssimata, la metrica in questione misura quindi l'area in grigio rapportata all'area in nero. Estendendo il diagramma al caso multidimensionale le aree diventano ipervolumi da cui il nome della metrica.

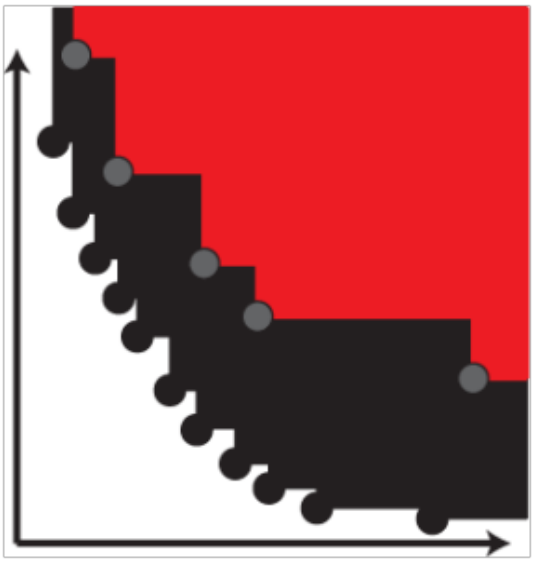


Figura 4.18: Hypervolume

4.3.5 Grado di violazione

Abbiamo introdotto un'altra metrica in grado di misurare l'eventuale grado di violazione dei vincoli, questa misura è calcolata sull'insieme $\mathcal{S}$ e non necessita di nessuna frontiera di riferimento. Calcoliamo cioè la violazione relativa per ciascun vincolo che è stato tagliato dal modello ciclico originale in base a due considerazioni, la prima è che tutti i vincoli sono lineari del tipo:

$$C(\mathbf{x}_a) = \sum_{i \in \mathbf{x}_a} x_i + k \geq 0$$

dove k è una costante e $\mathbf{x}_a \subseteq \mathbf{x}$ è il vettore delle variabili da cui C dipende. La seconda è che i domini sono tutti finiti e limitati, quindi otteniamo il valore massimo di violazione mediante la minimizzazione del vincolo stesso:

$$c^* = \min_{\mathbf{x}_a} C(\mathbf{x}_a)$$

Valutiamo poi la violazione effettiva di ciascun punto della frontiera di Pareto mediante la seguente equazione $c' = C(\mathbf{x'}_a)$, dove $\mathbf{x'}_a$ è l'assegnamento effettivo della soluzione in esame. La violazione relativa risulta quindi pari a $v = \frac{c'}{c^*}$. Siano n il numero totale dei punti sulla frontiera e m il numero di vincoli che

subiscono un taglio, allora in totale avremo $n \cdot m$ valori di violazione relativa che dobbiamo aggregare, quindi ne calcoliamo media e valore massimo.

Capitolo 5

Implementazione del sistema

Il software che implementa il sistema descritto in questa tesi è interamente scritto usando il linguaggio di programmazione *Java*, in questo capitolo ne forniamo una breve descrizione sia del suo funzionamento che dei vari moduli che lo compongono.

5.1 Funzionamento del simulatore

Il simulatore è basato su tre fasi principali che descriviamo brevemente nei paragrafi seguenti.

5.1.1 Generazione automatica

La prima fase è a sua volta suddivisa in due parti schematizzate in Figura 5.1 e mostrate di seguito.

Bacino idrico

La prima di queste si occupa di produrre alcuni sistemi idrici basandosi su dei parametri di generazione che consentono di dimensionare la grandezza del bacino e la sua complessità. Un sistema idrico è costituito da dei nodi disposti in modo da formare una struttura dati ad albero. Ciascun nodo fa riferimento

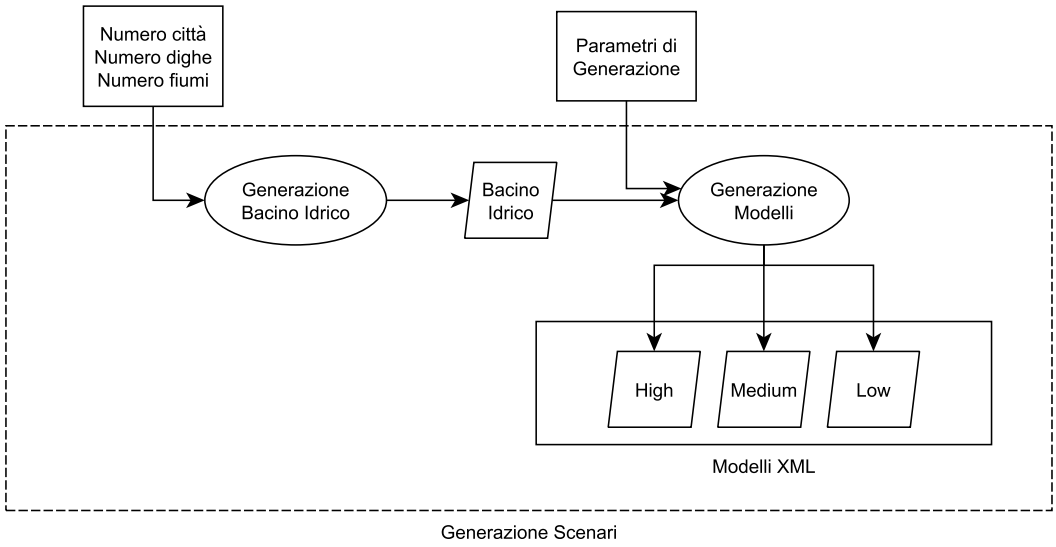


Figura 5.1: Schema della prima fase

ad uno schema comune che viene specializzato in base alle caratteristiche del nodo stesso, quindi, per indicare gli agenti usiamo un nodo città e uno diga, mentre adoperiamo un nodo iniziale per indicare l'inizio di un corso d'acqua avente portata iniziale nota e un nodo unione per gestire le confluenze di più fiumi.

Modello

Per ogni sistema idrico prodotto generiamo tre tipologie di scenari aventi differenti disponibilità d'acqua, ciò ci consente di simulare le condizioni di piena, morbida e magra di un fiume. Grazie a questa distinzione siamo in grado di produrre tre tipi di modelli traducendo mediante i vincoli descritti nella Sezione 4.1.3 ogni sistema idrico.

La generazione del modello necessita inoltre la scelta degli elementi del dominio che vengono posti lungo l'intervallo in cui la parabola della funzione obbiettivo assume valori positivi.

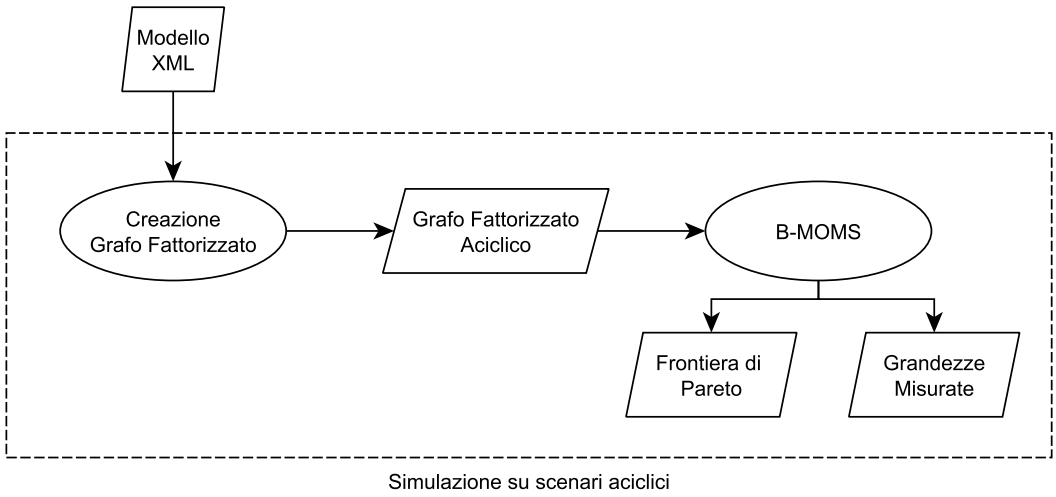


Figura 5.2: Schema della seconda fase per scenari inizialmente aciclici

5.1.2 Simulazione

La seconda fase consiste nell'applicazione dell'algoritmo B-MOMS e di conseguenza della misura di varie grandezze utili in fase di valutazione. Questa operazione può essere fatta immediatamente sul modello in questione se il suo grafo fattorizzato è aciclico come mostrato dallo schema in Figura 5.2. Altrimenti necessita delle operazioni descritte nel prossimo paragrafo e schematizzate in Figura 5.3.

Rimozione cicli

Nel caso in cui il grafo fattorizzato è ciclico applichiamo le tecniche di assegnamento dei pesi descritte nella Sezione 4.2.2 e quindi otteniamo il relativo grafo aciclico mediante l'applicazione dell'algoritmo MST.

5.1.3 Valutazione

L'ultima fase si occupa di calcolare tutte quelle metriche necessarie alla valutazione dell'algoritmo usato, occorre ancora una volta fare una distinzione tra i modelli con grafo fattorizzato aciclico dagli altri. Per i primi valutiamo esclusivamente l'ottimalità della soluzione confrontandola con quella ottenuta

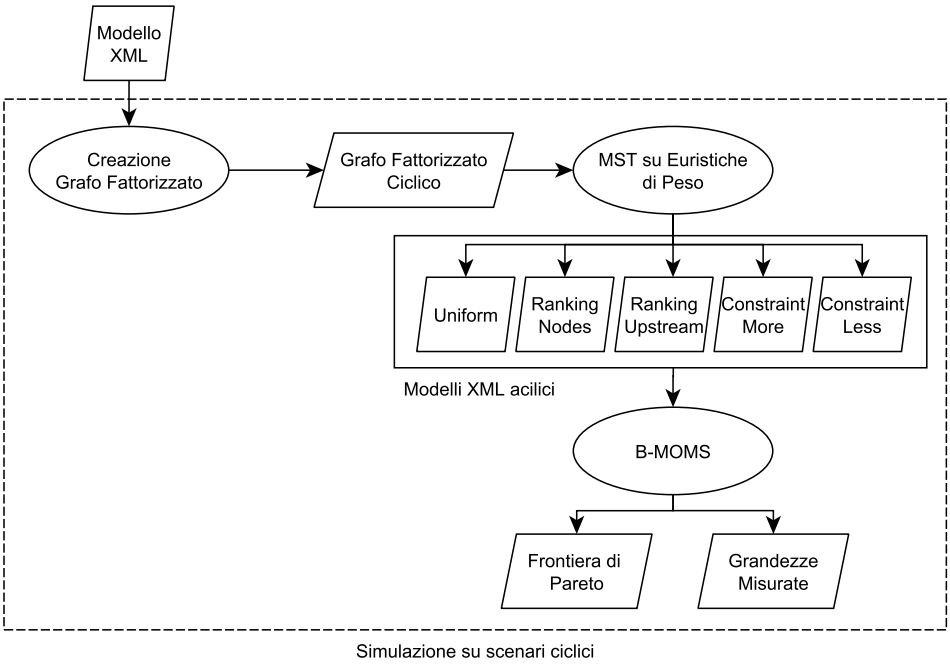


Figura 5.3: Schema della seconda fase per scenari inizialmente ciclici

mediante l'algoritmo di forza bruta e confrontiamo i tempi di esecuzione di questi due approcci, come indicato nello schema di Figura 5.4.

Per i modelli con grafo ciclico invece dobbiamo stabilire innanzitutto la fattibilità delle soluzioni e la frontiera di riferimento così da poter calcolare le due metriche considerate, cioè hypervolume e generational distance. Un'altra metrica che consideriamo in questa fase è il grado di violazione nel caso di soluzioni non fattibili, schematizziamo tutto ciò in Figura 5.5.

5.2 Parametri di generazione

Otteniamo una grande varietà di scenari grazie alla possibilità di agire sui vari parametri del sistema, per limitare le casistiche abbiamo dovuto effettuare delle scelte su quali parametri lasciare costanti e quali invece variare. Per mostrare la validità dell'approccio, nel caso di parametri costanti, abbiamo eseguito i test utilizzando un ulteriore valore, indicato con un apice, per ciascun parametro,

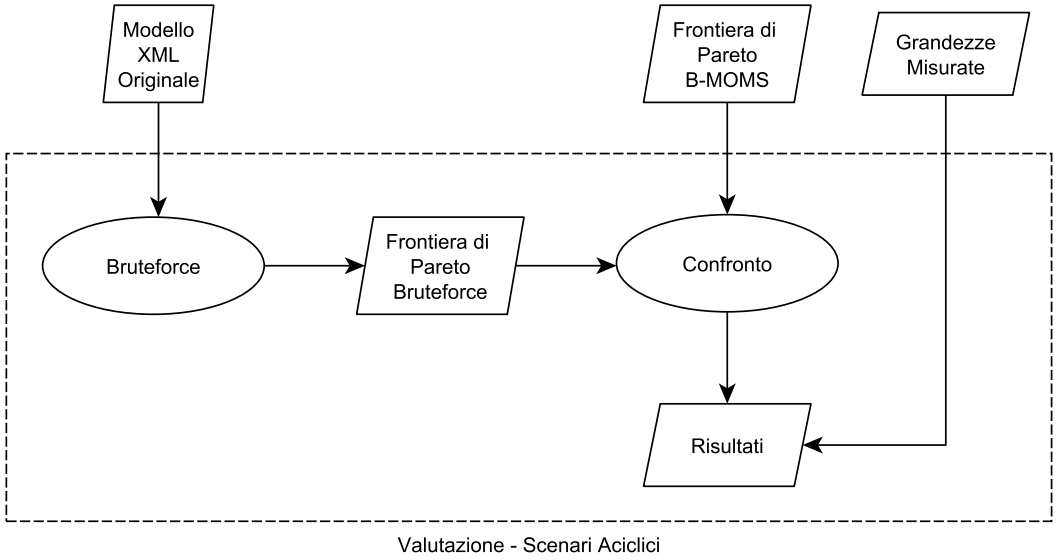


Figura 5.4: Schema della terza fase per scenari inizialmente aciclici

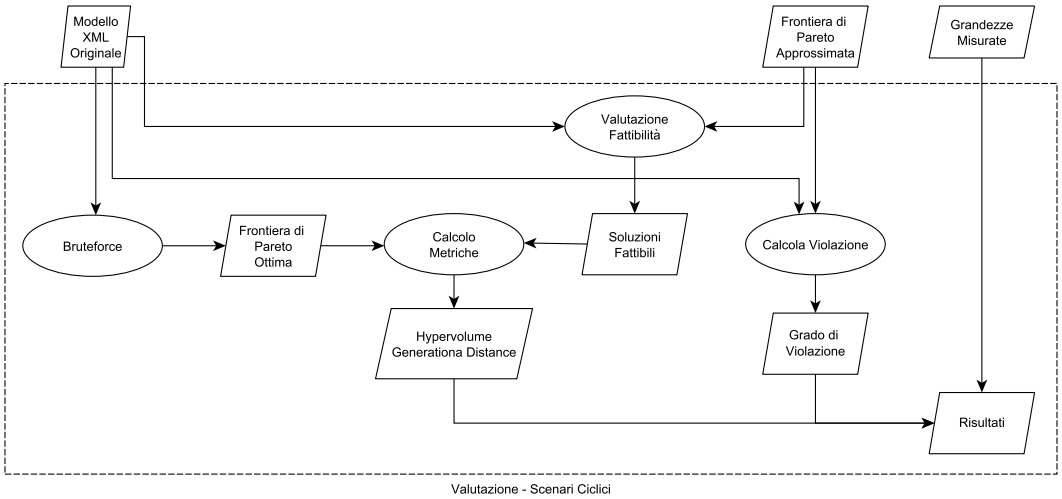


Figura 5.5: Schema della terza fase per scenari inizialmente ciclici

mediante i quali otteniamo risultati analoghi. Richiamiamo di seguito tutti i parametri su cui possiamo agire.

- *Numero di città*, è un intero positivo che facciamo variare nel corso delle simulazioni, è utile poterlo cambiare in quanto consente di determinare, insieme al numero di dighe, la complessità del sistema.
- *Numero di dighe*, è analogo a quanto previsto per il caso precedente.
- *Numero di agenti*, equivale alla somma del numero di città e di dighe, essendo un parametro derivato dai primi due è anch'esso un intero positivo, indica in modo abbastanza significativo la complessità del sistema.
- *Numero di fiumi*, anche questo è un intero positivo variabile e rappresenta il numero di corsi d'acqua presenti nel sistema.
- *Numero di elementi del dominio*, quantifica la dimensione del dominio di ciascuna variabile, assumiamo che questo valore sia lo stesso per tutti gli agenti e ovviamente sia un numero intero maggiore di zero. Anche questo parametro è fondamentale per determinare la complessità di un sistema, rappresenta infatti la base della legge esponenziale che determina la complessità computazionale asintotica dell'algoritmo di ottimizzazione.
- *Abbondanza di acqua*, è un parametro di tipo stringa che indica la quantità d'acqua presente nel fiume, può assumere uno di questi valori $\langle Low, Medium, High \rangle$ posti in ordine crescente di presenza di acqua nell'intero sistema.
- *Coefficienti delle parabole*, sono la terna dei coefficienti reali $\langle a, b, c \rangle$ di una parabola nella forma $ax^2 + bx + c = 0$. Distinguiamo due possibili terne di valori, una relativa alle città $\langle -0.15, 2.5, -8 \rangle$ e una per le dighe $\langle -0.06, 6.5, 0 \rangle$. Scegliamo questi valori per coerenza rispetto al lavoro di Yang et. al[17] e li teniamo costanti in tutte le simulazioni (i valori alternativi sono $\langle -0.1, 9, -12 \rangle'$ per le città e $\langle -0.08, 4, -3 \rangle'$ per le dighe).

- *Portata d'acqua iniziale del fiume i-esimo* Q_i, è il valore reale della portata d'acqua disponibile al nodo più a monte del fiume i-esimo, è un termine variabile ma non si può controllare direttamente, infatti, dipende dal numero di agenti che deve soddisfare e dall'abbondanza di acqua prevista. É determinata dalla regola $Q_i = nk_j$ dove n rappresenta il numero di agenti che sono direttamente collegati a questo fiume fino al punto in cui affluisce in un corso d'acqua di portata maggiore, k_j invece dipende dalla tipologia prevista, usiamo i seguenti valori: $k_{Low} = 4$, $k_{Medium} = 10$ e $k_{High} = 25$ ($k'_{Low} = 3$, $k'_{Medium} = 8$ e $k'_{High} = 20$).

- *Portata minima in uscita da un nodo*, indicato con α_2 o α_3, è un valore reale che teniamo costante. Rappresenta la quantità di acqua che deve continuare a fluire a valle di un nodo, è un parametro normativo e vale $\alpha_2 = \alpha_3 = 9$ ($\alpha'_2 = \alpha'_3 = 5$).

- *Quantità d'acqua presente nelle dighe* S_i, è anch'esso un valore reale che può variare in base all'abbondanza di acqua, può assumere i seguenti valori: $S_{Low} = 3$, $S_{Medium} = 8$ e $S_{High} = 10$ ($S'_{Low} = 2$, $S'_{Medium} = 5$ e $S'_{High} = 8$).

- *Fabbisogno minimo dei nodi città*, indicato con α_1, assume valore costante pari a $\alpha_1 = 12$ ($\alpha'_1 = 8$) è basato sui valori proposti nel lavoro [17].

5.3 Moduli del sistema

In questa sezione illustriamo i vari moduli che compongono il sistema e uno schema che evidenza le interazioni tra di essi in Figura 5.6.

5.3.1 Generazione fiume

Questo modulo fornisce i vari metodi per la creazione dei sistemi idrici, di fatto permette la generazione parametrica di una struttura dati ad albero che rappresenta uno scenario idrico i cui nodi sono gli agenti del sistema. Questi nodi fanno tutti riferimento allo schema di una classe astratta che raccoglie in

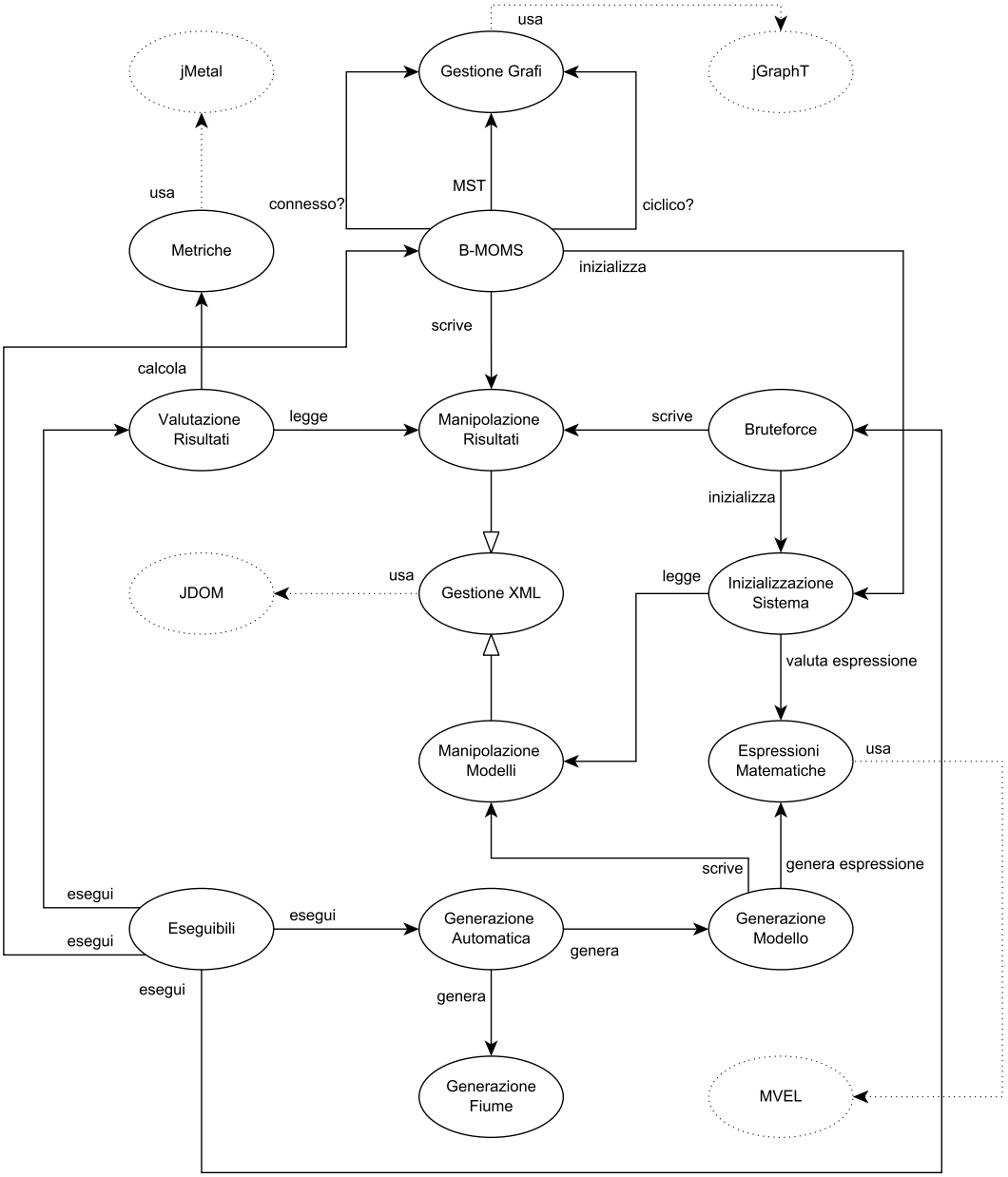


Figura 5.6: Schema dei moduli e delle loro interazioni

sè gli attributi e i metodi comuni. Quest'ultima viene specializzata in modo differente a seconda del tipo di agente o a seconda che si tratti di nodi utilità come quello che specifica la portata iniziale di un fiume o i nodi che permettono la confluenza di due o più corsi d'acqua in uno unico.

5.3.2 Generazione modello

In questa componente viene svolta la traduzione dal sistema idrico al relativo modello composto prevalentemente dai vincoli del sistema. La traduzione produce tre tipologie di scenari, ciascuno tenta di simulare il corso d'acqua con differenti condizioni relativamente alla quantità d'acqua disponibile lungo il letto del fiume. Questi scenari sono denominati *High, Medium e Low*, rispettivamente rappresentano condizioni di piena, morbida e magra di un fiume.

5.3.3 Gestione grafi

Questo modulo permette la creazione, manipolazione ed esportazione di grafi, è basato sulle librerie fornite dal progetto open source *JGrahpT* [21]. Per esempio usiamo questo modulo per la creazione del grafo fattorizzato a partire dal modello descritto mediante file XML, e, in caso di grafo ciclico, viene usato per la fase di bilanciamento del grafo mediante pesi e la successiva applicazione dell'algoritmo MST. Ciascun grafo presente nel sistema viene esportato nel formato standard *graphml* così da avere a disposizione in maniera del tutto automatizzata una loro visualizzazione.

5.3.4 Gestione XML

La gestione dei file XML è una parte essenziale del software in questione, mediante la quale possiamo gestire in modo persistente gli scenari generati e tenere traccia dei risultati delle varie simulazioni. L'intero modulo è costruito attorno alle librerie *JDOM* [20] le quali forniscono una serie di strumenti per gestire questi file in modo rapido ed efficace.

Distinguiamo due casistiche principali descritte nelle sezioni seguenti.

Manipolazione dei modelli

Ogni modello è salvato in un file XML costituito da due parti principali, la prima relativa alle variabili e la seconda che contiene i vincoli. Come è possibile vedere nell'esempio mostrato in Appendice A.1 ad ogni variabile è associato un nome univoco e un dominio costituito da un insieme finito di valori.

La parte relativa ai vincoli è invece un po' più articolata, infatti, ogni vincolo ha un nome univoco, un insieme di variabili da cui esso dipende (elencate nell'elemento *scope*) e la rappresentazione della funzione del vincolo scomposta in due parti. La prima di queste contiene la parte destra del vincolo, cioè le condizioni (elemento *booleans*), mentre la seconda contiene il valore assunto dalla funzione, cioè le parte sinistra del vincolo, se la corrispondete condizione è verificata (elemento *utilities*).

Sia i vincoli che le variabili contengono inoltre l'attributo chiamato *pos* che indica la posizione alla quale il vincolo o la variabile fa riferimento all'interno del vettore obbiettivo, così come mostrato nella Sezione 3.4.

Siamo quindi in grado di fornire i metodi utili alla creazione di tale struttura partendo da un qualsiasi scenario e la sua eventuale modifica.

Manipolazione dei risultati

Tutti i valori ottenuti in una simulazione vengono immagazzinati in un file XML per poter essere disponibili nella fase successiva di valutazione dei dati. La struttura di questo file, mostrata in Appendice A.2, è semplice e autoesplicativa, raccoglie informazioni in merito ai tempi di esecuzione, alla soluzione trovata e alle metriche calcolate.

5.3.5 Espressioni matematiche

Tutte le parti relative alla manipolazione delle espressioni matematiche sono gestite in questo modulo che si avvale dell'ausilio delle librerie *MVEL*[22]. Queste ultime sono open source e scritte in Java, forniscono delle funzioni che

ci permettono di valutare varie espressioni matematiche scritte sotto forma di stringa.

5.3.6 Metriche

Questo modulo non introduce molte funzionalità ma ci permette di sfruttare le librerie Java *jMetal*[5] che consentono di calcolare direttamente le due metriche usate nella fase di valutazione, ovvero hypervolume e generational distance. Inoltre al suo interno è presente l'algoritmo per il calcolo della violazione relativa.

5.3.7 Implementazione dell'algoritmo B-MOMS

Questo modulo contiene l'implementazione dell'algoritmo B-MOMS proposta in [3], più alcune modifiche necessarie sia a una corretta interpretazione del modello in formato XML sia per tener traccia delle varie grandezze misurate.

5.3.8 Eseguibili

Sono presenti molti eseguibili che consentono di eseguire simulazioni parziali o complete, per un solo scenario o per un loro insieme. Per esempio possiamo lanciare l'esecuzione dell'algoritmo B-MOMS per un insieme di modelli, calcolare le varie misure prestazionali, elencare i dati ottenuti su varie simulazioni, generare l'insieme delle soluzioni di riferimento e così via.

Capitolo 6

Realizzazioni sperimentali e valutazione

In questo capitolo mostriamo i principali test eseguiti e ne analizziamo i dati ottenuti.

6.1 Grandezze misurate

Riepiloghiamo tutti i dati collezionati durante le varie simulazioni utili per la fase di valutazione.

- Numero dei messaggi scambiati dall'algoritmo B-MOMS per ottenere la frontiera di Pareto.
- Tempo di esecuzione espresso in secondi relativo ai seguenti algoritmi:
  - B-MOMS
  - MST
  - Bruteforce
- Frontiera di Pareto ottima trovata mediante l'algoritmo di forza bruta ($\mathcal{S}^*$).

- Frontiera di Pareto trovata dall'algoritmo B-MOMS ($\mathcal{S}$).
- Frontiera di Pareto dei soli punti fattibili rispetto al modello originale ($\mathcal{S}_F$).
- Media e valore massimo della violazione relativa calcolata su $\mathcal{S}$.
- Metrica hypervolume calcolata su $\mathcal{S}_F$.
- Metrica generational distance calcolata su $\mathcal{S}_F$.

6.2 Test svolti

Abbiamo eseguito i test distinguendo due casi in base alla presenza o meno di cicli nel grafo fattorizzato, infatti, nel caso di un grafo già inizialmente aciclico la soluzione trovata è ottima quindi molte delle metriche presentate sono del tutto inutili da considerare.

Per eseguire i test usiamo un computer dotato di un processore Intel Core i5-2410M a 2.30GHz e di 6GB di RAM avente Windows 7 a 64 bit come sistema operativo.

6.2.1 Test su grafi aciclici

Per prima cosa dobbiamo far notare che gli scenari che hanno un grafo già inizialmente aciclico sono un sottoinsieme di tutti gli scenari generabili, sono in generale più semplici e meno significativi rispetto ai casi reali. È tuttavia vero che lo stesso sistema idrico presentato dal più volte citato articolo di Yang et al. è rappresentabile mediante un tale modello. In Figura 6.1 mostriamo uno schema che descrive tutti i fiumi che possiedono grafo fattorizzato aciclico, infatti ogni sistema idrico costituito da dighe nei nodi interni, segnati con D, e da nodi generici, indicati con N, sulle foglie e sulla radice, hanno il corrispondente grafo fattorizzato aciclico. Per nodo generico intendiamo un nodo relativo ad un agente città o diga.

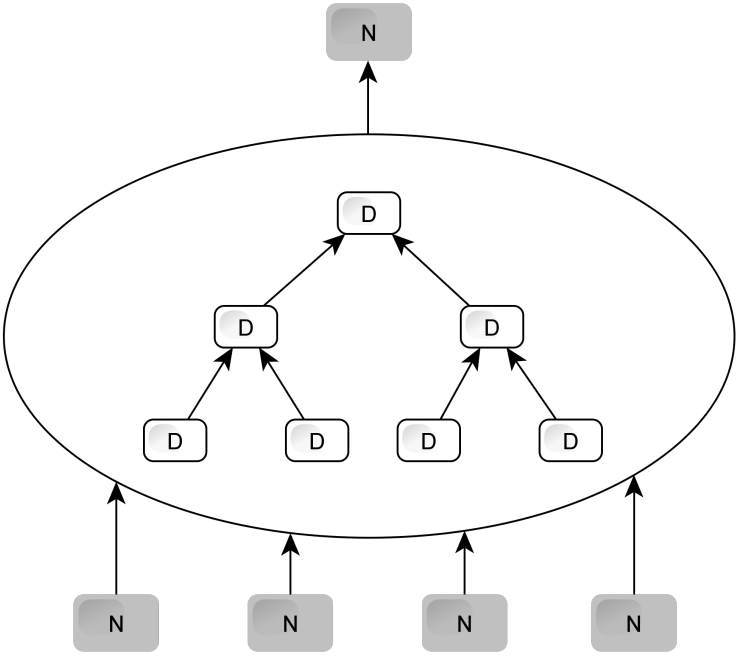


Figura 6.1: Schema dei fiumi aventi grafo aciclico

Eseguiamo delle prove su scenari via via più complessi per un totale di 5 casi, ciascuno dei quali riferito a 7 sistemi idrici nelle 3 diverse condizioni di abbondanza idrica, cioè 21 scenari per casistica. Su questi misuriamo il tempo di esecuzione sia dell'algoritmo B-MOMS che dell'algoritmo di forza bruta (o bruteforce). In Tabella 6.1 ne mostriamo i tempi di esecuzione medi al variare del numero di agenti e della dimensione del dominio, quindi al crescere della complessità del sistema.

In Figura 6.2 evidenziamo le differenze tra i due approcci, in cui il tempo di esecuzione di B-MOMS cresce molto più lentamente rispetto a quello relativo all'algoritmo di forza bruta. Tale andamento è confermato dall'analisi teorica della complessità dei due algoritmi, infatti, siano a il numero di agenti e d la dimensione del dominio allora il bruteforce ha complessità asintotica pari a $\mathcal{O}(d^a)$, mentre per B-MOMS vale $\mathcal{O}(d^{a_{MAX}})$ dove a_{MAX} indica il numero di variabili che vincolano la funzione con arità maggiore. Nel caso di modelli che generano grafi aciclici quest'ultimo valore è superiormente limitato dal numero di fiumi che compongono lo scenario più uno, quindi sia f il numero di fiumi

		Tempo di esecuzione medio [s]	
n°Agenti	*/Dominio/*	*B-MOMS*	*Bruteforce*
5	5	0.1860	0.5800
6	6	0.2508	5.8637
6	8	0.3506	33.6365
6	10	0.4475	136.7959
7	6	0.4525	294.6860

Tabella 6.1: Confronto dei tempi medi di esecuzione tra B-MOMS e bruteforce

che compongono il sistema idrico allora vale la disequazione $a_{MAX} \leq f + 1$.

6.2.2 Test su grafi ciclici

Le cose cambiano radicalmente quando si passa a delle simulazioni su grafi inizialmente ciclici in cui la soluzione trovata non è quella ottima. Dobbiamo quindi misurarne la bontà e in più fare delle valutazioni relative alla fattibilità o meno di alcuni punti della frontiera di Pareto.

Confronto del tempo di esecuzione tra MST e B-MOMS

Eseguiamo alcuni test per valutare l'incidenza del tempo impiegato dall'algoritmo MST rispetto al calcolo della frontiera di Pareto effettuato mediante B-MOMS, per esempio mostriamo in Figura 6.3 i tempi medi di esecuzione per entrambi gli algoritmi riferiti a scenari con dimensione del dominio pari a 8 al crescere del numero di agenti, in cui evidenziamo che il primo è praticamente ininfluente. Per ciascun dei sei casi previsti, la media è riferita a 9 sistemi idrici nelle tre condizioni di abbondanza di acqua per un totale di 27 scenari.

Numero di messaggi scambiati

Analizziamo di seguito l'andamento del numero di messaggi scambiati al crescere della complessità del sistema, sia per quanto riguarda il numero di agenti che per il dominio. Nel primo test confrontiamo quindi il numero di messaggi

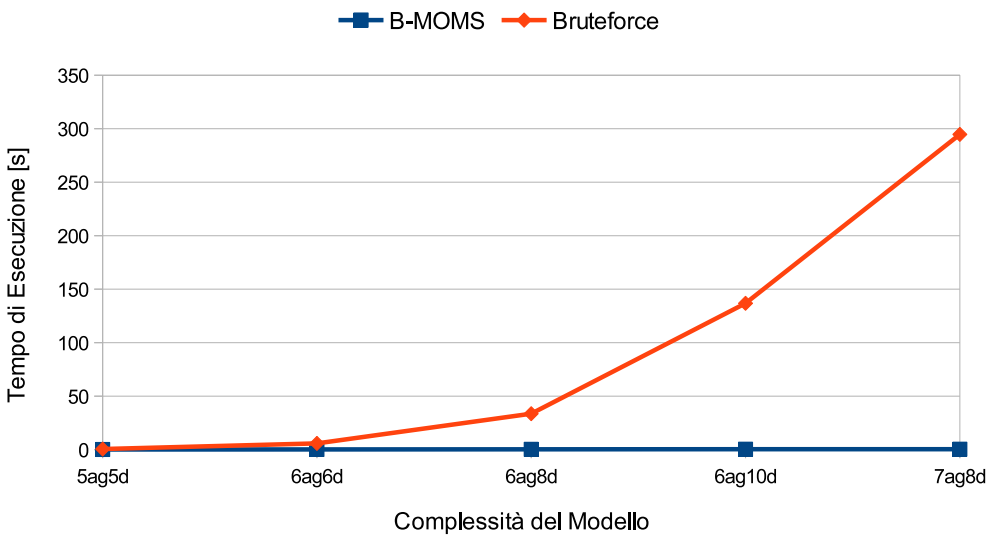


Figura 6.2: Confronto dei tempi di esecuzione su grafi aciclici

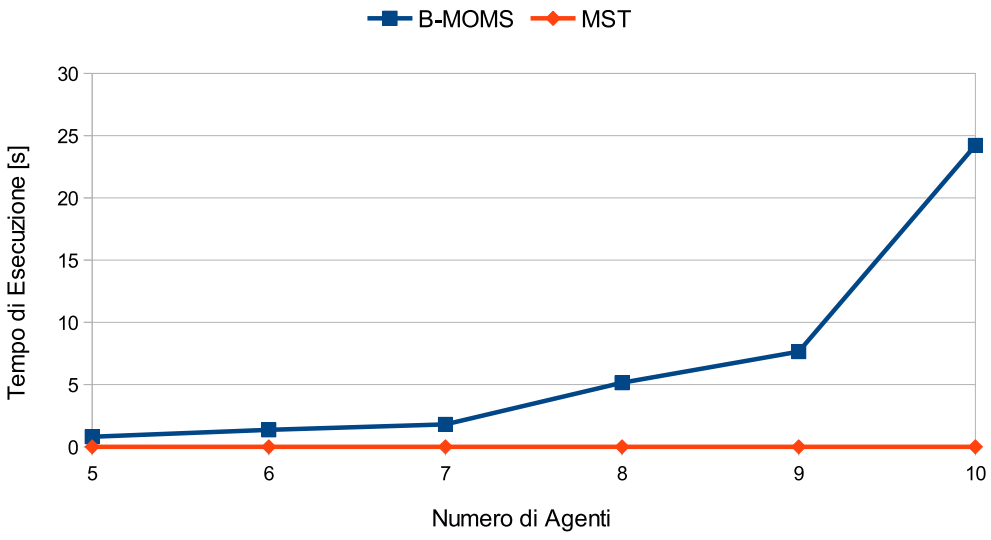


Figura 6.3: Confronto tra il tempo di esecuzione di B-MOMS e MST

	Numero medio di messaggi				
N°agenti	*U*	*RU*	*RN*	*CL*	*CM*
5	192	188	189	187	182
6	273	253	194	252	191
7	343	333	292	326	280
8	393	394	402	394	403
9	509	542	408	474	397
10	639	725	335	749	326

Tabella 6.2: Numero medio di messaggi scambiati da B-MOMS al crescere del numero degli agenti

scambiati dall'algoritmo B-MOMS per ottenere la frontiera di Pareto al crescere del numero di agenti, tenendo fissa a 8 la dimensione del dominio di ciascuna variabile. Per ciascun dei sei casi previsti, generiamo 9 sistemi idrici nelle tre condizioni di abbondanza di acqua per un totale di 27 scenari. Le medie dei dati raccolti sono disponibili nella Tabella 6.2 e nel relativo grafico di Figura 6.4, in cui usiamo le abbreviazioni mostrate nella Sezione 4.2.2 per riferirci alle tecniche di peso. Grazie a questi dati concludiamo che le tecniche RN e CM sono in genere migliori rispetto alle altre, infatti queste ultime due tendono a produrre un numero di messaggi costante all'aumentare della complessità, mentre le altre tendono a crescere.

Facciamo la stessa cosa al crescere della dimensione del dominio tenendo fisso il numero di agenti pari a 5 e utilizzando lo stesso numero di scenari per ognuno dei sei casi, mostriamo i dati ottenuti nella Tabella 6.3 e nel relativo grafico di Figura 6.5. In questo caso, invece, non troviamo nessuna euristica migliore delle altre e l'andamento è grossomodo costante.

Tempo medio di esecuzione

Usando gli stessi scenari dei test della Sezione 6.2.2 analizziamo il tempo di esecuzione al crescere della dimensione del dominio valutandolo su scenari di 5 agenti, come mostrato nella Tabella 6.4 e nel relativo grafico di Figura 6.6.

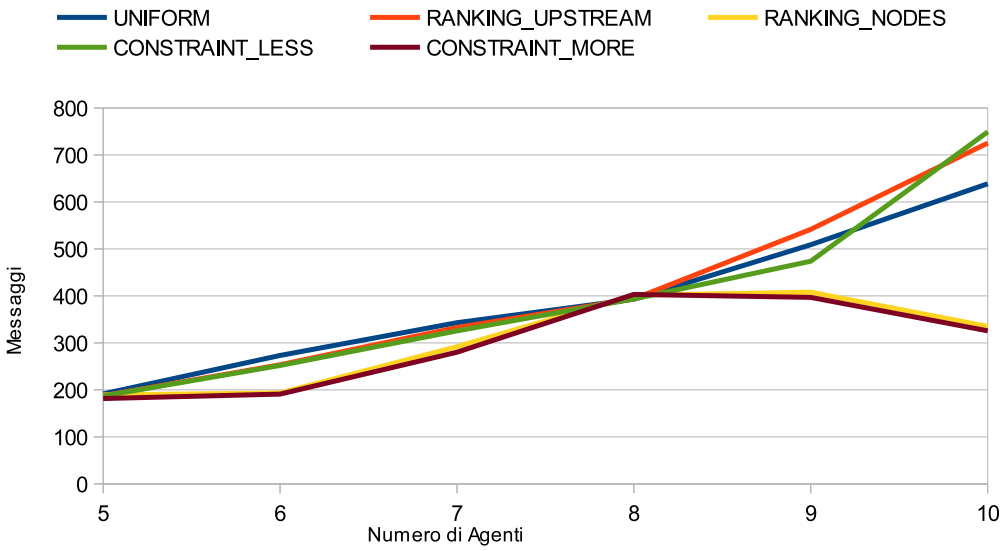


Figura 6.4: Grafico del numero di messaggi scambiati al variare del numero di agenti

	Numero medio di messaggi				
/Dominio/	*U*	*RU*	*RN*	*CL*	*CM*
5	182	181	173	186	175
10	198	198	187	200	185
15	199	195	183	195	183
20	199	199	191	196	185
25	199	188	181	191	183
30	194	196	191	195	193

Tabella 6.3: Numero medio di messaggi scambiati da B-MOMS al crescere della dimensione del dominio

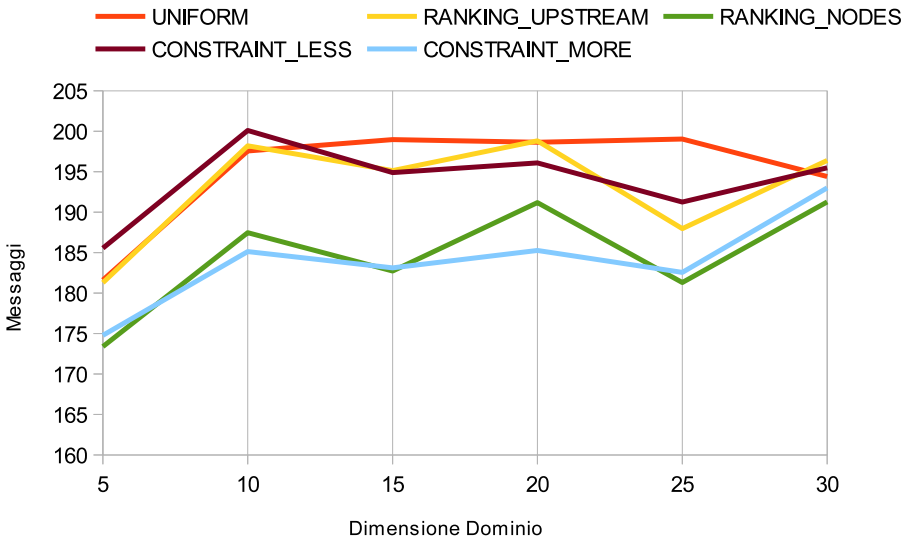


Figura 6.5: Grafico del numero di messaggi scambiati al variare della dimensione del dominio

	Tempo di esecuzione medio[s]				
/Dominio/	*U*	*RU*	*RN*	*CL*	*CM*
5	0.126	1.303	2.987	0.163	0.538
10	0.364	1.213	1.071	0.315	0.769
15	0.676	1.478	2.016	0.525	0.973
20	2.569	5.774	7.607	9.274	11.126
25	5.506	9.382	12.290	15.915	16.947
30	14.572	28.265	31.711	45.465	41.456

Tabella 6.4: Confronto dei tempi di esecuzione medi al crescere della dimensione del dominio

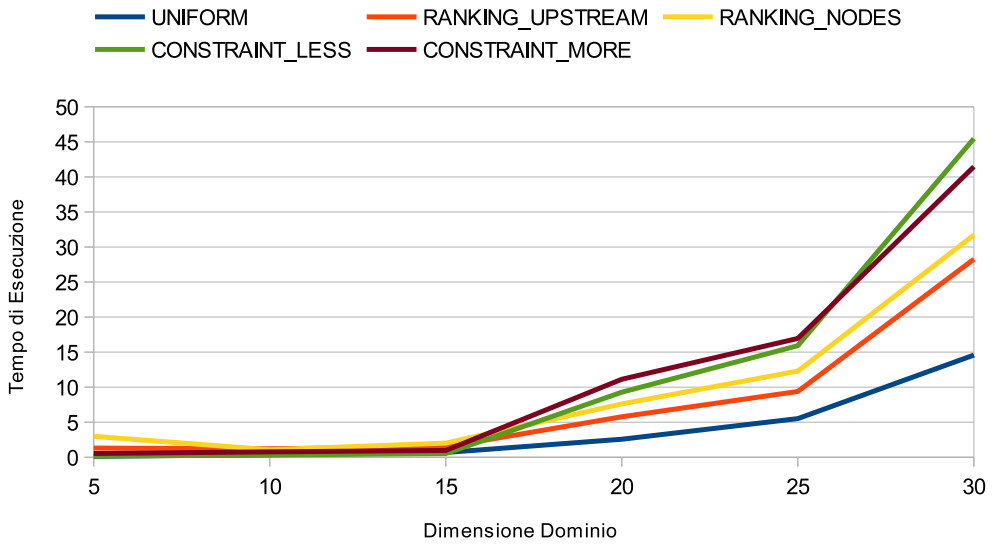


Figura 6.6: Grafico del tempo di esecuzione medio con numero di agenti costante

Da questo test risulta che le tecniche di peso hanno tempi di esecuzioni simili eccetto U che è in genere più rapida.

Inoltre, per scenari con dimensione del dominio pari a 8, eseguiamo i test al variare del numero di agenti, il tutto è proposto nella Tabella 6.5 e nel relativo grafico di Figura 6.7. Al crescere del numero di agenti non siamo in grado di stabilire quale sia l'euristica migliore rispetto al tempo di esecuzione medio in quanto sono circa equivalenti.

Possiamo però far notare l'andamento crescente del tempo di esecuzione di B-MOMS all'aumentare sia del numero di agenti che della dimensione del dominio.

Bontà delle soluzioni

Nel caso in cui esistano almeno tre soluzioni fattibili ($|\mathcal{S}_F| \geq 3$) possiamo inoltre calcolare le metriche esposte nelle Sezioni 4.3.4 e 4.3.3.

Proponiamo in Figura 6.8 l'inverso della metrica generational distance cal-

	Tempo di esecuzione medio[s]				
N°agenti	*U*	*RU*	*RN*	*CL*	*CM*
5	0.334	0.509	0.928	1.092	1.196
6	0.417	1.729	1.840	2.420	3.141
7	1.200	1.681	1.975	0.370	1.695
8	0.879	4.955	6.767	6.571	7.086
9	2.142	8.825	6.687	7.668	7.395
10	21.460	27.793	14.214	31.255	26.364

Tabella 6.5: Confronto dei tempi di esecuzione medi al crescere del numero di agenti

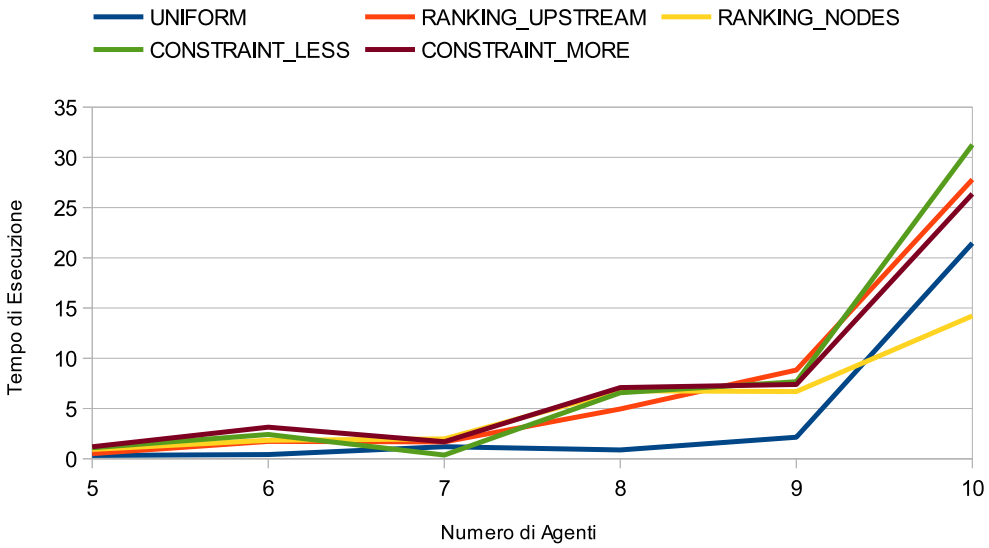


Figura 6.7: Grafico del tempo di esecuzione medio con dominio costante

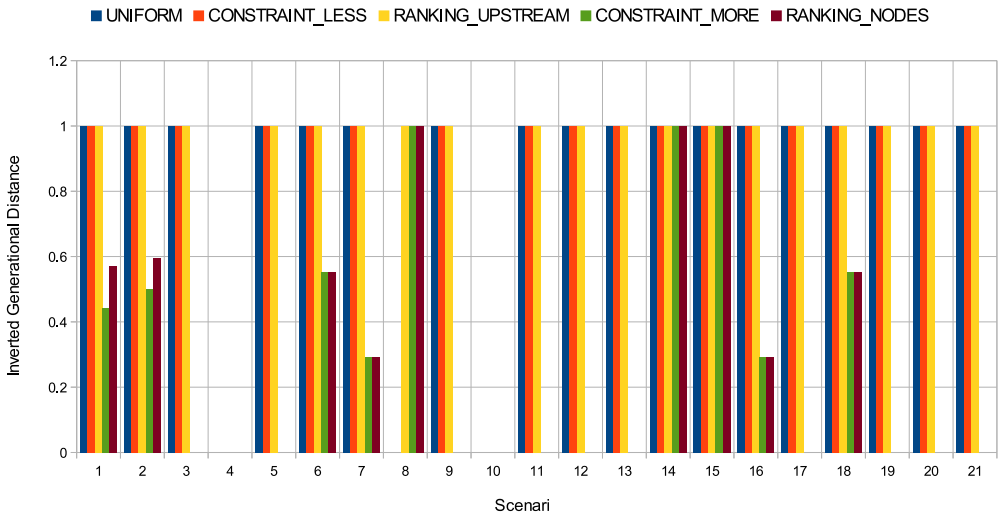


Figura 6.8: Grafico della metrica generational distance inversa

colata su 7 sistemi idrici nelle tre diverse condizioni di abbondanza idrica, quindi in totale su 21 scenari tutti composti da 8 agenti e aventi variabili con dominio di 7 elementi, in cui i valori alti indicano una buona soluzione. Applichiamo l'inversione delle metrica poiché alcune frontiere non hanno sufficienti punti e quindi si creerebbe confusione con le soluzioni che hanno metrica nulla perché molto vicine al riferimento e quindi di buona qualità.

Un esempio di metrica hypervolume calcolata sullo stesso insieme di frontiere è esposto in Figura 6.9, ricordiamo inoltre che dove la metrica assume valori più alti allora maggiore sarà la qualità della soluzione.

Rispetto alla bontà della soluzione, le tecniche U, CL e RU producono risultati equivalenti e in genere migliori, queste ultime sono quindi da preferire alle altre se si cerca una buona qualità delle soluzioni.

Nell'ultimo grafico proposto (Figura 6.9) a volte la metrica hypervolume è nulla, tale questione merita un'ulteriore analisi in quanto deriva dal fatto che le soluzioni trovate sono degeneri, cioè, considerando un problema di massimizzazione, sono disposte lungo gli assi. Ciò significa che per ogni soluzione esiste almeno un valore inferiore all'estremo minore del riferimento scelto.

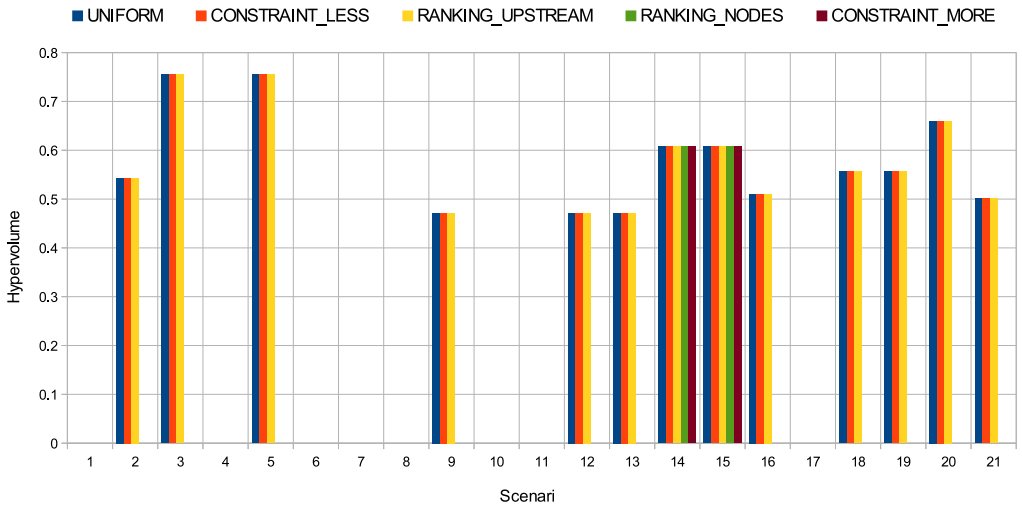


Figura 6.9: Grafico della metrica hypervolume

Grado di violazione

Nei casi in cui, a causa del numero insufficiente di soluzioni fattibili, non si possono calcolare le metriche di bontà della soluzione, allora optiamo per il grado di violazione che ci consente di stabilire di quanto i vincoli fisici sono stati violati. Mostriamo per esempio i dati relativi a delle simulazioni su 16 scenari tutti composti da 8 agenti e dimensione del dominio pari a 7, sia per quanto riguarda il valore medio di violazione (Figura 6.10) sia per il valore massimo (Figura 6.11). Il grado di violazione per ogni scenario è piuttosto contenuto, infatti la violazione media non supera il 15% e quella massima il 25%. Confrontando le varie euristiche emerge che RN è sempre migliore delle altre eccetto nell'ultimo caso, la seconda tecnica di peso da preferire per ridurre il grado di violazione è CM, infine troviamo le altre tre che si equivalgono tra loro.

Soluzioni dominate

Nel caso in cui dal taglio degli archi non conseguono soluzioni che dominano alcuni punti la frontiera di riferimento allora i valori delle metriche calcolati

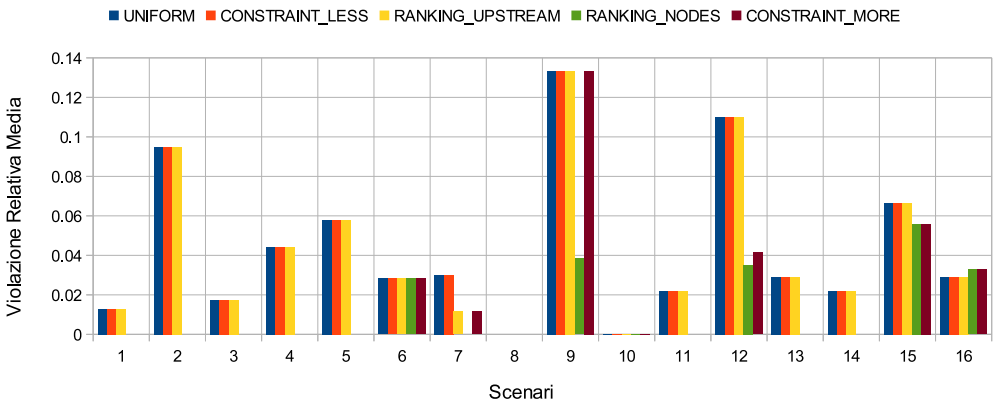


Figura 6.10: Grafico della violazione relativa media

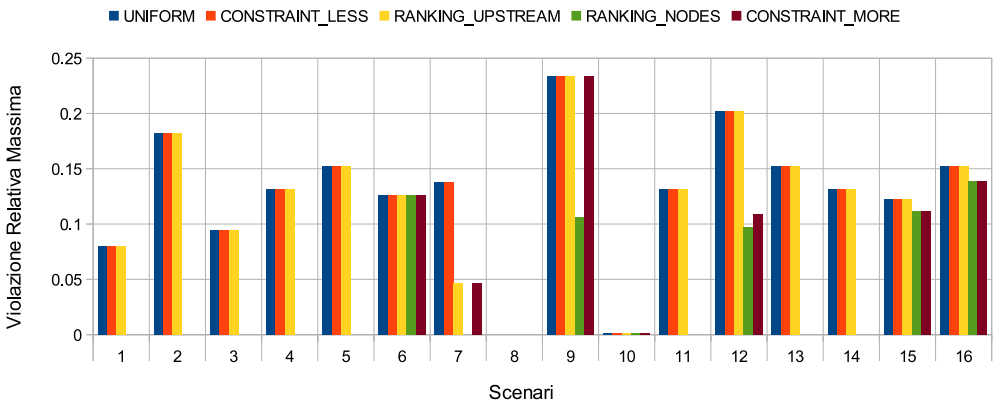


Figura 6.11: Grafico della violazione relativa massima

assumono un significato maggiore, andiamo quindi ad analizzare nei dettagli questi casi. Ci riferiamo a delle simulazioni composte da 8 agenti aventi dimensione del dominio pari a 7, su 40 scenari considerati solo in 12 di questi esiste almeno un'euristica che produce soluzioni interamente dominate, data la presenza di 5 tecniche di peso ciò accade 26 volte su 200. In ognuno dei 26 casi essi hanno tutte le soluzioni fattibili rispetto al modello originale, cioè $\mathcal{S} = \mathcal{S}_F$. Inoltre solo in 5 casi su 26, circa il 20%, la metrica hypervolume assume valore diverso da zero.

Capitolo 7

Risultati e conclusioni

In questo lavoro abbiamo valutato una particolare implementazione dell'algoritmo B-MOMS mediante la simulazione su scenari realistici generati in modo automatico. Nel corso del lavoro abbiamo distinto due classi di scenari. La prima di queste è relativa a modelli con grafo dei vincoli aciclico, in questo caso abbiamo verificato che l'algoritmo risolve in modo ottimale il problema come previsto da precedenti risultati teorici. Riguardo a questo caso possiamo notare un'ottima scalabilità dei tempi di esecuzione rispetto a un approccio centralizzato al crescere della complessità del sistema.

La seconda classe di scenari è invece caratterizzata da modelli con grafo dei vincoli ciclico, perciò richiede una modifica del modello stesso con conseguente perdita di ottimalità. Abbiamo quindi cercato delle euristiche che ci permettessero di ottenere le soluzioni migliori possibili. Per valutarle abbiamo simulato l'algoritmo su vari scenari in diverse condizioni misurandone varie grandezze. Da queste valutazioni concludiamo che non esiste un'euristica migliore delle altre in ogni caso, ma ciò dipende in base alla grandezza misurata. Infatti le tecniche RN e CM risultano migliori osservando il grado di violazione, mentre sono da preferire le tecniche RU, U e CL nel caso si cerchi una buona qualità delle soluzioni.

I risultati ottenuti non potevano essere previsti dal lavoro che ha dato origine a questa tesi, in quanto esso si basava su un sistema idrico piuttosto

semplice con un modello privo di cicli, non era quindi evidente che rendere aciclico un modello più complesso potesse portare conseguenze così critiche.

Ipotizziamo dunque due possibili evoluzioni, la prima consiste nel cambiamento dell'algoritmo utilizzato passando a uno completo. La seconda ipotesi è di continuare la ricerca in questa direzione accettando l'assenza di garanzia per la qualità delle soluzioni.

Bibliografia

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

[2] D. Anghileri, A. Castelletti, F. Pianosi, R. Soncini-Sessa, and E. Weber. Optimizing watershed management by coordinated operation of storing facilities. *Journal of Water Resources Planning and Management*, 139(5):492–500, 2013.

[3] E. Bontempi. Studio di sistemi idrici come problemi di ottimizzazione distribuiti multiobiettivo: modelli e algoritmi. Tesi di Laurea Magistrale, Politecnico di Milano, 2011/2012.

[4] F. M. Delle Fave, R. Stranders, A. Rogers, and N. R. Jennings. Bounded decentralised coordination over multiple objectives. In *The 10th International Conference on Autonomous Agents and Multiagent Systems*, January 2011.

[5] J. J. Durillo, A. J. Nebro, and E. Alba. The jMetal framework for multi-objective optimization: design and architecture. In *Congress on Evolutionary Computation 2010*, pages 4138–4325, Barcelona (Spain), July 2010.

[6] 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, January 1983.

[7] M. Giuliani, A. Castelletti, F. Amigoni, and X. Cai. Multi-agent systems optimization for distributed watershed management. In *Proceedings of the International Environmental Modelling and Software Society (iEMSs)*, pages 309–316, Leipzig (Germany), July 2012.

[8] M. Giuliani, A. Castelletti, F. Amigoni, and X. Cai. Distributed optimization as a tool to balance efficiency and practicabiity in watershed management. Under review, 2013.

[9] K. Madani. Game theory and water resources. *Journal of Hydrology*, 381(3–4):225–238, 2010.

[10] K. Madani and J. Lund. California's Sacramento–San Joaquin delta conflict: From cooperation to chicken. *Journal of Water Resources Planning and Management*, 138(2):90–99, 2012.

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

[12] 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, February 2011.

[13] S. J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Pearson Education, 2003.

[14] J. Waterbury. Legal and institutional arrangements for managing water resources in the Nile Basin. *International Journal of Water Resources Development*, 3(2):92–104, 1987.

[15] K. Watkins. *Human Development Report 2006: Beyond Scarcity. Power, Poverty and the Global Water Crisis*. United Nations Development Programme, 2006.

[16] D. Whittington, X. Wu, and C. Sadoff. Water resources management in the Nile Basin: The economic value of cooperation. *Water Policy*, 7(3), 2005.

[17] Y. E. Yang, X. Cai, and D. M. Stipanović. A decentralized optimization algorithm for multiagent system–based watershed management. *Water Resources Research*, 45(8), 2009.

[18] 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*, 10:673–685, 1998.

[19] E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271, 1999.

[20] JDOM - Librerie Java open source per la manipolazione di file XML. `http://www.jdom.org/`.

[21] JGrapT - Librerie Java open source per la gestione di grafi. `http://jgrapht.org/`.

[22] MVEL - Librerie Java open source per la manipolazione di espressioni matematiche. `http://mvel.codehaus.org/`.

Appendice A

Specifiche XML

A.1 Modello in formato XML

Di seguito illustriamo alcune parti di un file XML che ci permette di rappresentare un modello. Quest'ultimo è composto da sette agenti ciascuno dei quali controlla una variabile identificata dal nome univoco e dall'indice della posizione occupata nel vettore degli obbiettivi. Per ogni variabile è presente l'elenco dei valori del suo dominio. Per esempio, tra le altre, contiene le variabili x_1 e x_7, la prima di queste occupa la prima posizione e il suo dominio è $D_1 = \{1.0, 11.7, 17.5, 23.3, 29.1, 35.0, 40.8, 45.2\}$. Il modello comprende inoltre la funzione obbiettivo f_{o1} che dipende solo da x_1 come si può evincere dalla sua espressione equivalente a $-0.15x_1^2 + 6.5x_1 - 8$. La funzione f_{11} è invece relativa a un vincolo del sistema, dipende da $\mathbf{x}_{11} = \{x_4, x_5, x_6\}$ ed è mostrata nell'Equazione A.1, in cui $u(\mathbf{x}_{11}) = x_4 - x_5 - x_6 - 9.0$.

$$f_{11}(\mathbf{x}_{11}) = \begin{cases} 0 & x_4 - x_5 - x_6 \geq 9 \\ u(\mathbf{x}_{11}) & 9 \geq x_4 - x_5 - x_6 \geq 0 \\ -\infty & \textit{altrimenti} \end{cases} \tag{A.1}$$

```
<?xml version="1.0" encoding="UTF-8"?>
<graph objectives="8">
  <variables>
    <variable name="x1" pos="1">
```

```
      <domain>1.0,11.7,17.5,23.3,29.1,35.0,40.8,45.2</domain>
    </variable>
    <!-- ... -->
    <variable name="x7" pos="7">
      <domain>0.0,11.9,17.9,23.8,29.8,35.7,41.7,44.1</domain>
    </variable>
  </variables>
  <functions>
    <function name="fo1" pos="1" arity="1">
      <scope>
        <var>x1</var>
      </scope>
      <utilities>
        <expr>-0.15*x1*x1+6.5*x1-8.0</expr>
      </utilities>
      <booleans>
        <expr>true</expr>
      </booleans>
    </function>
    <!-- ... -->
    <function name="f11" pos="8" arity="3">
      <scope>
        <var>x4</var>
        <var>x5</var>
        <var>x6</var>
      </scope>
      <utilities>
        <expr>0</expr>
        <expr>x4-x5-x6-9.0</expr>
        <expr>-2147483648</expr>
      </utilities>
      <booleans>
        <expr>x4-x5-x6&amp;gt;=9.0</expr>
        <expr>x4-x5-x6&amp;gt;=0</expr>
        <expr>x4-x5-x6&amp;lt;0</expr>
      </booleans>
    </function>
    <!-- ... -->
```

```
  </functions>
</graph>
```

A.2 Risultati in formato XML

Il documento XML dei risultati è suddiviso in vari elementi, ciascuno dei quali corrispondente ad una tecnica di peso. Per ognuna di queste ultime salviamo varie grandezze come la frontiera di Pareto espressa in termini di costo, il tempo di esecuzione di MST e di B-MOMS, e il numero di messaggi scambiati. A queste ultime vanno aggiunte le grandezze calcolate nella fase di valutazione cioè le metriche hypervolume e generational distance, il sottoinsieme delle soluzioni fattibili e il grado di violazione sia per ciascuna soluzione sia in forma aggregata attraverso la media e il valore massimo. Infine a sè stante vi è l'elemento relativo al tempo di esecuzione dell'algoritmo di forza bruta.

```
<?xml version="1.0" encoding="UTF-8"?>
<OptimizationLog>
  <pareto_UNIFORM>
    <MST_Execution_time_in_seconds>
      0.003
    </MST_Execution_time_in_seconds>
    <B-MOMS_Execution_time_in_seconds>
      1.445
    </B-MOMS_Execution_time_in_seconds>
    <cost elements="18">
      <!-- frontiera di Pareto -->
    </cost>
    <feasible_cost elements="5">
      <!--
        frontiera di Pareto composta dalle soluzioni fattibili
      -->
    </feasible_cost>
    <violations>
      <!-- violazioni relative per ciasuna soluzione -->
    </violations>
    <mean_violation>0.049</mean_violation>
```

```
    <max_violation>0.253</max_violation>
    <messages_numbers>350</messages_numbers>
    <hypervolume>0.981</hypervolume>
    <feasible_hypervolume>0.754</feasible_hypervolume>
    <generational_distance>0.0</generational_distance>
    <feasible_generational_distance>
      0.0
    </feasible_generational_distance>
  </pareto_UNIFORM>
  <pareto_RANKING_UPSTREAM>
    <!-- ... -->
  </pareto_RANKING_UPSTREAM>
  <pareto_RANKING_NODES>
    <!-- ... -->
  </pareto_RANKING_NODES>
  <!-- tecniche CL e CM -->
  <Bf_exec_time_in_seconds>442.032245988</Bf_exec_time_in_seconds>
</OptimizationLog>
```