POLITECNICO DI MILANO

Corso di Laurea Specialistica in Ingegneria per l'Ambiente e il Territorio

Facoltà: Ingegneria Civile, Ambientale e Territoriale

OTTIMIZZAZIONE DELLE RETI DI SERBATOI:
un approccio neurale

Relatore: Prof. Giorgio Guariso

Correlatore: Ing. Daniele De Rigo

Tesi di Laurea di:

Daniele Cremonesi

Matricola 724781

Anno accademico 2009 – 2010

INDICE GENERALE

INDICE DELLE FIGURE

INDICE DELLE TABELLE

INDICE DEI GRAFICI

ABSTRACT

La gestione dei sistemi naturali complessi, come ad esempio quelli idrici, richiede la contemporanea conoscenza di molti fenomeni e dei valori di molte grandezze: è pertanto un'attività complessa, volta a soddisfare le esigenze di tutti i soggetti, chiamati Portatori di Interesse, che traggono un qualche beneficio dal funzionamento del sistema stesso.
Per definire la gestione ottima di tali sistemi sono state proposte tecniche ormai classiche, come la Programmazione Dinamica, che ricercano il valore ottimo della funzione obiettivo in modo ricorsivo; queste tecniche vanno incontro, però, a diversi problemi, il più importante dei quali è costituito dall'aumento esponenziale dei tempi di calcolo con l'aumentare, anche di poco, dello stato del sistema. Da qui l'esigenza di fare ricorso a nuovi modelli, quali le reti neurali, che, apprendono da dati di esempio come comportarsi, emulando il sistema nervoso umano.
Se tali modelli sono ben addestrati, riescono a rispondere bene ad un'ampia casistica di ingressi, fornendo in uscita valori soddisfacenti. Lo scopo di questo lavoro è esplorare in che modo è possibile utilizzare le reti neurali per gestire un sistema idrico complesso (quello del fiume Zambesi), dopo aver costruito numerose serie sintetiche di addestramento. Tali serie, nel caso specifico, sono costruite utilizzando un algoritmo di gestione ottima in anello aperto, che cioè assume la perfetta conoscenza di tutti gli afflussi futuri. Le reti neurali sono tarate in modo da replicare la gestione ottima così calcolata, ma in anello chiuso, cioè in base allo stato corrente del sistema e quindi possono essere poi impiegate in tempo reale per l'effettiva gestione.

RINGRAZIAMENTI

Quando uno studente arriva a scrivere le ultime righe di una Tesi di Laurea, è perché ha raggiunto un importantissimo traguardo, che definire solamente professionale sarebbe troppo riduttivo. La redazione e la successiva discussione della Tesi rappresentano un momento fondamentale della vita di uno studente universitario, che giunge al termine di un intenso percorso, di arricchimento anche umano.
Una Tesi di Laurea non può svilupparsi, comunque, senza l'interesse e l'attenzione di un Relatore, che segue, suggerisce e guida lo studente nella redazione del testo. Il primo, più sentito ringraziamento, devo rivolgerlo pertanto al Professor Giorgio Guariso (DEI – *Dipartimento di Elettronica ed Informazione*), il quale ha costantemente monitorato tutto il lavoro e mi ha fornito, più volte, spunti ed idee per correggere gli errori concettuali, la forma ed il testo scritto nel suo complesso.
Un lavoro di questo tipo richiede una consistente parte di calcolo, che non avrebbe potuto essere sviluppata in modo compiuto senza i validi suggerimenti ed aiuti dell'Ingegner Daniele De Rigo (DEI – LITA, *Laboratorio di Informatica Territoriale e Ambientale*), che ha saputo guidarmi e suggerirmi, con consigli utilissimi, laddove le righe di codice sembravano ostacoli insormontabili.
Il lavoro qui svolto rappresenta solamente l'ultimo passo di un percorso bellissimo, faticoso ma nel contempo entusiasmante, che mi ha permesso di ottenere risultati talvolta molto soddisfacenti, talvolta un po' meno: niente sarebbe stato possibile, però, senza il sostegno dei miei genitori e della mia famiglia, che mi ha insegnato, soprattutto, a non arrendermi mai davanti alle difficoltà della vita, ed a risolvere i problemi tutti insieme, anche quando sembravano non avere soluzione. A tutti loro, grazie di cuore.
Il cammino compiuto in questi anni al Politecnico è stato reso ancora più bello dalla presenza, costante e continua, dei miei compagni di corso, alcuni diventati amici prima ancora che compagni.
Fin dall'inizio, insieme a Marco M., Paolo, Stefano, Alvise, Enea, Pasquale e Michele, le faticose giornate all'insegna di formule da imparare, esercizi – anche impossibili – da risolvere, relazioni da svolgere, battute scherzose, momenti allegri, … partite a carte …, e tanto, tanto, ridere, sono passate più leggere. A loro, negli anni, se ne sono aggiunti tanti altri (spero di non dimenticare qualcuno, e, se l'ho fatto, non me ne si voglia): Loris, Arianna, Anna, Andrea V., Gabriele, Alberto, Riccardo, Gloria, Andrea G., Francesco, Filippo, Chiara G., Stefania B., Elisa F., Letizia, Sara, ed altri ancora; un grazie particolare lo devo a Giovanni, con cui è sempre stato interessante discutere dei problemi più disparati, e tra questi, sono venuti a galla interessanti spunti ed idee per la tesi.
I momenti passati al LITA sono stati proficui e piacevoli, seppur brevi: grazie a tutte le persone, ragazze e ragazzi che ho incontrato.
Al di fuori dell'Università, non mi hanno mai lasciato gli amici di sempre, con cui ho condiviso tante avventure, in cui ho saputo trovare sempre un sostegno, una bella parola, tanto aiuto nei momenti più difficili, tanti consigli per affrontare i problemi: a Guido, Laura, Marco L., Elisa M., Federico, un abbraccio fortissimo ed un grazie che non può esaurirsi in queste righe.
Esistono sicuramente tante altre persone che hanno contribuito ad accompagnarmi, in un modo o nell'altro, fino a questo punto. A tutti loro, un altro grazie.

PREMESSA

"[…] Quello che stiamo analizzando non è altro che un problema di scelta strategica: vengono presentate un certo numero di opzioni, sono esaminabili una alla volta e bisogna fare la propria scelta nel momento stesso dell'esame, senza possibilità di tornare indietro. Ogni fallimento dei predecessori è informazione preziosa per determinare la strategia dei successori. […]" *[Le Scienze, 2009]*

Ogni attività scientifica ed ogni progetto ingegneristico deve essere portata a termine con un'attenta analisi, fin dalla ricerca dei dati, o dalla loro ricostruzione artificiale nella misura più veritiera possibile, qualora quelli a disposizione siano esigui, quando non del tutto assenti.
La risoluzione di un problema, anche molto semplice, come quello cui la citazione precedente si riferisce, ossia la ricerca della migliore libreria di Charing Cross Road a Londra, ma soprattutto molto complesso, in cui entrino in gioco fattori fondamentali per lo sviluppo ed il benessere delle persone, come l'utilizzo dell'acqua, spinge a ricercare metodi sempre più raffinati e complessi, talvolta ancora agli albori del loro sviluppo, e, proprio per questo, da esplorare.
La gestione ottimale della risorsa idrica è fondamentale per garantire a tutti coloro che ne hanno bisogno, di disporne in misura adeguata – anche averne troppa può essere un problema – in modo tale da poter soddisfare le proprie necessità, senza impedire agli altri di poter fare altrettanto.
Il tema non è certamente nuovo, perché da alcuni decenni moltissimi studiosi se ne occupano, e sono state elaborate strategie anche molto proficue per risolvere nel modo migliore possibile il problema, sebbene non sia semplice.
Ciò che è, all'interno di queste procedure risolutive, maggiormente soggetto a cambiamenti, si voglia per le continue e rapide innovazioni informatiche, si voglia per la quantità di dati, sempre crescente, di cui si dispone, sono i modelli su cui l'architettura risolutiva si appoggia.
Sovente, i modelli classici non funzionano adeguatamente quando le dimensioni fisiche del problema crescono troppo. Tale aspetto non è affatto secondario, e nemmeno improbabile: se un centro abitato si rifornisce di acqua da un torrente, esso è verosimilmente legato ad un piccolo lago, sulle cui sponde si susseguono diversi altri centri abitati, si pesca, la sua acqua si utilizza per navigare e per irrigare i campi. E a valle del centro abitato sul torrente potrebbe esserci un grande fiume, che riceve acqua da molti altri torrenti simili a quello descritto, e che fornisce acqua ad altri agricoltori, ad una centrale idroelettrica che fornisce elettricità a tutti i centri abitatati, … Cambiare qualcosa nel prelievo idrico del villaggio sul torrente potrebbe causare esondazioni o siccità lungo il fiume o alterare i livelli del lago, con evidenti, disastrose, conseguenze.
Motivi, questi, per cui non è sufficiente limitare il modello per la gestione del prelievo idrico al solo torrente, ma occorre espanderne i contorni al lago, al fiume, alla centrale.
È molto semplice, quindi, aumentare considerevolmente il numero di variabili di cui occorre tenere conto, e questo può rendere inefficaci i modelli classici.
Si rende necessario, ed anche questo è un filone di ricerca molto attivo, trovare nuove tipologie di modelli, che sappiano, ad esempio, apprendere dai dati, e capiscano quali comportamenti adottare a seconda della situazione: le Reti Neurali, emulatori del sistema nervoso umano, riescono a fare questo.
In questo lavoro di Tesi si cerca di utilizzare una rete neurale per riprodurre il comportamento ottimale di un sistema idrico, ormai ampiamente studiato e validato, quello del Fiume Zambesi in Africa, avendo come parametri di confronto i risultati di un

ottimizzatore lineare, Relax, anch'esso sufficientemente validato da poter essere considerato affidabile.
Antecedentemente a questo, occorre recuperare quanta più informazione possibile in ingresso al sistema (tipicamente rappresentata dall'insieme degli afflussi), ma è raro che si disponga di dati completi e vasti forniti dalla storia realmente occorsa: è necessario "costruire"questi dati, in modo sintetico, con altri modelli.
Tutto ciò rende senza dubbio approssimato il risultato: ma proprio l'approssimazione e l'incertezza fanno parte della vita di uno studioso. Sta a lui trovare un giusto compromesso tra affidabilità ed incertezza, valutando sino a che punto può inventare artificialmente dati che rispecchino la casistica della natura, nota solo a posteriori, e fino a quando può "spingere" il modello.
La ricerca è appassionante anche per questo.

"Considerate la vostra semenza:
fatti non foste a viver come bruti,
ma per seguir virtute e canoscenza"
(Dante Alighieri, Divina Commedia, Inferno, Canto XXVI, 118 – 120)

Questo lavoro di Tesi si articola come segue:
dopo una breve Introduzione, nel Capitolo 1 si descrivono i modelli classici di gestione, basati sulla Programmazione Dinamica, che entrano in crisi quando la complessità del problema cresce. Per questo, nel Capitolo 2 si descrivono le Reti Neurali, e si presentano alcune delle numerose attuali applicazioni. Nel Capitolo 3 si introduce, insieme al problema in generale, l'importante aspetto dell'addestramento delle Reti e la procedura che si intende utilizzare per risolverlo. Il Capitolo 4, puramente descrittivo, introduce sotto vari aspetti l'area interessata da questo studio, ossia il bacino del Fiume Zambesi, presentandone sia gli aspetti geografici e socio – economici, sia quelli di maggiore interesse, tecnici ed ingegneristici. Il Capitolo 5 è dedicato agli afflussi, scelta resasi necessaria per descrivere, in modo particolareggiato, come si sono costruite le serie sintetiche. Il Capitolo 6, fulcro del lavoro di Tesi, descrive tutte le sperimentazioni effettuate con le Reti Neurali e le conclusioni cui si è pervenuti. Concludono, completando, il lavoro, tre Appendici, in cui si descrivono il funzionamento del software di ottimizzazione lineare Aquafun (A1), gli algoritmi implementati in esso, in particolare Relax (A2), e le funzioni Matlab elaborate per sviluppare la Tesi (A3).

Milano, maggio 2010

INTRODUZIONE AL LAVORO

L'*Agenzia Europea dell'Ambiente* (*AEA*, o *EEA – Environmental European Agency*) ha elaborato, tra il 1994 e il 1996, uno schema (si faccia riferimento alla rappresentazione nella seguente Figura 1) che ben descrive l'evoluzione di un sistema naturale soggetto a pressione antropica: lo schema *DPSIR*, secondo il quale sul sistema agiscono dei *Determinanti* che generano delle *Pressioni* che ne alterano lo *Stato*. Tale alterazione, o variazione, genera un *Impatto* non solo sul sistema stesso, ma anche sulla società che in esso vive, la quale organizza e attua delle *Risposte* con lo scopo di cancellare – ma molto più spesso, data l'impossibilità di annullare un effetto, si limita a mitigare, o ridurre – l'azione che ha alterato lo stato del sistema. Nella Figura seguente si mostra come le Risposte siano rivolte non solo alla riduzione degli Impatti (freccia numero 4), ma anche a intraprendere un'azione preventiva, andando ad incidere sulle cause che concorrono alla generazione degli impatti stessi. Meno di sovente, quindi, le risposte incidono direttamente su Determinanti e Pressioni, in modo che gli Impatti che questi generino siano più tenui, comunque meno stressanti per il sistema in esame.
Di seguito, nella Figura 1, la rappresentazione dell'AEA:

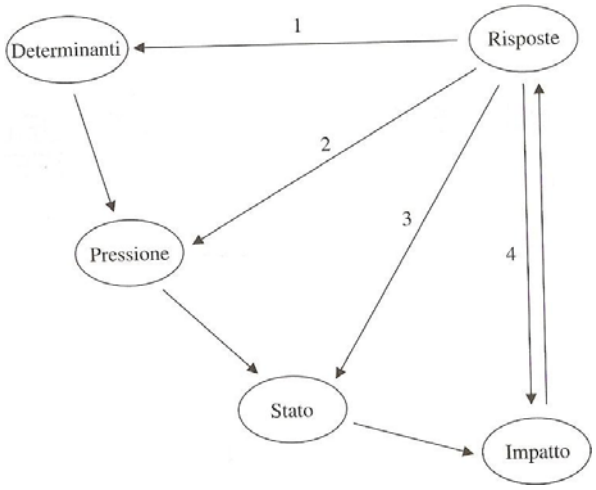

Figura 1 – Rappresentazione dello schema DPSIR elaborato da AEA, 1994 – 1996 (da Rodolfo Soncini – Sessa, 2004)

Le risposte si realizzano sotto diverse forme: *normative*, emesse per regolamentare gli interventi che possono, o meno, essere realizzati; *compensative*, quando si sceglie di ricompensare per via monetaria in proporzione al danno arrecato dall'alterazione del sistema.
Molto più di sovente, tuttavia, ciò che si sceglie di realizzare è una risposta complessa, che combini sia norme, sia compensazioni, sia *interventi strutturali* o *non strutturali*, che modificano la struttura ed il funzionamento del sistema in modo da renderlo il più adatto possibile a rispondere alla sollecitazione antropica. Si tratta, insomma, di scegliere un'*azione*, che spesso non viene attuata da sola, ma in combinazione con altre, all'interno di quella che viene chiamata *alternativa* (di essa, una buona definizione è: *"… un pacchetto integrato e coordinato di azioni …"* [Soncini Sessa, 2004]).
Sovente, le Risposte da attuare per mitigare un Impatto devono mutare nel tempo, adattandosi al nuovo contesto ambientale, per cui difficilmente possono essere *una – tantum*, prese e attuate una volta per tutte. Di solito, esse si organizzano in cicli, che si compongono di due parti, di cui la prima, denominata *pianificazione* si mette in pratica quando si sceglie l'alternativa e si mette in pratica; la seconda, chiamata *gestione*, si attua per "far funzionare" il sistema una volta che l'alternativa scelta entra a regime. In questa fase, oltre al

funzionamento, occorre monitorare anche gli effetti che si generano, al fine di attivare un nuovo ciclo quando questi effetti alterino il sistema in modo così sensibile da rendere necessaria una nuova risposta.
Per individuare ed attuare le alternative adatte allo specifico caso, occorre impostare un *Problema di Progetto*, al termine del quale si ottiene una *politica*, ossia un elenco ordinato (*sequenza temporale*) di regole che garantiscano il miglior funzionamento del sistema.
Per poter risolvere il Problema di Progetto occorre costruire un *modello* del sistema, sul quale applicare ricorsivamente un algoritmo che consenta di determinare la migliore politica.
Questo, in estrema sintesi, è lo scopo del presente lavoro di Tesi: si fa riferimento ad un sistema idrico complesso realmente esistente, quello del bacino del Fiume Zambesi in Africa, che sarà modellizzato in modo opportuno. Sul modello, avvalendosi di software ed algoritmi adatti, sarà calcolata la migliore politica di funzionamento.
Il problema brevemente introdotto nelle righe precedenti è piuttosto difficile, sia da formulare, sia da risolvere, per diversi motivi:

- Spesso si ha a che fare con sistemi naturali molto complicati, con numerosi componenti che interagiscono tra loro molto fittamente, determinando difficili legami: da qui la complessità di rappresentazione modellistica, che rischia di essere o troppo banale, o troppo dettagliata. In entrambi i casi si arriva a dei risultati molto lontani dall'ottimo, o per eccessiva approssimazione, o per eccessiva richiesta di risorse per i calcoli. Occorre saper costruire un modello abbastanza preciso, ma sufficientemente agile e leggero, in modo che vengano considerati tutti gli aspetti principali, trascurando quelli ininfluenti (assai rari, per la verità), ma badando che non si ricada in una descrizione troppo superficiale ed approssimativa del sistema stesso;
- I sistemi naturali evolvono con continuità sia nel tempo, sia dal punto di vista fisico. Ciò rappresenta un problema, perché nessuna macchina è in grado di effettuare calcoli nel continuo, per cui occorre anzitutto *discretizzare* i valori assumibili, operazione che comporta un'approssimazione nei risultati. Se la discretizzazione è troppo ampia, si rischia di tralasciare situazioni importanti o molto critiche, se troppo densa, si rischia di appesantire troppo le macchine che operano nei calcoli. Occorre prestare attenzione anche all'intervallo di valori che si ritiene il sistema modellizzato possa assumere: per questioni risolutive, occorre assegnare degli *estremi* (che di solito rappresentano vincoli fisici) che non siano né troppo restrittivi, né troppo morbidi, in modo che, per certi versi, siano rispettate tutte le situazioni che possono verificarsi in natura, per altri che non si eseguano calcoli inutili per situazioni irrealizzabili in natura;
- I sistemi naturali sono in continua evoluzione, per cui individuare un *orizzonte temporale finito* cui arrestare la simulazione sul modello è una scelta sbagliata, ma, d'altro canto, non è possibile nemmeno pensare di effettuare simulazioni "all'infinito". Secondo le necessità, è possibile ottimizzare il sistema sia su *orizzonte finito* sia su *orizzonte infinito*, necessitando, però, entrambi i casi, di accorgimenti per rendere o veritieri i risultati (nel caso finito), o possibile la simulazione (nel caso non finito);
- Spesso ci si accorge dell'eccessiva complessità del sistema troppo tardi: con il proseguire della simulazione, diventano evidenti le eccessive variabili in gioco, il cui numero diventa troppo elevato per poter essere gestito con i calcolatori. Bisogna trovare delle modalità per ridurre la dimensione sia dello stato (ossia le variabili interne, che "descrivono" la struttura del sistema stesso), sia dei controlli e dei disturbi;
- Il modo più semplice di simulare un sistema è operare *fuori linea*, prima che il sistema entri in attività, o prima che l'alternativa scelta venga attuata. Così facendo, semplicemente, si osservano quali sono i risultati conseguenti alle scelte effettuate, ma non è il modo migliore di procedere, dal momento che è possibile migliorare

sensibilmente le prestazioni se si istruisce il sistema, in modo che apprenda dagli errori commessi e non li ripeta più (*politiche con apprendimento*). Ancora meglio è riuscire ad operare durante la vita del sistema, così da poter decidere passo dopo passo come procedere, in base alla precisa situazione che si è verificata all'istante precedente (politiche *in linea*). Ciò è vantaggioso rispetto all'azione fuori linea, anzitutto perché non obbliga a calcolare tutte le possibili soluzioni a priori, e poi consente di procedere conoscendo la reale situazione attuale e non quella prevista con una politica fuori linea. Anche in quest'ultimo caso è possibile operare con o senza apprendimento, ma le richieste computazionali diventano sempre più onerose, pertanto di sovente irrealizzabili.

I punti elencati in precedenza evidenziano come il problema introdotto sia di una portata e di una complessità molto vaste. Potrebbe sembrare semplificabile nel cercare la soluzione più banale, ossia risolvere una politica in linea con apprendimento usando un modello molto semplice ed una discretizzazione né troppo ampia, né troppo fitta, ma ciò non è possibile anzitutto perché ogni sistema naturale è diverso dagli altri, in secondo luogo perché non per tutte le descrizioni sono possibili tutte le strategie: occorre individuare, caso per caso, cosa è possibile o non è possibile fare, e all'interno di quanto disponibile, qual è la strada migliore da percorrere.
Anche dal punto di vista dei modelli che si possono adottare occorre ipotizzare differenti approcci: non sempre, infatti, le opzioni più immediate e più semplici forniscono i risultati migliori in tempi accettabili o quantomeno ragionevoli. Si deve considerare, sempre, l'opzione di utilizzare modelli più complicati, meno immediati sia nella comprensione sia nell'uso, ma maggiormente indicati per trattare il problema in oggetto.

Negli ultimi anni si sono sviluppati nuove tipologie di modelli, sempre più complessi e raffinati: tra questi, si collocano le Reti Neurali, che rappresentano una nuova frontiera, ancora in gran parte da scoprire. Con le reti neurali si tenta di riprodurre il funzionamento di uno dei più complessi degli apparati dell'essere umano, ossia il sistema nervoso.
Questi modelli possono apprendere, mediante relazioni matematiche non particolarmente difficili implementate al loro interno, dagli errori compiuti nelle varie simulazioni effettuate nel tempo, e modificare ricorsivamente i propri parametri, fino a che non riescano ad ottenere i risultati più vicini all'ottimo.
Se il processo di apprendimento viene effettuato con un grandissimo numero di simulazioni, sarà molto probabile che la rete neurale riesca a rispondere bene a tutte, o quasi, le casistiche che in natura si possono verificare: dato che, pur complesso e moderno, rimane sempre un modello, anche la rete neurale ha a che fare con l'incertezza, che accompagna imprescindibilmente tutte le simulazioni, per di più effettuate sui sistemi naturali.
Sarà quindi compito dell'analista dare una buona forma ed un buon addestramento alla rete neurale, in modo tale che la stocasticità, non eliminabile, sia ridotta al minimo.

L'idea alla base del presente lavoro di Tesi è, dunque, questo: utilizzando un modello di un sistema ambientale esistente (nella fattispecie un sistema idrico, assai versatile e facilmente studiabile), già ampiamente validato e simulato, si costruisce e addestra una rete neurale che ben descriva il comportamento del sistema, soggetto ad un numero di ingressi (rappresentati dagli afflussi) molto grande e strutturati in modo molto differente tra loro, con l'intento di rappresentare tutti i casi che si possono verificare in natura. La validazione del modello neurale viene effettuata mediante un semplice confronto dei risultati, tra quanto prodotto dalla rete e le uscite di un semplice software di ottimizzazione lineare, anch'esso ampiamente collaudato, il quale, per ognuno degli ingressi, ne fornisce l'uscita ottima, rappresentata dal beneficio ricavabile dalla vendita di energia elettrica prodotta dagli impianti collocati nel sistema.

1

RICERCA DEL FLUSSO OTTIMO NELLE RETI DI SERBATOI

Le scelte che riguardano i sistemi ambientali, siano essi idrici o di altro tipo, sono molto complesse e difficili da prendere, perché non esistono "strade giuste o strade sbagliate" che conducono ad un risultato ottimale, esistono solamente modelli che vi si avvicinano di più o di meno. Occorre agire spesso rapidamente e in modo adeguato, affinché gli effetti sia sul sistema stesso sia sull'ambiente che lo circonda non siano troppo gravi o irreparabili. Per aiutare il Decisore, cui spetta l'ultima azione, a seguire la strada giusta, sono stati sviluppati diversi modelli e differenti metodologie, che il presente capitolo ha lo scopo di descrivere brevemente.

1.1 SISTEMI AMBIENTALI: RETI DI SERBATOI

1.1.1 RUOLO DEI SISTEMI IDRICI

Dal momento che trattare le problematiche di ottimizzazione in modo astratto è molto complesso e per nulla banale, si cerca, in questa prima fase del lavoro di Tesi, di spiegarne le linee generali facendo riferimento a casi realistici.
I sistemi ambientali che necessitano di essere controllati sono molto numerosi, e spaziano in moltissimi ambiti, differenti per tipologia, struttura e funzionamento.
Di certo, tra tutti i sistemi, quelli idrici sono quelli che maggiormente devono essere soggetti a controlli e regolamentazioni, perché dal loro buon funzionamento dipendono non solo tantissime attività umane, ma anche la salvaguardia e la tutela dell'ambiente, dei paesaggi e degli ecosistemi che vivono e si sviluppano intorno ad essi.
Oggi, nessun corso d'acqua è lasciato libero di evolvere in modo spontaneo: storicamente, dapprima sono stati costruiti argini e pareti di contenimento, affinché il corso dalla sorgente alla foce fosse controllato e guidato, per evitare danni alle campagne ed ai centri abitanti circostanti l'alveo, soprattutto nei periodi di piena, quando il livello delle acque si alza, anche di molto, in tempi piuttosto rapidi. Successivamente, sono state costruite delle dighe, degli sbarramenti in corrispondenza delle bocche degli emissari dai laghi (trasformandoli in *serbatoi* [Soncini Sessa, 2004]), affinché fosse possibile garantire sempre il corretto approvvigionamento idrico, rilasciando una maggiore quantità di acqua quando necessario (ad esempio per irrigare i campi), e trattenendola invece in altri momenti.
La trasformazione dei laghi in serbatoi ha reso necessario un controllo costante del livello dell'invaso, in modo tale sia da garantire l'incolumità di coloro che vivono sulle sponde, sia da permettere sempre il corretto svolgimento di tutte le attività per cui l'acqua è necessaria

(attività di pesca, agricoltura, navigazione, produzione di energia elettrica, crescita e sviluppo di ecosistemi, …).
Tutto quanto detto finora si verifica poiché l'uomo, da sempre, ha tentato di trasformare il territorio secondo le sue necessità, adattandovi i sistemi naturali, in modo che da questi potesse essere soddisfatto un sempre maggior quantitativo di bisogni.
Forse, il più evidente esempio della trasformazione dei sistemi naturali per soddisfare le necessità umane riguarda proprio i sistemi idrici, che svolgono numerose funzioni:

- L'acqua è un elemento essenziale alla vita di tutte le specie di piante ed animali, non solamente dell'uomo;
- I sistemi idrici consentono, grazie ad opportune realizzazioni, di irrigare i campi agricoli, da cui l'uomo ricava una consistente parte degli alimenti con cui si nutre;
- I sistemi idrici assolvono, oggi, ad una funzione importantissima: consentono di produrre, disponendo di opportuni dislivelli (*salti motore*), energia elettrica, grazie alle centrali idroelettriche;
- Da ultimo, ma non certo per importanza, una buona regolazione ed un utilizzo "saggio" dei sistemi idrici consente di laminare gli effetti delle piene durante le stagioni umide e garantendo, per converso, un maggiore approvvigionamento idrico nei periodi di siccità; la laminazione si attua con ampie escursioni del livello degli invasi dei serbatoi

Al fine di ottimizzare lo svolgimento dei compiti cui i sistemi idrici sono chiamati ad assolvere, l'uomo ha cercato di massimizzarne lo sfruttamento, sia regolando i laghi ed i fiumi naturali che questi collegano, sia costruendo nuovi manufatti, come i canali, le dighe e le traverse, che ampliano le possibilità di sfruttamento e consentono un maggiore controllo delle acque.
Per tutte le motivazioni esposte in precedenza, si rende necessario riuscire ad ottimizzare l'allocazione della "risorsa acqua" all'interno di un sistema idrico. Il migliore modello per tale scopo è quello della *rete di serbatoi*.

1.1.2 STRUTTURA DI UNA RETE DI SERBATOI

Una rete di serbatoi è un insieme finito di nodi che sono tra di loro collegati da archi unidirezionali, anche essi in numero finito, entro i quali scorre il flusso del bene [Taddio, Togni, 1993] [Grossman *et al.*, 1995].
Nello specifico, una rete di serbatoi è anche una rete di trasporto, da intendersi come un grafo orientato, connesso, e privo di autoanelli, cioè canali che escono da un serbatoio e rientrano nello stesso. In una rete di questo tipo deve essere previsto un nodo sorgente, privo di archi entranti, ed un nodo pozzo, privo di archi uscenti, dove con sorgente e pozzo (sono da intendere come delle "porte di collegamento" tra il sistema ed il mondo esterno, motivo per cui non deve essere presa alcuna decisione relativa ad essi [Grossman *et al.*, 1995]); ad ognuno degli archi è associato un valore di vincolo sulla capacità massima e minima [Colorni, 1984] [Busacker, 1965] [Grossman *et al.*, 1995].
Evidentemente, nel riferimento ad un sistema idrico, i nodi rappresentano non solo i serbatoi (capaci di accumulare risorsa, e rilasciarla nel tempo), ma anche le traverse, mediante le quali il flusso proveniente da un arco viene suddiviso in due o più archi differenti: la decisione di ripartizione tra i flussi è una delle variabili decisionali da includere nel Problema di Progetto, a differenza di quanto accade per le semplici confluenze, dove flussi provenienti da archi distinti si unificano all'interno di un nuovo arco più capiente [Grossman *et al.*, 1995], mentre gli archi unidirezionali rappresentano i fiumi, o canali, entro i quali l'acqua scorre per gravità

esclusivamente da monte verso valle. La capacità massima associata ad ogni arco rappresenta la massima portata che può circolare nel canale.
Una rete di serbatoi, normalmente, riceve degli ingressi sottoforma di afflussi meteorici da quello che è il bacino imbrifero, o bacino idrologico [Greppi, 2005] [Soncini Sessa, 2004], ossia tutta la porzione di territorio che convoglia alla sezione di chiusura le acque che si riversano sul sistema o scorrono su di esso. La tipologia e la quantità di detti afflussi non sono facilmente ed immediatamente determinabili: essi dipendono dalla localizzazione del sistema, in misura maggiore dalla latitudine rispetto alla longitudine, dalla climatologia del territorio e da altri fattori, tra cui le attività ed influenze umane (può capitare, ad esempio, che in zone soggette a climi aridi, quali i deserti, vengano costruiti delle grosse cisterne di raccolta delle acque per uso agricolo: localmente, in presenza di forti evaporazioni, si osservano degli incrementi di precipitazione, dovuta proprio all'evaporazione dell'acqua raccolta e conservata nelle cisterne). Tutto ciò rende fortemente aleatorio e stocastico il ruolo, e la natura degli afflussi, tipicamente considerati come dei *disturbi stocastici*. Tali afflussi, nella modellizzazione della rete, si supporranno provenire da un unico nodo, quello identificato con la sorgente.
Parimenti agli ingressi, la rete avrà dei flussi idrici in uscita, da determinare risolvendo il Problema di Progetto, per garantire il miglior svolgimento di tutte le attività elencate in precedenza: i *rilasci*, che devono essere determinati ad ogni passo temporale con cui la dinamica del sistema viene determinata. Essi si ipotizzano convogliare tutti in un unico punto, il pozzo, che generalmente è un distretto irriguo, oppure un punto qualsiasi che si colloca come chiusura del sistema.
Senza entrare in questa sede nei dettagli della rete di serbatoi fisicamente esistente sulla quale si basa il presente lavoro (quella del bacino del Fiume Zambesi, dettagliata nel Capitolo 4), si definisce, nella Figura 1.1 a pagina seguente, un semplice schema di una generica rete di serbatoi, nella quale si mettono in evidenza la simbologia e le variabili che normalmente si considerano in un modello.

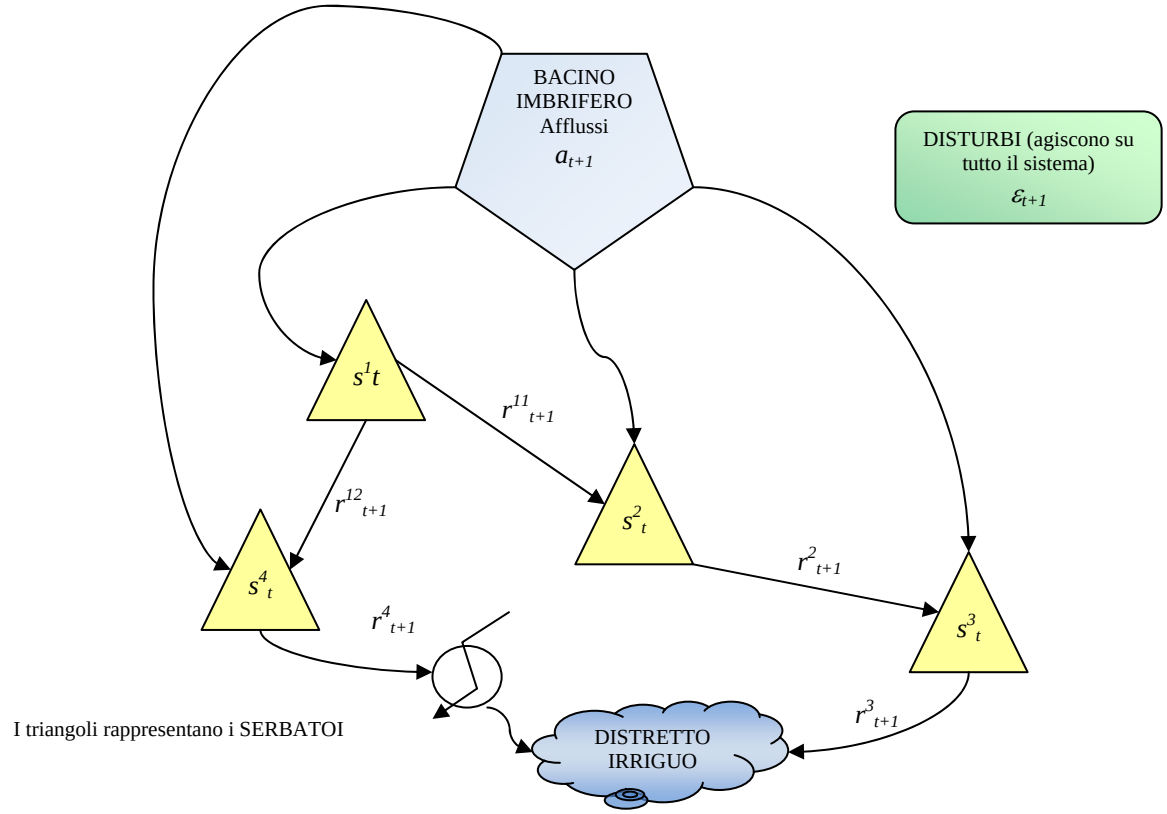

Figura 1.1 – Esempio di rete di serbatoi, in cui compaiono il bacino imbrifero che determina gli afflussi (a), i serbatoi (s), il distretto irriguo cui una parte dei rilasci(r) dei serbatoi è diretta,una centrale per la produzione di energia idroelettrica e i disturbi stocastici che agiscono su tutto il sistema; sovente gli stessi afflussi vanno trattati come un disturbo stocastico, e si suppongono tutti generati dal "nodo sorgente", il bacino imbrifero, mentre tutti i flussi convogliano nel "nodo pozzo", ossia il distretto irriguo; la simbologia utilizzata è quella convenzionalmente adottata

Usualmente, la rete di serbatoi è un sistema dinamico, che muta quindi nel tempo; le equazioni che descrivono il modello devono, pertanto, essere anch'esse dinamiche. La più semplice equazione che descrive una rete di serbatoi è l'equazione di transizione di stato, che descrive come muta l'invaso ogni serbatoio nel tempo. Prendendo come riferimento la simbologia della Figura 1.1, tale equazione si scrive come

$$s_{t+1} = s_t + a_{t+1} - r_{t+1} + \varepsilon_{t+1} \qquad (1.1),$$

dove *t* indica lo specifico passo temporale, *s* indica la variabile di stato che muta nel tempo (si noti che essa compare sia a sinistra sia a destra del simbolo di uguale), e precisamente rappresenta il volume di invaso del lago, *a* individua gli afflussi (essendo, come detto, stocastici, il loro valore sarà noto realmente soltanto al passo temporale successivo; prima, semplicisticamente, essi possono solo essere previsti o stimati), ε indica tutti i generici disturbi che insistono sul sistema (per esempio, l'evaporazione), mentre *r* indica il volume idrico che dovrà essere rilasciato, al fine di ottenere il miglior risultato per l'assolvimento delle funzioni che si chiedono al sistema stesso, come spiegato nel precedente paragrafo 1.1.1.

Come detto, numerosi sono coloro che traggono un qualche beneficio dal sistema idrico: questi attori saranno molto interessati ad alcune attività, e sostanzialmente indifferenti ad altre (ad esempio, i proprietari dei campi agricoli saranno molto interessati a massimizzare il profitto generato dai campi stessi, e quindi riterranno fondamentale che una grossa quantità di acqua consenta loro, in determinati periodi dell'anno, di irrigare le colture per raccoglierle e venderle; essi saranno invece sostanzialmente indifferenti alla quantità di acqua che verrà inviata alle centrali idroelettriche per la produzione di energia elettrica; chi si occupa di

salvaguardia e tutela ambientale avrà invece interesse a veder preservati gli habitat naturali e le specie che in essi vivono, mentre risulterà indifferente sia al profitto agricolo, sia a quello idroelettrico). Ognuno di questi attori, definito *portatore di interesse* esprime il raggiungimento di un certo valore dell'obiettivo mediante una funzione di utilità, che rappresenta proprio la soddisfazione del portatore nei confronti dell'obiettivo. [Soncini Sessa, 2004]
Per gli scopi che interessano al presente lavoro, comunque, si considereranno solamente funzioni di utilità relative alla produzione di energia elettrica. Esse avranno forma monotona crescente, con valori marginali tanto più bassi quanto più alti saranno i flussi idrici circolanti. Per quanto riguarda la struttura generale di tali funzioni, si rimanda al seguente Paragrafo 1.1.3, mentre per i dettagli sul caso in esame si rimanda al Capitolo 4.

1.1.3 TRASFORMAZIONE DELLA RETE IN SUPERGRAFO

Per poter utilizzare il software di ottimizzazione per il calcolo dell'allocazione ottima dei flussi (Aquafun, il cui funzionamento è spiegato nell'Appendice A1), le reti devono essere definite in un modo piuttosto rigido, date la scarsa flessibilità del programma stesso.
Anzitutto, devono essere considerate solamente delle reti proprie e connesse, costituite da non meno di due nodi ed un arco, questi definiti in modo tale che qualunque partizione dell'insieme dei nodi in due sottoinsiemi disgiunti e complementari tra loro, si abbia sempre un arco che presenta il nodo di partenza in un sottoinsieme e quello di arrivo nell'altro. Come facilmente immaginabile dalle descrizioni effettuate nei paragrafi precedenti, i nodi sorgente devono essere caratterizzati da un flusso positivo entrante, i nodi pozzo da un flusso negativo uscente, ed, infine, i nodi *di scambio*, nei quali circolano solamente flussi interni alla rete, devono essere caratterizzati da un flusso netto nullo. Nella definizione degli archi, invece, ed anche questa caratteristica è intuitiva, deve essere quantificata la funzione di utilità di trasporto del flusso, e devono essere inseriti i vincoli di minima e massima capacità.
Per semplificare di molto la ricerca dell'allocazione ottima dei flussi, le funzioni di costo devono essere di forma semplice, in modo che l'ottimizzazione stessa possa essere compiuta rimanendo nell'ambito della programmazione lineare. Tali funzioni vengono associate solamente agli archi, e determinate in base ai valori di flusso circolanti in questi, esclusivamente per mantenere leggero il quantitativo dei dati concorrenti alla definizione della rete. Specificatamente, le funzioni di utilità devono avere tre caratteristiche:

1) Essere continue;
2) Essere derivabili;
3) Essere concave (verso il basso, ma se di costo, verso l'alto);
4) Essere lineari a tratti, quindi con derivata a scalino decrescente per utilità crescenti, per poter essere facilmente implementate nel software di ottimizzazione.

L'adozione di funzioni di utilità con queste caratteristiche consente senza problemi di applicare algoritmi di ottimizzazione lineari, per le cui descrizioni e peculiarità si rimanda all'Appendice A3.
La metodologia di risoluzione più immediata di un problema di allocazione ottima del flusso consiste nella *trasformazione della rete* in un supergrafo, e nella ricerca del minimo costo: questa seconda condizione si ottiene semplicemente cambiando il segno alle funzioni di utilità, mentre per ottenere il supergrafo occorre applicare una procedura più complessa, spiegata nel seguito della presente trattazione. [Grossman *et al.*, 1995]
Una rete reale può presentare un numero di sorgenti e pozzi maggiore di uno: per semplicità, le reti modellizzate dovranno essere ricondotte ad avere una sola sorgente (che sarà chiamata *fittizia*), ed un solo pozzo, anch'esso *fittizio*. Questi elementi saranno collegati rispettivamente alle sorgenti reali ed ai pozzi reali mediante archi ad utilità nulla e con capacità pari,

evidentemente saturata, ai valori reali di afflusso o rilascio. È possibile, dunque, mantenendo inalterate le caratteristiche di generalità, considerare reti con una sola sorgente ed un solo pozzo.
La rete naturalmente evolve nel tempo: non è possibile, nemmeno facendo ricorso alle più moderne tecnologie, effettuare una simulazione su orizzonte infinito, pertanto occorre stabilire un numero di periodi di simulazione (per i casi trattati nel presente lavoro la discretizzazione sarà a passo mensile).
Per quanto riguarda la ricostruzione della dinamica del sistema, viene creata una copia della rete per ognuno dei periodi di simulazione, introducendo anzitutto un'*ipersorgente* che collega le sorgenti ad ogni passo mediante canali a costo nullo con capacità pari ai flussi totali del periodo in questione, ed anche un *iperpozzo* di raccolta dei flussi. Essi, tuttavia, devono poter essere allocati in differenti modi, in maniera tale da consentire la ricerca dell'ottimo: pertanto, i canali che conducono dai pozzi all'iperpozzo sono a capacità infinita, così da semplificare la definizione del sistema vincolare.
I serbatoi, come facilmente intuibile, possono immagazzinare risorsa al loro interno: per tale motivo ogni serbatoio al tempo *t* è collegato al corrispondente serbatoio al passo successivo, *t+1*.
La rete così definita, pertanto, si caratterizza da due tipi di archi: quelli fisici che collegano i diversi nodi all'interno di uno stesso periodo, e quelli temporali, che invece collegano gli stessi nodi a due istanti successivi; introducendo tale modellizzazione, l'accumulo di risorsa nei serbatoi non deve più essere trattata distintamente dagli altri flussi in circolazione nella rete, dal momento che anche quello accumulato è un flusso che scorre negli archi temporali: i serbatoi possono, quindi, essere considerati come nodi qualsiasi.
La rete modellizzata in questo modo, con le replicazioni della struttura nel tempo, viene rappresentata in quello che è chiamato *supergrafo*, una cui generica rappresentazione è riportata nella seguente Figura 1.2.

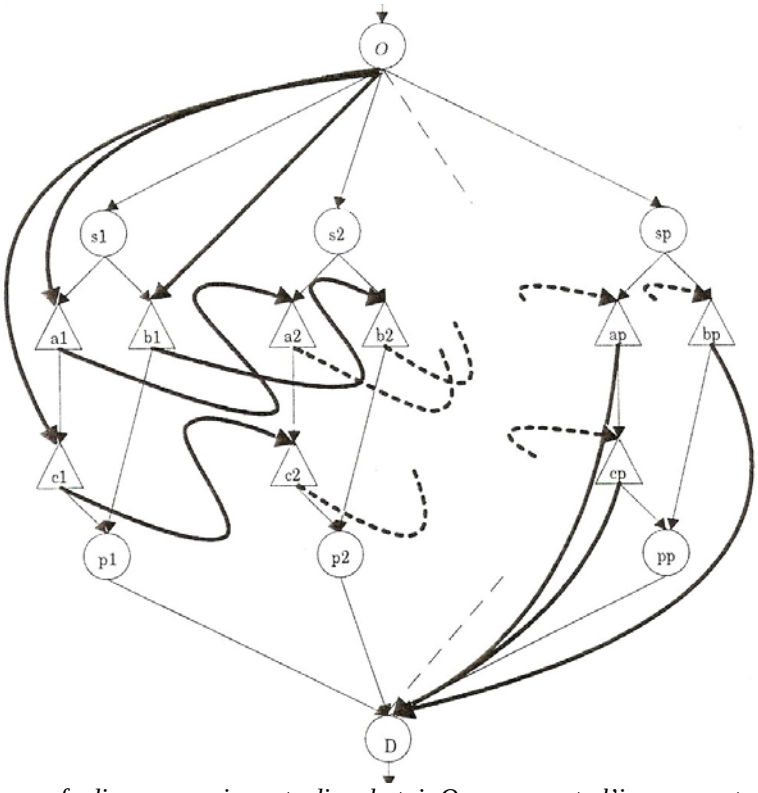

Figura 1.2 – Supergrafo di una generica rete di serbatoi; O rappresenta l'ipersorgente, S1, S2, …, Sp le sorgenti fittizie, a, b, c i serbatoi, p i pozzi e D l'iperpozzo (da Taddio, Togni, op. cit.*)*

Mancano, a questo punto, ancora due fenomeni che devono essere opportunamente modellizzati: le perdite dalla rete e gli eventuali ritardi nel trasporto del flusso lungo i canali.
Nel caso dei serbatoi, per modellizzare le perdite, si considera un tasso costante, che tiene conto sia dei fenomeni evaporativi, sia delle perdite vere e proprie: è un'ipotesi senz'altro riduttiva, ma la cui adozione non inficia la validità dei risultati e non espande troppo la complessità computazionale del problema [Taddio, Togni, 1993]. Nel supergrafo, si introduce un nodo dispersore che ha il compito di accumulare tutte le perdite e rilasciarle nell'iperpozzo.
Per quanto riguarda, invece, i ritardi nel trasporto negli archi, potrebbe capitare che alcuni flussi raggiungano i serbatoi di destinazione dopo l'orizzonte temporale, fenomeno che non è fisicamente né sensato, né possibile. Tali archi si collegano, quindi, ad un serbatoio relativo ad un periodo fittizio, con l'unico scopo di raccogliere i flussi uscenti dall'orizzonte di progetto. Occorre pertanto apportare delle modifiche al supergrafo. La seguente Figura 1.3 rappresenta tali modifiche, con l'introduzione sia del nodo dispersore, sia di quello che raccoglie i flussi fittizi.

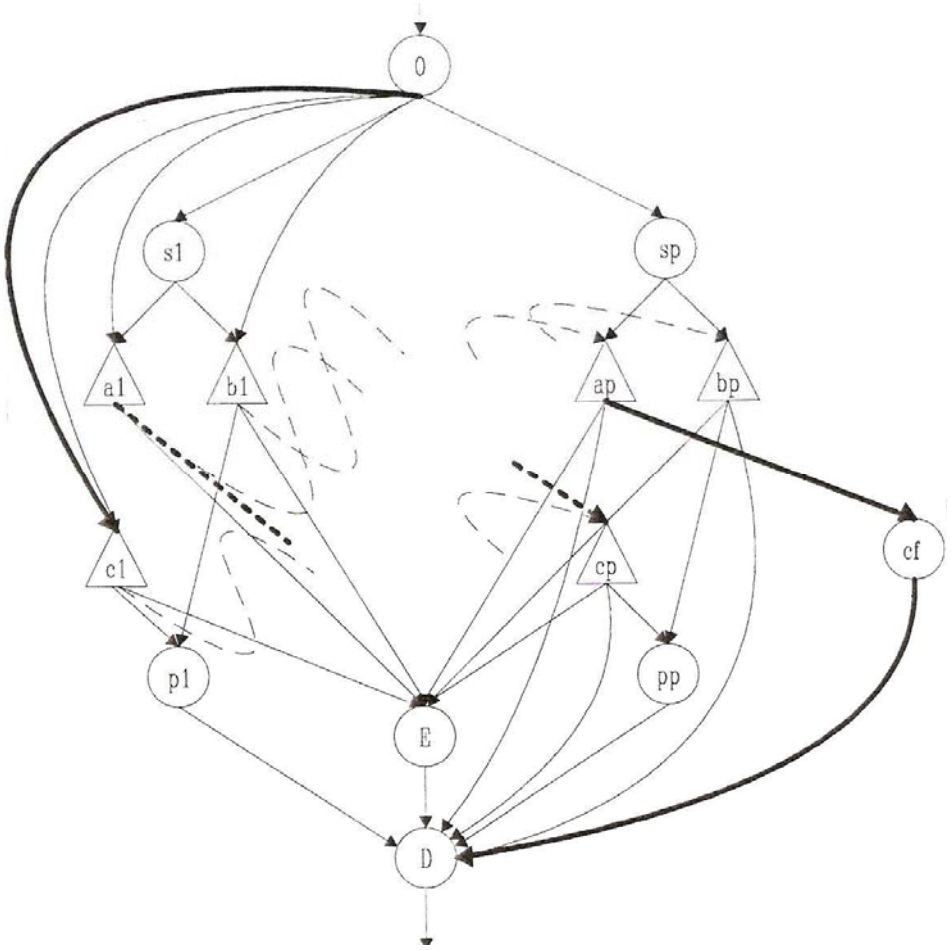

Figura 1.3 – Supergrafo modificato; in esso compaiono il nodo dispersore E e quello fittizio cf, adibito alla raccolta dei flussi uscenti dall'orizzonte di progetto.

1.2 POLITICA DI GESTIONE DI UN SISTEMA COMPLESSO

I sistemi naturali sono delle realtà molto complesse, formate da numerosi elementi e fenomeni interagenti tra loro in un funzionale equilibrio. L'alterazione di questo ha sempre delle conseguenze, raramente impercettibili, molto più spesso assai devastanti.
Per tale motivo, occorre "far funzionare" il sistema antropizzato nel modo migliore possibile, in modo tale che le conseguenze sull'ambiente e sui fattori interagenti siano ridotte, o non dissimili da quelle che si sarebbero avute se il sistema stesso fosse stato lasciato evolvere in modo naturale, oppure migliori.
Occorre pertanto definire una *politica di regolazione*, ossia una sequenza (si definisce in tal modo un elenco ordinato di oggetti che non segue una regola particolare; si definisce invece "successione" un elenco, pur sempre ordinato, ma con una regola particolare che lega tra loro gli elementi, come ad esempio quella di Fibonacci [Soncini Sessa, 2004]) di azioni di controllo, che portino al risultato migliore conseguibile.
Mediante la definizione della politica, si riesce a semplificare la forma del Problema di Progetto, perché anche le azioni gestionali sono ricondotte a quelle di pianificazione, con l'evidente ed indubbio vantaggio che la stesura e la risoluzione del Problema non deve più avere due distinte forme, ma può essere ricondotto ad una sola, come sarà spiegato nel paragrafo successivo.
La gestione del sistema deve essere programmata per ognuno dei passi temporali di vita del sistema. L'ampiezza di detti passi non può essere stabilita a priori, perché ogni struttura ha delle sue proprie caratteristiche di funzionamento: esistono sistemi che devono essere controllati ogni ora, altri giornalmente, altri, come nel caso del complesso analizzato in questo lavoro, mensilmente.
La politica, pertanto, si costituisce come un elenco di funzioni che producono, ad ogni passo, il miglior risultato ottenibile in un dato contesto avendo a disposizione un certo tipo di tecnologie. Tali funzioni prendono il nome di *leggi di controllo*, la cui uscita spesso non ha senso che sia unica, perché nei casi reali si hanno molte situazioni di partenza differenti che forniscono risultati equivalenti, almeno dal punto di vista delle prestazioni e delle conseguenze. Si parla, allora, di *leggi di controllo a più valori – APV* se producono più risultati equivalenti (altrimenti, se uno solo è il risultato determinato da una configurazione di input, si parla di *legge ad un solo valore – AUV*). In conseguenza di questo, si avranno anche *politiche a più valori* oppure *politiche ad un solo valore*.

1.3 LA FORMA DEL PROBLEMA DI PROGETTO

Si è detto che la procedura migliore per individuare delle alternative e i loro effetti sul sistema è quella di impostare, e risolvere, un Problema di Progetto, al fine ottenere una Politica che consenta di gestire il sistema nel modo migliore possibile.
Per la definizione di un Problema si fa riferimento alla rete di serbatoi, definita nel Paragrafo 1.1.2, ed è possibile, per semplicità espositiva, scindere il Problema di Progetto in due distinte parti [Stam *et al.*, 1998]:

1) Nel caso in cui ci si occupi solamente di azioni pianificatorie, che devono essere prese una volta per tutte, si definisce un *Problema di pura pianificazione*;
2) Nel caso in cui, invece, di interesse sia unicamente la gestione continua del sistema (sul quale non si prevedono interventi che ne alterano la struttura fisica), il Problema di progetto prende il nome di *Problema di pura gestione* ovvero *Problema di controllo ottimo*.

Usualmente, sul sistema idrico che viene antropizzato, è necessario sia attuare degli interventi invasivi che ne alterano la struttura, sia, successivamente, controllarlo e regolarlo in modo opportuno: ecco che allora si redige, e si risolve, un Problema di progetto completo, nel quale, però, le due parti descritte precedentemente si mantengono separate. Esse vengono risolte una dopo l'altra: prima la parte di pianificazione e poi la parte di gestione, che viene però ricondotta, in qualche modo, alla precedente, tramite la formulazione di una *politica di gestione*. Così facendo, la forma del Problema di Progetto si può ridurre alla sola parte di pianificazione, che viene qui descritta.
La formulazione più generale è:

$$J(u^{p*}, p^*) = \min_{u^p, u_t} {}^{Criterio}_{\{\varepsilon_t\}_{t=0,1,\dots,h}} \left[i\left(s_0^h, u^p, u_0^{h-1}, w_0^{h-1}, \varepsilon_1^h\right)\right] \qquad (1.2),$$

dove:

- $J(u^p, p^x)$ indica l'*obiettivo*, lo scopo della gestione del sistema, funzione sia delle alternative di pianificazione scelte u^p, sia della politica adottata p^* (è ovvio che il valore dell'obiettivo cambia a seconda della politica che si adotta; esso deve usualmente essere minimizzato, sia rispetto alla decisione pianificatoria sia rispetto a quella gestionale ($\min_{u^p, p^*}$ ...). Tuttavia, facendo riferimento alle utilità idroelettriche introdotte in precedenza, ha senso che esse siano massimizzate, non minimizzate: mediante un semplice cambio di segno ci si può ricondurre alla forma canonica del Problema di Progetto;
- Su tutti i sistemi naturali agiscono una serie di disturbi, definiti da ε, che complicano la ricerca della soluzione perché sono in ogni caso non deterministici, (quindi rendono non deterministica anche la soluzione del problema): occorre applicare un opportuno criterio per trattarli (di solito si usa il *valore atteso di Laplace* o il *criterio della prestazione scelta di Wald*);
- L'obiettivo viene calcolato tramite gli *indicatori i*, i quali indicano le utilità delle diverse alternative per ognuno dei soggetti interessati al problema. Detti indicatori si calcolano, solitamente, mediante funzioni, le quali dipendono da numerose variabili: lo *stato del sistema* (*x* o, nel caso specifico di una rete di serbatoi, *s*), cioè l'insieme delle variabili che definiscono la condizione presente del sistema, determinata dalla storia passata di questo e influenzante la sua storia futura [Soncini Sessa, 2004], i disturbi agenti (ε, che influenzano le prestazioni del sistema), le scelte di pianificazione effettuate (u^p), la politica adottata, da intendersi come la sequenza dei controlli ($u_{t,t=0,\dots,h-1}$), lo *scenario di progetto* (*w*), costituito da tutte quelle grandezze che mutano nel tempo, influiscono sulla vita e la struttura del sistema, ma il cui andamento non viene modificato dalle decisioni prese sul sistema medesimo;
- Occorre porre un limite temporale alla risoluzione del Problema: bisogna definire un *orizzonte di progetto* (finito, infinito o mobile), che viene indicato con *h* nell'equazione (1.2).

La descrizione dell'obiettivo, effettuata mediante l'equazione (1.2), è soggetta a numerosi vincoli, che riguardano la descrizione di tutte le variabili descritte ai punti precedenti, ed anche la stessa *politica* (scritta, in questo caso, come AUV)

$$p \triangleq \{m_t;\ t = 0,1,\dots,h\} \qquad (1.3),$$

dove le *m* rappresentano le *leggi di controllo* che determinano i controlli *u*.

Volendo specificare la forma del Problema di Progetto per una rete di serbatoi, si possono considerare, anzitutto, come variabili di stato *s* i vettori che descrivono, ad ogni istante, i volumi di invaso di tutti i serbatoi che costituiscono la rete. Appare quindi chiaro che tali volumi non possono assumere valori qualsiasi, ma devono essere vincolati tra un minimo ed un massimo, dati dalla fisica del sistema (usualmente, i canali sono considerati come dei semplici archi di collegamento tra serbatoi, dunque sono privi di stato: ad essi sono associati solamente un'utilità che fa riferimento alla quantità di risorsa idrica trasferita tra i due nodi, ed eventualmente un ritardo di trasporto, quantificabile in un certo numero di istanti). Come indicatori usualmente si considerano le quantità di energia elettrica prodotta, e, conseguentemente, venduta, da cui è possibile risalire all'utilità, mentre i disturbi principali sono gli afflussi al sistema, di natura stocastica, e quindi incerta. Evidentemente, l'obiettivo del problema sarà massimizzare la somma complessiva delle utilità, tramite l'allocazione ottima dei flussi uscenti dai vari serbatoi: essi andranno ripartiti correttamente tra i canali, in modo tale che il beneficio complessivo sia il più alto possibile. In altri termini, occorre risolvere un *Problema di Assegnazione Dinamica* (*Dynamic Assignment Problem – DAP*), che consiste nel determinare l'utilizzo ottimale delle traverse e dei serbatoi di accumulo, al fine di massimizzarne il beneficio. [Grossman *et al.*, 1995]

Per la risoluzione del Problema qui presentato in forma sintetica si fa ricorso a differenti approcci, sviluppabili in situazioni differenti, dipendenti sia dalla struttura del sistema che si sta analizzando, sia dalla tipologia di soluzione che si cerca di ottenere. In estrema sintesi, le soluzioni più comuni vengono cercate ricorrendo a quello chiamato *approccio funzionale*, che consiste nella ricerca della politica mediante la definizione delle funzioni che compongono le leggi di controllo da cui essa è formata: è una procedura molto onerosa, perché si ha quasi sempre a che fare con un numero molto grande, quando non infinito, di variabili e valori, che ne inficiano le possibilità risolutive.

Una buona alternativa è costituita dall'*approccio parametrico*, mediante il quale si rinuncia a cercare le singole funzioni, ma si definisce la tipologia (detta *classe*) cui esse devono appartenere: ognuna di queste classi è descritta da un numero finito di parametri, per cui anche la politica rientra nelle dimensioni finite del numero di parametri individuati. Questo è il caso trattato in questo lavoro, dal momento che le *reti neurali* sono definite da un numero finito (il più piccolo possibile) di parametri.

Entrambi tali approcci sono possibili se tutta l'informazione necessaria alla definizione delle equazioni del Problema di Progetto è disponibile a priori, opzione che spesso non è verificata: è bene allora cercare di dare al sistema la possibilità di apprendere sia dalle decisioni errate prese in passato, sia dalla nuova informazione mano a mano acquisita (*approccio con apprendimento*), oppure si può rinunciare a descrivere il sistema dal punto di vista modellistico, calcolando la politica con un *approccio model – free*.

1.4 PROGRAMMAZIONE DINAMICA

1.4.1 DEFINIZIONE

La *programmazione dinamica* (individuata con la terminologia anglosassone dalla sigla *DP – Dynamic Programming*) è la prima, più semplice metodologia risolutiva dei Problemi di Progetto. Sinteticamente, essa è una tecnica di progettazione degli algoritmi che prevede la scomposizione del Problema in elementi più semplici, detti sottoproblemi, per ognuno dei quali si cerca una soluzione mediante tecniche di ottimizzazione.
La Programmazione Dinamica può essere applicata, in via teorica, a tutti i problemi, anche se il suo principale difetto riguarda la *maledizione della dimensionalità*, dal momento che applicandone gli algoritmi, si va incontro ad un'espansione delle dimensioni delle variabili di interesse troppo elevata, soprattutto quando lo stato ed i controlli del sistema assumono dimensioni molto rilevanti. Ciò è dovuto al fatto che occorre calcolare tutti i valori per tutti gli istanti di simulazione, e questo spesso richiede sia ingenti capacità di memoria, sia ampie capacità di calcolo, che, ancora oggi, possono rappresentare un problema.
Per capire i concetti principali legati alla Programmazione Dinamica, è bene fare riferimento al ragionamento effettuato da Bellman, colui che definì e dimostrò l'equazione della Programmazione Dinamica [Bellman, 1957]. Il punto di partenza riguarda la nascita della Teoria della DP, sviluppata per risolvere problemi matematici concernenti gli studi di processi a più stadi di decisione, spesso anche a più decisori diversi. Tali decisioni, in un modo o nell'altro, influiscono sulle variabili di stato del sistema, dal momento che equivalgono a delle trasformazioni di stato, e, inoltre, i risultati di quanto deciso ai passi precedenti guidano nel prendere le decisioni future, con lo scopo ultimo di massimizzare alcune funzioni che permettono di calcolare le variabili che definiscono lo stato finale del sistema.
L'idea da cui nasce la Programmazione Dinamica è molto semplice, ed immediata, e consiste nel considerare, per risolvere un problema qualunque, tutte le possibili sequenze di decisioni per lo specifico sistema, calcolare i valori finali per ognuna di esse, e quindi valutare il massimo – o i massimi – tra queste.
Sebbene questo modo di procedere sia il più corretto, e il migliore dal punto di vista teorico, non è quasi mai praticabile, non tanto per l'eccessiva richiesta computazionale (cui i moderni calcolatori possono quasi sempre far fronte tranquillamente), quanto piuttosto per la richiesta e la ricerca di informazioni preliminari necessarie.
L'idea della programmazione dinamica è proprio quella di semplificare notevolmente il quantitativo di informazioni da possedere, rendendo necessario disporre solamente dei valori delle grandezze unicamente per l'istante attuale, poiché si dispone di un'equazione che riesce a determinare la scelta migliore solamente in funzione delle grandezze relative all'istante per il quale si stanno eseguendo i calcoli. Si capisce che, così facendo, si riduce di molto l'onere di informazioni richiesto, perché, si riesce a calcolare il valore di una grandezza che influenza tutta la storia futura, ed influenzata da tutta quella passata, senza dover capire come si comporterà il sistema successivamente, e senza aver previsto prima come esso si sarebbe comportato al momento del calcolo.
Tuttavia, se ci si mantiene nell'ambito deterministico (peraltro quasi mai verificato in campo ambientale), è possibile sempre seguire la strada cosiddetta "enumerativa": calcolare per ogni passo tutti i valori di tutte le grandezze; se ci si trova però di fronte ad un qualunque tipo di disturbo che introduca un'aleatorietà nel sistema, l'approccio enumerativo non è possibile: si rende necessario seguire la strada proposta da Bellman, quella della Programmazione Dinamica appena introdotta.
Uno solo, quindi, è il concetto alla base di questo modo di procedere: la politica ottima viene determinata ad ogni istante prendendo la decisione relativa solamente ad esso, calcolata a partire dalla conoscenza delle sole variabili di stato relative a quel preciso istante.

Seguendo la Programmazione Dinamica, si determina pertanto una politica ottima che gode dell'importantissima proprietà che *"… whatever the initial state and initial decisions are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decisions…"* (*"… Qualunque siano lo stato e la decisione iniziale presi sul sistema, le restanti decisioni devono costituire una politica ottima in riferimento allo stato risultante dalla prima decisione …"*), che costituisce il cosiddetto *Principio di Ottimalità*. [Bellman, 1957]
Le equazioni che sono state sviluppate per risolvere un problema facendo ricorso alla programmazione dinamica sono tra le più complesse mai elaborate, ma sono, nello stesso tempo, anche molto funzionali, dal momento che consentono di utilizzare un approccio completamente nuovo.
I primi studi di Bellman vennero applicati a problemi deterministici su orizzonte finito, che vennero poi rapidamente estesi ai casi stocastico, come detto estremamente più realistico, e su orizzonte infinito.

1.4.2 DERIVAZIONE FORMALE DELL'EQUAZIONE DI BELLMAN

Al fine di poter utilizzare gli algoritmi (di essi, un esempio relativo al caso più semplice, quello su orizzonte finito, viene riportato nel seguente Paragrafo 1.4.3) elaborati per risolvere problemi del tipo di quelli presentati in precedenza, occorre che siano verificate le seguenti, essenziali, ipotesi:

- Il sistema deve essere privo di disturbi deterministici (ovverosia, quando sono presenti disturbi, questi devono essere stocastici);
- Il sistema deve essere un *automa*, ovverosia gli insiemi entro cui stato, disturbi, e controlli assumono valori devono essere finiti;
- Il sistema dei vincoli cui il sistema è soggetto (rappresentato dall'insieme *D* introdotto nei paragrafi precedenti), deve essere *separabile*, ossia deve poter essere descritto mediante equazioni per ognuno dei passi temporali di analisi.

Se sussistono queste condizioni, allora è possibile formalizzare il ragionamento di Bellman in equazioni.
L'idea di fondo è quella della massimizzazione di una certa funzione dipendente dai valori assunti dallo stato: se si assume, come nel caso delle reti di serbatoi per la produzione di energia elettrica che tale funzione rappresenti un'utilità, ad esempio il beneficio derivato dalla vendita dell'elettricità prodotta, significa, in qualche modo, associare tale beneficio alla decisione presa ad ogni passo temporale, e fare in modo che esso sia il più alto possibile. Come anticipato prima, inoltre, la decisione presa ad un determinato istante influenza non solo quello immediatamente seguente, ma anche tutti quelli futuri, ed a sua volta è influenzata da tutte quelle prese in precedenza.
Occorre dunque adottare una politica mediante cui ottimizzare tutti gli stadi partendo dall'istante iniziale in poi.
Per semplificare la procedura operativa da seguire, una strada è quella di pesare il costo immediato (quello determinato dall'adozione di un preciso controllo al passo attuale) con l'utilità del valore dello stato che verrà raggiunto al passo successivo (si definiscono x_t il valore dello stato al passo attuale, x_{t+1} al passo successivo, g_t il valore del costo idroelettrico al passo attuale). L'idea da cui partire per derivare formalmente un'equazione in grado di descrivere la Programmazione Dinamica è quella di considerare il costo totale atteso che si avrebbe partendo dallo stato x_t assumendo che in tutti i passi successivi (da $t+2$ in poi) vengano prese decisioni ottime.

Se si definisce con $H^*_{t+1}(x_{t+1})$ il costo futuro ottimo (totale atteso) – *optimal cost – to go* –, la decisione ottima per il passo attuale potrebbe essere individuata rendendo minima, rispetto alla decisione, la somma del costo attuale e del costo atteso ottimo futuro.
Con queste assunzioni è possibile scrivere l'*Equazione di Bellman*, che rappresenta il fondamento della Programmazione Dinamica:

$$H^*_t(x_t) = \min_{u_t \in \mathcal{U}_t} Criterio_{\varepsilon_{t+1}}[g_t(x_t, u_t, w_t, \varepsilon_{t+1}) + H^*_{t+1}(x_{t+1})] \qquad (1.4),$$

in cui la terminologia è coerente con quella utilizzata in precedenza. Mediante l'Equazione di Bellman è possibile calcolare il costo futuro ottimo al tempo *t*, noto quello al tempo *t+1*, per cui il problema può essere risolto un passo alla volta. La seguente Figura 1.4 vuole spiegare meglio come opera l'Equazione di Bellman: come detto, il problema viene considerato un passo alla volta, "tagliando" i legami con il passato ed il futuro; il primo vincolo deve essere risolto con la conoscenza del valore attuale dello stato x_t, il secondo con la conoscenza del costo futuro ottimo $H^*_{t+1}(x_{t+1})$.

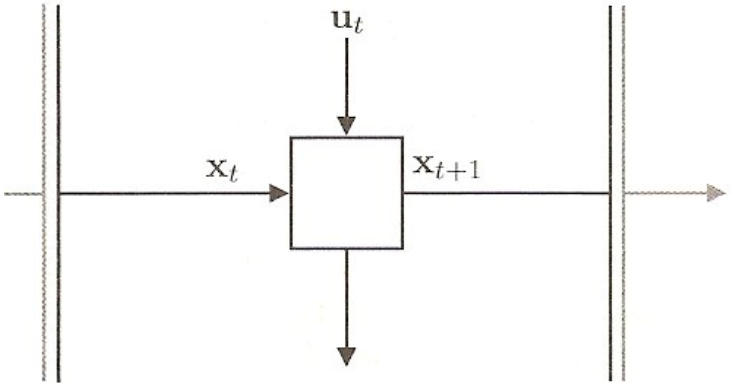

Figura 1.4 – Esempio di come opera l'Equazione di Bellman, considerando un solo passo temporale per volta (da Rodolfo Soncini – Sessa, op. cit.*)*

1.4.3 ESEMPIO DI ALGORITMO RISOLUTIVO DI BELLMAN

Si riporta nel seguito un semplice esempio dell'algoritmo sviluppato da Bellman per la risoluzione del Problema di Progetto su orizzonte finito (da *0* ad *h*) con la Programmazione Dinamica.

Passo 0: inizializzazione

Si ponga il valore costo futuro ottimo finale pari al costo istantaneo dell'istante finale, spesso corrispondente ad una penale da pagare per garantire che siano mantenute sul sistema certe condizioni. Si assegni pertanto:

$$H^*_h(x_h) = g_h(x_h), \forall x_h \in S_h \qquad (1.5),$$

dove S_h rappresenta l'insieme entro cui lo stato del sistema può assumere valori.

Passo 1

Per tutti gli istanti precedenti $t = h-1, h-2, \dots, 1$, si calcolino i costi futuri mediante l'Equazione di Bellman (equazione (1.16)), applicata ricorsivamente.

Passo 2: terminazione

A $t=0$ si calcoli mediante l'equazione (1.4) il valore del costo iniziale $H_0^*(x_0)$ per $x_0 = \overline{x_0}$. Il valore così ottenuto rappresenta l'ottimo della funzione obiettivo e le h funzioni $H_t(\circ)$ calcolate individuano la funzione di Bellman del problema.

L'algoritmo qui presentato si fonda su alcune ipotesi, qui non riportate, che sono di larghissima applicazione, tali da poter essere sempre verificate nei casi reali.
Mediante opportuni accorgimenti è possibile impostare un algoritmo analogo per risolvere problemi su orizzonte mobile ed infinito, tutti con ipotesi di validità molto lasche, sempre verificate nei casi reali. Non essendo lo scopo di questo lavoro risolvere problemi facendo ricorso alla Programmazione Dinamica, si omette in questa sede di riportare la formulazione di tali algoritmi.
Tuttavia, al fine di chiarire maggiormente come opera la Programmazione Dinamica, nel seguente Paragrafo 1.4.4 si riporta un semplice esempio applicativo dell'Equazione di Bellman.

1.4.4 ESEMPIO APPLICATIVO DELL'EQUAZIONE DI BELLMAN

Si immagini di avere un sistema come quello rappresentato nella seguente Figura 1.5, che si sviluppa su un orizzonte di progetto finito di quattro passi temporali.

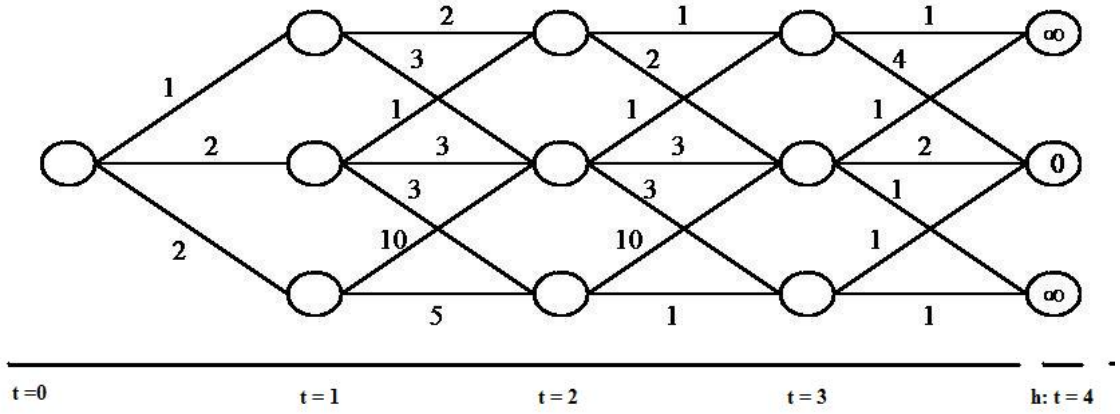

Figura 1.5 – Semplice esempio di sistema risolvibile facendo ricorso alla Programmazione Dinamica; i cerchi rappresentano i possibili stati raggiungibili ai vari istanti, mentre i valori sugli archi sono i costi associati (rielaborazione personale da Soncini, 2007)

Il sistema presentato parte da una certa situazione iniziale, rappresentata dallo stato a $t = 0$. All'istante successivo, il sistema può trovarsi in tre stati differenti, dai quali sono successivamente raggiungibili altri tre stati, secondo i collegamenti resi possibili dalla presenza degli archi; percorrere un arco in luogo di un altro ha un costo assegnato. Ciò di cui si necessita per procedere con la Programmazione Dinamica è la conoscenza dei costi finali, associati ai singoli stati che eventualmente si raggiungono. Come si vede, all'istante $t = 4$, ad ognuno dei tre possibili stati, sono associati costi differenti: in particolare al primo ed al terzo valore di stato è associato un costo infinito, mentre al secondo è associato un costo nullo.
L'algoritmo di Bellman procede a ritroso, individuando, per ognuno degli stati a $t = 3$, qual è il costo futuro ottimo atteso ($H_{t+1}(x_{t+1})$), calcolando la somma del costo attuale (determinato dalla percorrenza di uno specifico arco) con quello futuro (dato dal raggiungimento di uno specifico stato). Pertanto, se all'istante $t = 3$ ci si trova nel primo stato, il costo complessivo più basso sarà pari a 4, pari al costo di percorrenza dell'arco discendente verso lo stato all'istante successivo a costo minimo, ossia il secondo (infatti, partendo dal primo valore di stato, non è possibile, all'istante successivo, trovarsi nel terzo, dal momento che manca il

collegamento, e migrare nel primo presenta un costo infinito). La seguente Figura 1.6 mostra qual è, secondo l'algoritmo di Bellman, il comportamento del sistema dall'istante *3* all'istante *4*.

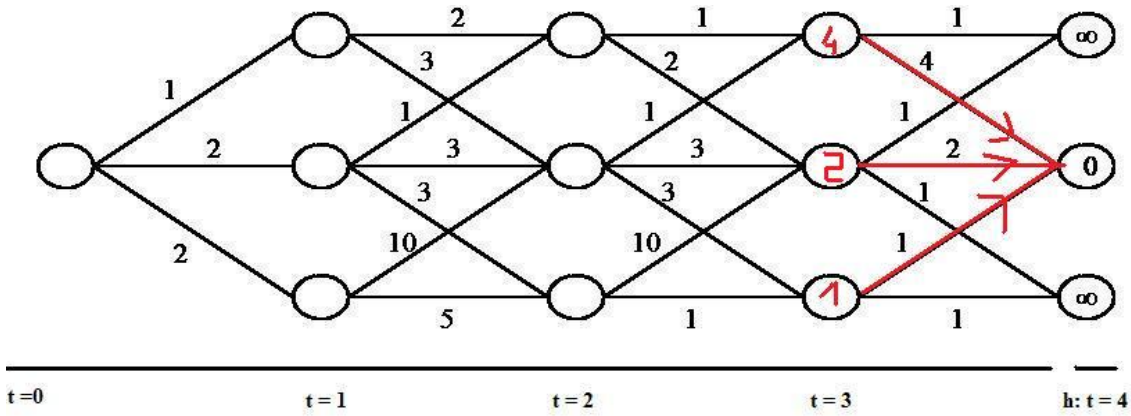

Figura 1.6 – Calcolo del costo minimo tra gli istanti t = 3 e t = 4

Il ragionamento esposto viene ripetuto a ritroso fino all'istante iniziale. I cammini che determinano il minimo costo lungo tutto l'orizzonte di progetto sono rappresentati nella seguente Figura 1.7.

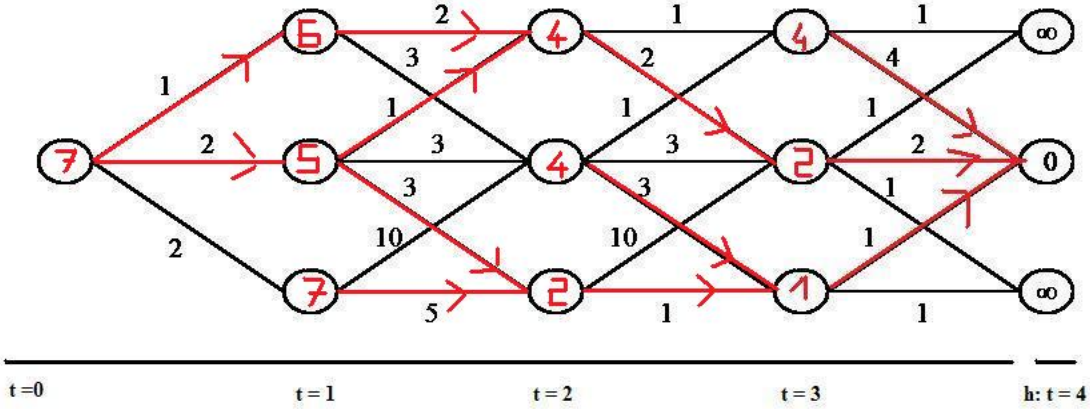

Figura 1.7 – Cammini a minimo costo di percorrenza lungo tutto l'orizzonte di progetto

L'algoritmo della Programmazione Dinamica, quindi, è in grado di determinare, per ognuno degli stati possibili ad ogni istante temporale, qual è il percorso che determina il costo complessivo più basso fino al raggiungimento dell'orizzonte.
Colui che deciderà poi come utilizzare il sistema, potrà scegliere uno qualsiasi dei cammini determinati, con la garanzia, comunque, di avere il costo più basso. Per l'esempio presentato qui, i tre possibili percorsi a minimo costo sono rappresentati nella seguente Figura 1.8.

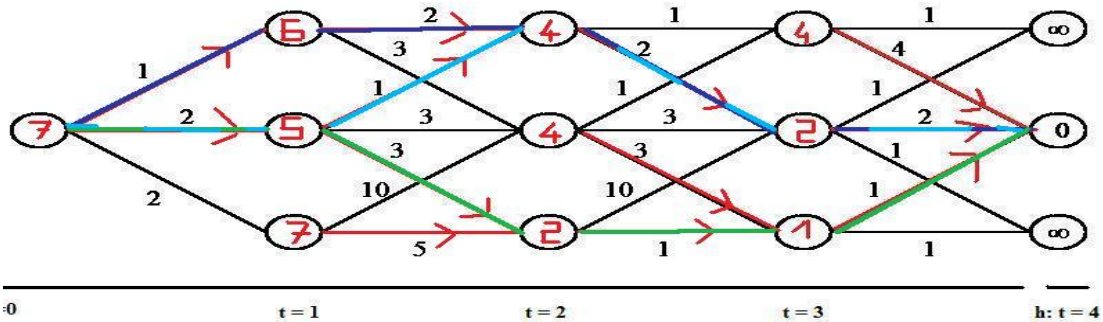

Figura 1.8 – Rappresentazione dei tre percorsi a minimo costo (blu, verde, azzurro)

1.4.5 BENEFICI E PROBLEMI CONNESSI ALLA PROGRAMMAZIONE DINAMICA

L'utilizzo nella Programmazione Dinamica per la risoluzione di problemi di ottimizzazione ha l'indubbio vantaggio di ridurre la complessità computazionale associata alla procedura di ricerca del massimo, o del minimo, di una funzione. Se si pensa, infatti, all'esempio presentato al precedente Paragrafo 1.4.4, non vengono compiuti tutti i cammini possibili alla ricerca della soluzione minima (*procedura esaustiva*), discriminando successivamente in base al costo, ma ad ogni passo ci si limita ad osservare l'immediato futuro, e si calcola il cammino minimo solo per lo specifico passo. Così facendo, al termine della procedura, si è calcolato il percorso che, dato uno specifico stato iniziale, consente di minimizzare il costo relativo ad ogni singolo passo, e quindi il costo totale.
Assumendo altresì che la procedura interessi h passi (fino all'orizzonte di progetto), che vi siano N_u controlli possibili e N_x valori di stato ammissibili, se si applica la procedura esaustiva devono essere effettuati N_u^h confronti, mentre applicando la Programmazione Dinamica la quantità si riduce ad un valore pari a

$$N_u + (N_x \times N_u) \times (h-1) \qquad (1.6).$$

Appare evidente, allora, che, al crescere della lunghezza dell'orizzonte temporale di interesse, la dimensione dei calcoli da eseguire con la procedura esaustiva cresce esponenzialmente, mentre quella della *DP* cresce linearmente. Inoltre, l'adozione della procedura esaustiva determina una crescita del tempo di calcolo necessario per la soluzione del problema esponenziale con l'aumento della dimensione dello stato e del controllo.
I vantaggi della Programmazione Dinamica apparirebbero, dunque, innegabili ed estremamente convenienti.
Inoltre, applicando l'algoritmo di Bellman, si ha la garanzia che il sistema si stabilizzi sempre sul cammino a costo minimo, anche se, ad un certo punto, interviene una perturbazione che altera il percorso prestabilito. Ovviamente, più l'orizzonte di progetto è lungo, maggiore sarà la probabilità di riportarsi sul cammino più conveniente. A suffragio di quanto esposto, si faccia riferimento ancora una volta all'esempio esposto nel Paragrafo 1.4.4, ed alla seguente Figura 1.9.

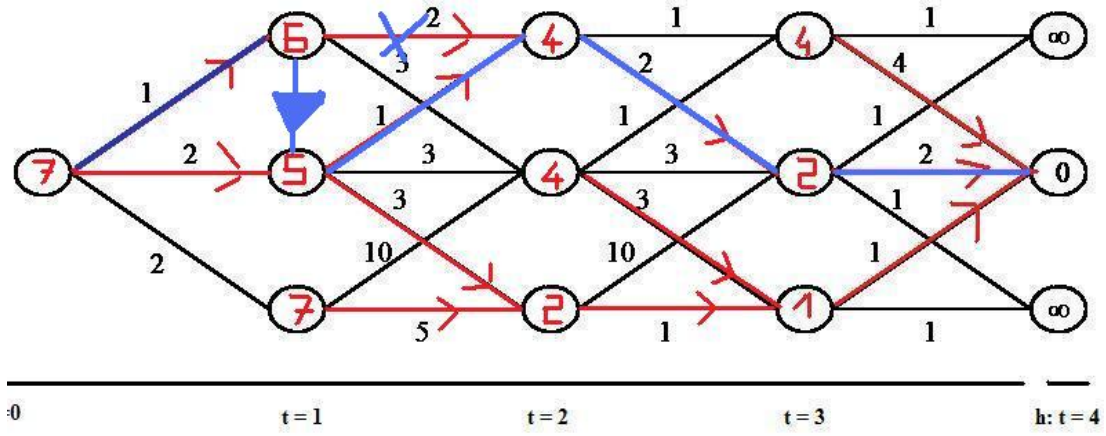

Figura 1.9 – Perturbazione del cammino ottimo e stabilità del sistema

Se si sta percorrendo il cammino migliore precedentemente indicato in blu, potrebbe verificarsi, all'istante $t = 1$, un disturbo tale per cui il sistema non si trova più nel primo valore di stato, ma "balza" al secondo, uscendo dal percorso ottimale. Tuttavia, l'algoritmo di Bellman consente al sistema di intraprendere, da quel punto, un nuovo cammino a minimo costo (rappresentato in azzurro nella Figura 1.9), che renderà i costi residui minimi; l'unica

penale da pagare è la differenza di costo tra lo stato in cui il sistema si trova a seguito della perturbazione e il costo dello stato in cui il sistema si sarebbe dovuto trovare.
La stabilità, pertanto, è uno dei più evidenti vantaggi della Programmazione Dinamica.
Tuttavia, essa ha un grosso limite, legato proprio alla dimensione degli insiemi entro cui stato e controlli assumono valori: è la cosiddetta *maledizione della dimensionalità*, per usare la terminologia dello stesso Bellman, cui si deve anche la dimostrazione dell'impossibilità applicativa degli algoritmi della *DP* quando le dimensioni dei vettori di stato e di controllo crescono troppo.
Infatti, la complessità computazionale è esponenziale con dette dimensioni, e questo rende, di fatto, impossibile determinare un valore ottimo per la soluzione, che deve essere quindi determinata con un'approssimazione, tanto maggiore quanto più grandi sono stato e controlli.
Se si definiscono la dimensione dello stato pari a n_x, quella dei controlli pari a n_u, quella dei disturbi pari a n_ε, occorre definire anche gli insiemi in cui stato, controlli e disturbi assumono valori, chiamati rispettivamente S_x, S_u e S_ε. Se l'insieme di stato contiene ad ogni istante x valori, l'insieme S_x conterrà x^{n_x} valori, ciascuno dei quali da minimizzare rispetto a tutti i possibili controlli, dopo aver filtrato tutti i disturbi che si possono manifestare: si evince come gli algoritmi siano sostanzialmente inapplicabili a livello computazionale, a meno che le dimensioni di stato, controllo e disturbi non siano molto piccole.
Inoltre, a meno che non siano di forma estremamente semplice, le soluzioni del Problema di Progetto non possono essere determinate utilizzando tecniche di Programmazione Lineare, dal momento che esse difficilmente si adattano a problemi dinamici.
La Programmazione Dinamica si rivela dunque di sovente inadatta per risolvere complessi problemi di ottimizzazione, e deve essere abbandonata, oppure modificata: si deve, in sintesi, fare ricorso ad altri metodi.
Una soluzione più sofisticata, che consente comunque di rimanere all'interno della Programmazione Dinamica, è il cosiddetto *metodo lessicografico* ovvero *metodo ε-constraints*, mediante il quale si assegna un valore di importanza agli obiettivi, risolvendo all'interno del Problema il primo, ed esprimendo gli altri come equazioni vincolari. [Soncini Sessa, 2004] [Giuliano *et al.*]

1.5 POLITICHE DI CLASSE FISSATA

La Programmazione Dinamica descritta finora utilizza quello che è stato chiamato *approccio funzionale*, ossia determina la politica ottima all'interno di tutte le politiche ammissibili. Come si è visto, però, essa non è applicabile a meno che le dimensioni delle variabili del problema non siano limitate. È possibile, in alternativa, fare ricorso ad un *approccio parametrico* che si limita, più semplicemente, a trovare i parametri che ottimizzano una politica la cui forma è assegnata a priori; indirettamente, un'altra strada consiste nel definire a monte della risoluzione la forma delle funzioni di costo futuro, e determinare da queste la politica.
Scegliere di utilizzare l'approccio parametrico consente di superare le ipotesi restrittive introdotte all'inizio del Paragrafo 1.4.2, dal momento che è possibile considerare un sistema che non è un automa, eventualmente anche soggetto a disturbi deterministici; facendo ricorso invece alla seconda alternativa presentata, ossia la scelta di funzioni di costo di classe fissata, non si possono rilassare i vincoli cui anche la Programmazione Dinamica deve soggiacere, ma è possibile allargare considerevolmente le maglie della griglia di discretizzazione.
La scelta di fissare a priori la classe delle funzioni di ottimizzazione verrà brevemente esposta nel seguito, mentre per quanto riguarda l'idea di fissare anticipatamente la classe delle funzioni di costo, essa verrà trattata nel Capitolo 2, giacché a questa tipologia di funzioni appartengono le Reti Neurali, su cui l'intera architettura di questo lavoro di Tesi si basa.

1.5.1 FORMA FUNZIONALE DELLE POLITICHE FISSATA A PRIORI

Si è detto che una politica è definita come una sequenza, finita o infinita, di leggi di controllo con una certa, specifica, forma funzionale. Se si conoscesse tale forma, del tipo

$$m_t^*(\circ) = \widetilde{m_t}(\circ\,|\theta_t) \qquad (1.7),$$

sarebbe possibile definire la politica ottima in modo univoco, una volta che sia stato determinato il vettore dei parametri θ che servono per specificare la forma funzionale della legge di controllo. Occorre perciò determinare il vettore

$$\theta^* = |\theta_0^*, \theta_1^*, \dots, \theta_h^*| \qquad (1.8),$$

che definisce per ognuno dei passi *h* di simulazione tutti i parametri da introdurre nella generica forma funzionale espressa dall'equazione (1.7) per specificarla.
Usualmente, per la stima del vettore dei parametri espresso dall'equazione (1.8) si possono utilizzare o il metodo dei minimi quadrati, oppure una formulazione simile all'Equazione di Bellman introdotta nel paragrafo 1.4.2 con l'equazione (1.4):

$$\hat{\theta} = \min_\theta Criterio_\varepsilon[\sum_{t=0}^{h-1} g_t(x_t, \widetilde{m_t}(x_t, \theta_t), w_t, \varepsilon_{t+1}) + g_h(x_h)] \qquad (1.9),$$

soggetta agli stessi vincoli con cui si scrive il Problema di Progetto, tra i quali si deve aggiungere la dipendenza della legge di controllo dal vettore dei parametri, e la definizione dello stesso vettore dei parametri:

$$\begin{cases} \widetilde{m_t}(x_t, \theta_t) \in \mathcal{U}_t(x_t) \\ \theta = |\theta_0, \theta_1, \dots, \theta_{h-1}| \end{cases} \qquad (1.10).$$

Completato con i vincoli espressi dalla precedente equazione, quello descritto è un Problema di Programmazione Matematica, assai più semplice, dal momento che l'operazione di ricerca del vettore dei parametri è equivalente ad un'operazione di pianificazione u^p, che deve essere compiuta una sola volta, all'atto della definizione della forma della politica.
Il principale problema nell'applicazione di questa metodologia riguarda la conoscenza a priori della forma funzionale delle leggi di controllo, giacché esse non sono note, perché devono essere calcolate. Se la classe funzionale non fosse corretta, si determinerebbe ugualmente una politica ottima (rispetto, ovviamente, alla classe funzionale individuata), ma essa si rivelerebbe del tutto inadatta, eventualmente pessima, con conseguenze tragiche sul sistema. Ciò che invalida – parzialmente almeno – l'applicazione di questo metodo, è il fatto che non esiste un metodo per determinare a priori una buona classe funzionale. Si lascia all'intuizione dell'Analista, il quale può, eventualmente, valutare a posteriori le prestazioni della politica determinata, sempre in modo soggettivo, perché non si conosce il comportamento della vera politica ottima.

1.5.2 CLASSE FISSATA DEI COSTI FUTURI

Dato che in questa categoria rientrano le Reti Neurali, si fornisce qui una solo una breve descrizione introduttiva, rimandando al successivo Capitolo 2 la descrizione dettagliata di queste strutture modellistiche.
L'idea che si pone come fondamento della struttura delle Reti Neurali riguarda il fatto che per identificare la funzione di Bellman per uno specifico problema occorre memorizzare un numero di controlli così elevato, da essere superiore alle reali possibilità di memoria dei più moderni calcolatori. Per ovviare a tale, importante, problema, si può immaginare che i controlli calcolati non siano totalmente liberi gli uni dagli altri, ma esistano tra di essi delle relazioni, che, almeno in parte, devono essere conosciute a priori. Pertanto, la funzione di Bellman si può approssimare con un'altra funzione, definita con $\tilde{H}(\circ, \theta)$, in cui il vettore dei parametri θ è costituito da elementi θ_t per ognuno dei passi temporali di interesse. Ognuna delle funzioni $\tilde{H}(\circ, \theta_t)$ così ottenute si chiama *funzione di punteggio* (*scoring function*), ha una struttura tempo invariante fissata a priori, mentre il vettore dei parametri relativo all'istante *t* individua la dipendenza dal tempo della funzione stessa. Al fine di semplificare i calcoli, è bene che le funzioni di punteggio siano determinate da vettori parametrici a bassa dimensionalità (*architetture compatte*), in modo tale che anche i risultati finali siano a bassa dimensionalità. Descritta l'opportuna architettura, occorre assegnare dei valori ai parametri, approssimando al meglio il valore della funzione punteggio con quello della funzione di Bellman:

$$\tilde{H}(\circ, \theta_t) \approx H_t^*(\circ) \qquad (1.11).$$

La procedura consta, pertanto, di due distinte fasi, sinteticamente riassumibili:

1) Scelta dell'architettura compatta;
2) Stima del vettore dei parametri che meglio approssima l'equazione di Bellman.

Di solito, l'architettura compatta che si sceglie è proprio quella delle Reti Neurali, mentre la seconda fase si effettua con il cosiddetto *addestramento della rete* precedentemente individuata.

2

LE RETI NEURALI: FORMA E FUNZIONAMENTO

Si descrive la classe di modelli delle Reti Neurali, che consentono di agire con meno vincoli, vista la capacità di apprendere propria di questi modelli

2.1 INTRODUZIONE

Esiste una tipologia di problemi che meno facilmente può essere risolta in termini matematici, come il riconoscimento di suoni o immagini, perché legato all'associazione di concetti e memorie che difficilmente la matematica può riprodurre con equazioni, oppure perché troppo complessi per trovare una soluzione con l'applicazione di algoritmi in tempi e costi ragionevoli. Questo, nello specifico, è quanto esposto nel precedente Capitolo 1, in cui si è dimostrato come le problematiche legate ai sistemi ambientali non possono essere trattati con le usuali tecniche analitiche della Programmazione Dinamica, ed occorre rivolgersi ad altre tipologie di modelli.
In sostanza, i sistemi artificiali su cui si basa la Programmazione Dinamica sono algoritmi molto potenti per ripetere numerose operazioni in sequenza predeterminata, ma non sono sistemi "intelligenti", perché non possono in alcun modo apprendere quanto elaborato in precedenza. [Gallo, 2007]
Invece, nei sistemi artificiali intelligenti rientrano i modelli che tentano di riprodurre la struttura, le dinamiche ed il funzionamento del cervello umano, le cui connessioni riescono, facilmente e rapidamente, a riconoscere ad esempio un suono o un'immagine e ad associarli a specifici eventi passati.
Le Reti Neurali rientrano in questa tipologia di modelli: essi sono strumenti molto forti, che, lavorando in parallelo, possono trattare contemporaneamente una buona quantità di dati con una discreta insensibilità al rumore, potendo mantenere buone prestazioni anche se una parte delle unità che compongono il sistema dovessero funzionare male o non funzionare affatto.
Di contro, non è possibile conoscere come le Reti Neurali implementino dei modelli al loro interno: occorre accettarne i risultati "a scatola chiusa", sapendo che comunque l'efficacia di questa tipologia di modelli (una cui struttura ottima in termini assoluti, comunque, non è identificabile) dipende da come è stata effettuata la procedura di taratura e di addestramento.

2.1.1 CENNI STORICI

La storia delle Reti Neurali ha le sue origini negli anni della Seconda Guerra Mondiale, quando, nel 1943, W. S. McCulloch e W. Pitts, descrissero il neurone artificiale e le sue connessioni con altre entità simili, andando a replicare, in modo artificiale, la struttura del cervello umano. Solo nel 1949, tuttavia, si inizia a parlare delle possibilità di apprendimento, grazie al lavoro di Hebb, che fu però, insieme al precedente, confutato da Von Neumann nel 1958, per la scarsa precisione e la chiarezza espositiva.
Tra il 1957 ed il 1958, comunque, Rosenblatt propone il primo vero schema "moderno" di rete neurale, ossia il *percettrone*, in grado di riconoscere forme ed associare configurazioni.
Il percettrone (descritto nel Paragrafo 2.3.2), supera le limitazioni della struttura binaria di McCulloch e Pitts, perché dotato di pesi sinaptici variabili, quindi in grado di apprendere.
Fino agli Anni '70 – '80, ossia fino all'avvento dei moderni calcolatori, le reti neurali cadono nel disinteresse generale della comunità scientifica, che vi riprende contatto con il lavoro di Werbos del *1974*, in cui si descrivono le basi matematiche per l'addestramento di reti neurali multi – strato, e la definizione delle *memorie auto – associative*.
La più moderna struttura delle reti neurali si raggiunge nel *1986*, quando Rumelhart, Hilton e Williams descrivono l'algoritmo di addestramento per *retropropagazione dell'errore*, grazie al quale è possibile modificare i pesi delle connessioni neuronali in modo sistematico, finché la risposta della rete non diventa uguale – o si avvicina il più possibile – a quella desiderata. L'algoritmo di *backpropagation* sarà oggetto del Paragrafo 2.4.3.
Oggi le reti neurali più moderne si presentano come strutture adattive capaci di apprendere e di emulare, operando prima in parallelo, poi modificando la propria struttura interna in modo da correggere, ove possibile eliminando, gli errori, con un approccio nell'affrontare i problemi completamente differente, anche rispetto alla Intelligenza artificiale classica.

2.2 STRUTTURA DELLE RETI NEURALI

2.2.1 CENNI STRUTTURALI DEL SISTEMA NERVOSO UMANO

Di tutti i sistemi nervosi presenti negli organismi pluricellulari, quello umano è il più complesso e rappresenta, ad oggi, il gradino più alto raggiunto dal processo evolutivo. Il cervello umano è un organo composto da entità cellulari chiamate neuroni, presenti in numero variabile tra 10^{10} e 10^{12}. [Macchiavello, 1992] Ogni cellula è connessa alle altre in modi differenti, fino a raggiungere 10^{4} legami per ogni cellula; tali connessioni possono variare in quantità e tipologia per rispondere ai diversi stimoli che provengono dall'esterno.
Ogni singolo neurone è composto da un corpo cellulare principale, da un assone, che invia segnali alle cellule circostanti e da numerosi dendriti, che si ramificano verso gli altri neuroni, ricevendo dei segnali recepiti dagli assoni, come rappresentato dalla seguente Figura 2.1.

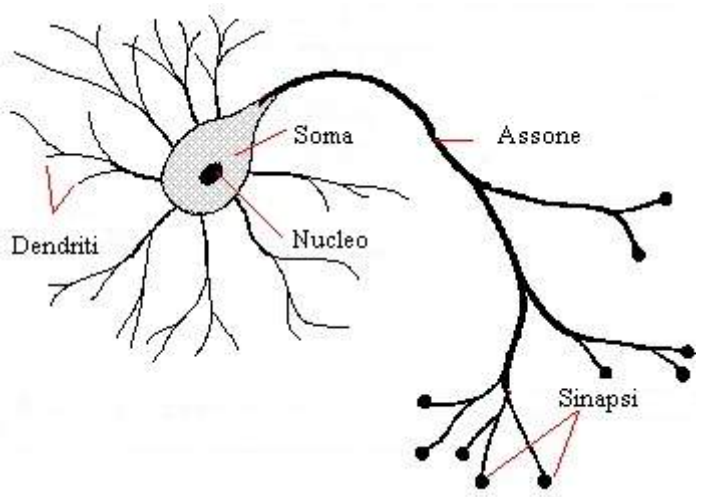

Figura 2.1 – Rappresentazione di un neurone del sistema nervoso; si notano il soma cellulare, i dendriti e l'assone

La membrana della cellula è sempre elettricamente carica, dal momento che riceve un'eccitazione dai dendriti. Superata una certa soglia di carica, il segnale viene inviato tramite l'assone agli altri neuroni, attraverso la regione di contatto che viene chiamata *sinapsi*, ove fisicamente si realizza la trasmissione elettrica.
Proprio le sinapsi tra i neuroni sono responsabili dell'apprendimento, dal momento che la forza della connessione neuronale è funzione dello spessore delle sinapsi: l'apprendimento varia con il variare dell'efficienza di connessione.
Sebbene il sistema nervoso umano appaia come estremamente complesso ed efficiente, esso non è, tuttavia, particolarmente performante, dal momento che la capacità di elaborare dati dei neuroni è notevolmente più bassa di quella dei componenti dei calcolatori (circa *1* millisecondo per il sistema nervoso, *10* nanosecondi per un computer [Macchiavello, 1992]), e, per tale motivo, l'informazione viene trasmessa, all'interno del sistema nervoso, in modo parallelo piuttosto che seriale.
A complemento di questa essenziale descrizione della struttura e del funzionamento delle reti neurali biologiche, si ricorda come, mentre nei calcolatori le unità adibite al calcolo siano distinte da quelle adibite alla memorizzazione dei dati, così non è nel cervello umano, dove non sono nettamente distinguibili le aree destinate alla memorizzazione da quelle destinate all'elaborazione.

2.2.2 CARATTERISTICHE BASILARI DI UNA RETE NEURALE ARTIFICIALE

Prima ancora di definire la struttura ed il funzionamento delle reti neurali artificiali, è bene porre l'attenzione su quelle che devono essere le caratteristiche essenziali che un tale modello deve possedere, in riferimento a quanto descritto per il sistema nervoso umano:

- Deve essere presente un numero molto elevato di componenti, ciascuno in grado di eseguire un numero limitato di operazioni semplici, quali medie pesate e capacità di attivazione (inibizione), sopra (sotto) una certa soglia;
- Una buona rete neurale artificiale deve disporre di un numero di connessioni tra gli elementi, al fine di riprodurre le sinapsi tra assoni e dendriti;
- Deve sussistere la possibilità di modificare l'intensità delle connessioni mediante un opportuno processo di apprendimento.

Proprio la fase di apprendimento riveste un'importanza fondamentale, dal momento che i pesi tra le connessioni (equivalenti delle sinapsi biologiche) neuronali non possono essere

determinati a priori in funzione del compito che la rete è chiamata a svolgere, ma devono essere appresi durante il funzionamento del modello: a seconda delle prestazioni ottenute, l'apprendimento modifica, indebolendo o rinforzando, le connessioni, oppure ne elimina alcune o ne crea delle altre.
Ciò premesso, la versatilità delle reti neurali consiste nella possibilità di determinarne contestualmente sia l'architettura sia il numero ed il valore dei parametri (mentre, ad esempio, nei modelli empirici occorre prima fissare la forma e poi tarare i parametri), ma questo amplia di molto il tempo di computazione e la procedura di addestramento.

2.2.3 DEFINIZIONE E STRUTTURA DELLE RETI NEURALI ARTIFICIALI

Prima di entrare nei dettagli delle differenti strutture esistenti per le reti neurali, è bene darne una definizione, semplice e compatta, che spieghi dal punto di vista matematico che cosa è e come opera: una Rete Neurale è un regressore non lineare che esprime una relazione funzionale esistente tra un vettore (detto anche *unità*, così come quelli di uscita) di input e una o più variabili di output [Soncini Sessa, 2007]. Ogni variabile di ingresso è collegata da una serie di neuroni, ciascuno dei quali svolge una semplice operazione, ossia diventa attivo se il segnale in ingresso è maggiore della soglia di attivazione determinata dalla *funzione di attivazione* (o *activation function*), trasforma la combinazione – solitamente lineare – degli ingressi in una sola uscita, mediante una *funzione di trasferimento*.
Una Rete Neurale non è un modello scritto in un particolare linguaggio di programmazione, ma è una macchina molto potente, dal momento che può essere istruita a compiere operazioni per imitazione, in maniera tale da disporre di esempi mediante i quali imparare.
Le reti, oltre all'essenziale strato di ingresso, devono possedere uno o più strati di elaborazione: in generale, vi sono L layer o strati, di cui i primi $L - 1$ nascosti, contenenti n neuroni ciascuno, mentre l'ultimo, visibile, è lo strato di output e contiene un numero q di neuroni pari alle variabili di uscita. Un esempio di generica rete multi – strato è riportata nella seguente Figura 2.2.

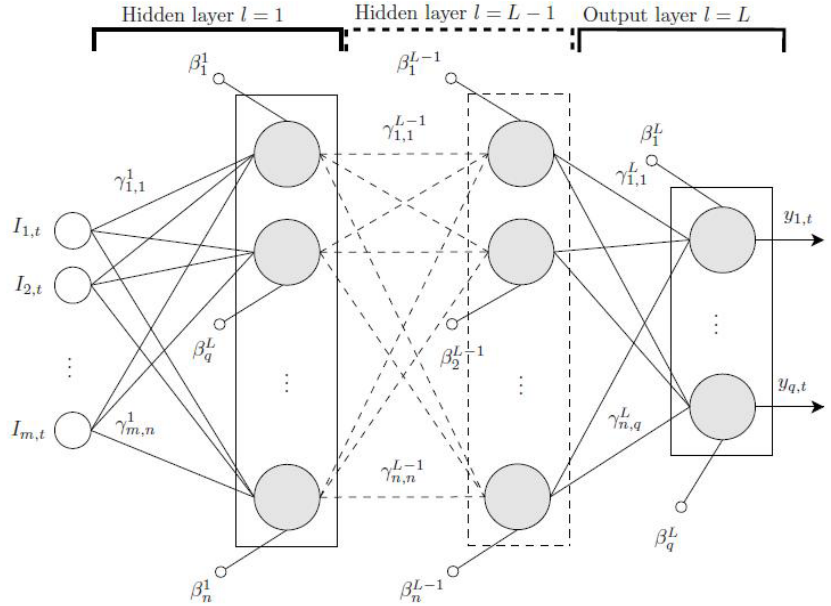

Figura 2.2 – Esempio di rete neurale artificiale multistrato, dove compaiono gli input I_m, i neuroni interni caratterizzati da peso e rumore, e le variabili di uscita y_q (da Soncini Sessa, op. cit.*)*

Il vettore di input (I) è costituito da m componenti che "nutrono" (dall'inglese *feed*) il primo strato di neuroni, quello appunto chiamato di input. Associati alle connessioni tra l'i-esimo neurone dello strato $l - 1$ con il j-esimo dello strato l si definiscono dei pesi (γ). Ogni neurone

presenta inoltre un valore costante (*activation treshold* o *bias*, definito da β), che consente di definire le traslazioni positive o negative della funzione di attivazione rispetto all'origine degli assi.
Per quanto potenzialmente di struttura complicata, le reti neurali sono quasi sempre di un'unica tipologia, ovverossia *feed – forward*, nelle quali l'informazione si muove in un'unica direzione, dagli input verso gli output passando per gli eventuali strati nascosti, senza percorrere cicli o anelli chiusi. Le reti così costituite sono molto versatili, tanto da essere utilizzate nella quasi totalità delle applicazioni attuali, perché si adattano bene a qualsiasi tipo di rappresentazione, consentendo, dove sia necessario modellizzare dei cicli o delle ricorsività, di introdurre il concetto di *reti neurali ricorrenti*.

2.2.4 TIPOLOGIE DI APPRENDIMENTO

Come anticipato, una delle fasi essenziali durante la costruzione di una rete neurale è l'apprendimento, in cui il modello impara a collegare gli ingressi alle uscite, sostanzialmente apprende la forma della funzione di trasferimento. Esistono differenti tipologie di apprendimento, in funzione anche della complessità e della struttura che si vuole dare alla rete:

1) *Apprendimento supervisionato* (*supervised learning*), che avviene nella quasi totalità dei casi, quando si hanno a disposizione numerosi dati, che rappresentano un'ampia casistica degli ingressi possibili, cui siano associate le relative uscite (*coppie input – output*). La rete impara a determinare la forma della funzione che lega ingressi e uscite note, modificando i propri parametri ed i pesi delle connessioni, tramite un algoritmo di minimizzazione dell'errore di stima delle uscite (di solito la retropropagazione dell'errore). Affinché possa essere utilizzata questa tipologia di addestramento, occorre che la rete abbia una buona capacità di generalizzazione, ovvero deve fornire una buona prestazione relativamente a set di dati non utilizzati per la calibrazione [Maier *et al.*, 2000]: sebbene possa apparire paradossale, una rete generalizza tanto di più, quanto minore è il numero di parametri che la definiscono, dal momento che un'architettura semplice si rivela più generale;
2) *Apprendimento non superivisionato* (*unsupervised learning*), nel quale l'addestramento si effettua unicamente facendo riferimento ai dati di input. Gli algoritmi di apprendimento non supervisionato cercano di raggruppare i dati in cluster che ne siano rappresentativi, tramite metodi prevalentemente probabilistici;
3) *Apprendimento per rinforzo* (*reinforcement learning*), che differisce dall'apprendimento supervisionato perché non si basa sul confronto di coppie di input – output, ma semplicemente consente alla rete di memorizzare la prestazione ottenuta introducendo un determinato ingresso ed assegnando ai parametri uno specifico valore: se tale prestazione è buona, la rete se ne servirà nuovamente per migliorare le proprie prestazioni, altrimenti la dimenticherà e non se ne servirà più. Ogni azione della rete determina un effetto sull'ambiente, il quale genera una risposta che la rete utilizza per valutare la propria prestazioni.

2.3 STRUTTURE NEURALI ARTIFICIALI

2.3.1 COMBINATORE LINEARE A SOGLIA

Il combinatore lineare a soglia è il primo esempio di semplice neurone, proposto da McCulloch e Pitts nel 1943. Esso si attiva quando raggiunge un livello di eccitazione sufficiente, determinato da una soglia di attivazione. La forma della funzione di attivazione *A* è data da:

$$A_i = \sum_{j=1}^{n} W_{ij} X_j \qquad (2.1),$$

mediante la quale ci si riferisce all'attivazione del neurone *i*, determinata dalla combinazione lineare dei pesi *W* delle connessioni tra il neurone *i* e gli altri *j*, e il valore dell'uscita dello strato precedente, *j*.
Il valore di soglia che deve essere superato dalla funzione di attivazione è determinato, in questa prima struttura neurale, da una funzione di trasferimento (ossia una rappresentazione matematica che collega gli ingressi ad un sistema e la risposta di quest'ultimo) a gradino di Heaviside, definita semplicemente come

$$\Theta(x) = \begin{cases} 0 \text{ per } x < 0 \\ 1 \text{ per } x > 0 \end{cases} \qquad (2.2)$$

Al fine di non creare problemi nella definizione della funzione di Heaviside nell'origine *0*, si è scelto di porre

$$\Theta(x = 0) = \frac{1}{2} \qquad (2.3)$$

in modo tale da poterla definire tramite la funzione segno (*sgn*):

$$\Theta(x) = \frac{1}{2}\big(1 + \operatorname{sgn}(x)\big) \qquad (2.4).$$

Questa funzione si utilizza per determinare il valore delle uscite di ognuno degli strati che compongono la rete:

$$X_j = \Theta(A_i - \theta_i) \qquad (2.5),$$

Dove A_i rappresenta la matrice degli ingressi, e θi il vettore dei bias.

2.3.2 PERCETTRONE SINGOLO E MULTI - STRATO

Il combinatore a soglia è stato migliorato con la costruzione del percettrone, il cui funzionamento e struttura furono individuati da Rosenblatt nel 1957, che si differenzia dalla struttura precedente quasi unicamente per la possibilità di aggiornare i pesi e di modificare il valore di soglia durante la consistente fase di addestramento, al fine di elaborare tutti i dati in ingresso allo strato per produrre un unico valore di uscita, pur rimanendo un semplice classificatore lineare binario.
La versatilità dei percettroni si manifesta soprattutto nella semplicità dell'algoritmo di implementazione, e nella generalità del loro utilizzo.

Il percettrone rappresenta la struttura più semplice di rete *feedforward* a singolo strato, che usa come funzione di attivazione la funzione sigmoide di tipo logistico piuttosto che il gradino di Heaviside. La funzione sigmoide si definisce come segue:

$$P(t) = \frac{1}{1+\exp(-t)} \quad (2.6),$$

una cui rappresentazione è riportata nella seguente Figura 2.3.

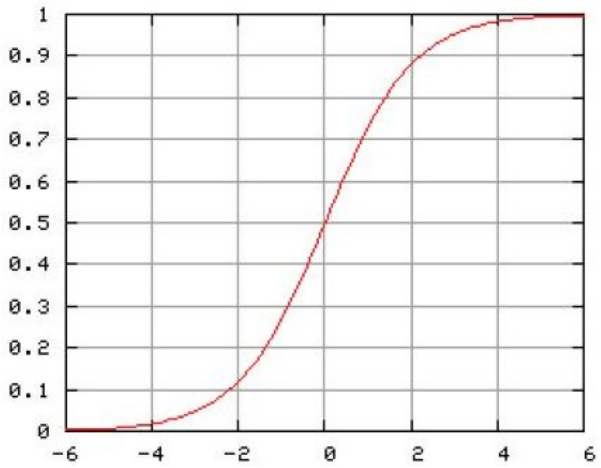

Figura 2.3 – Funzione di attivazione di tipo sigmoide

In ogni strato, l'input *I* ad un qualsiasi neurone *j* è dato dalla somma pesata (W_{ji} sono i pesi relativi) degli output dello strato precedente, *i*:

$$I_j = \sum_i W_{ji} x_i \quad (2.7),$$

mentre l'uscita del nodo *j* è ottenuta mediante la funzione di attivazione individuata con la precedente Equazione (2.6):

$$x_j = P(I_j) \quad (2.8).$$

La funzione di attivazione, descritta genericamente con l'Equazione (2.6), può assumere differenti forme, tra cui quella che segue, una delle più comuni [Hao, 1989], [Soncini Sessa, 2007]:

$$P(I_j) = \frac{1}{1+\exp\left(-\frac{I_j+\theta_j}{\theta_0}\right)} \quad (2.9),$$

dove θ_j equivale ad un *bias*, ossia una soglia da superare per l'attivazione del neurone, e corrisponde graficamente ad una traslazione lungo l'asse delle ascisse della funzione di attivazione, mentre θ_0 serve a modificare la forma della curva. Il valore assunto dal parametro θ_0 agisce sulla forma della funzione sigmoidale: quanto più tale parametro assume valori bassi, tanto più la sigmoide si avvicina ad una funzione di Heaviside, come mostrato nella seguente Figura 2.4.

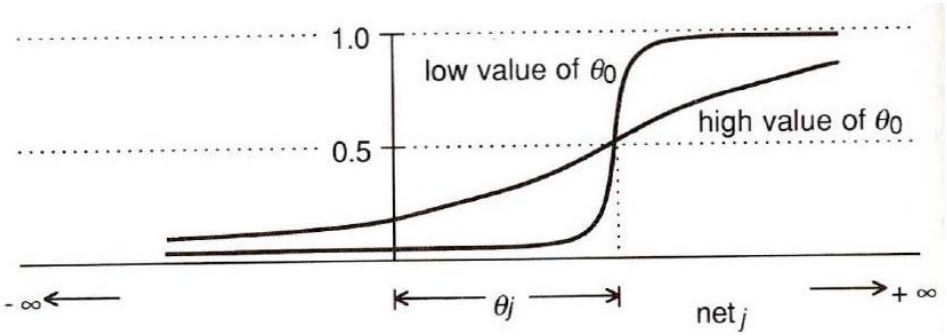

Figura 2.4 – Variazione della forma della curva sigmoidale al variare del parametro θ_0 (da Pao, op. cit.*)*

Possono esistere delle reti neurali con più strati nascosti, ed in questo caso prendono il nome di *multi – layer perceptron – MLP*.
Sebbene, grazie al Teorema dell'Approssimazione Universale, sia possibile ricondurre alla forma *MLP* quasi tutte le reti neurali, si è visto che, per le applicazioni pratiche, si ottengono dei risultati molto buoni quando è presente un solo strato nascosto.

2.4 CENNI ALLA CLASSIFICAZIONE ED ALLE TIPOLOGIE DI RETI NEURALI

I modelli di Reti Neurali si possono classificare in base ad alcune caratteristiche fondamentali, come la tipologia di utilizzo, la tipologia e quindi il tipo di apprendimento che sottosta al loro funzionamento, l'architettura con cui sono costruite le sinapsi neuronali.
La classificazione normalmente utilizzata fa riferimento all'utilizzo delle reti, i cui compiti sono molteplici:

- *Memorie associative*, in cui si cerca di costruire relazioni causa – effetto simili tra loro, che possano essere ricostruite anche quando alcuni input sono imprecisi o incompleti. È il caso delle reti prive di capacità di generalizzazione;
- *Simulatori di funzioni matematiche*, in grado di comprendere la funzione di trasferimento input – output basandosi su pochi esempi forniti in fase di apprendimento. Sono le reti maggiormente diffuse, e si avvalgono dell'algoritmo di retropropagazione dell'errore;
- *Classificatori*, che si utilizzano per catalogare dati sulla base di alcune loro caratteristiche comuni.

Oltre alla forma più utilizzata, quella *feed – forward*, esistono anche molte altre tipologie di reti neurali:

- *Radial Basis Function* (*RBN*), il cui funzionamento si basa sul concetto di distanza da un punto centrale: maggiore è tale distanza, minore è l'effetto degli input sul sistema;
- *Kohonen self – organizing network*, che, basandosi su un apprendimento non supervisionato, costruiscono autonomamente le connessioni tra i neuroni;
- *Recurrent Neural Network* (*RNN*), nelle quali le connessioni tra i neuroni possono essere cicliche, consentendo di studiare un comportamento dinamico degli elementi stessi nel tempo;
- *Simple ricurrent* (*SRNN*), sostanzialmente una variazione del percettrone multi – strato;
- *Stochastic*, nelle quali i parametri possono assumere valori stocastici, da individuare con distribuzioni probabilistiche;

- *Modular Neural Network*, costituite da numerosi elementi semplici, che operano insieme per risolvere problemi più complessi.

2.5 ADDESTRAMENTO DELLA RETE NEURALE

2.5.1 DESCRIZIONE GENERALE DELLA PROCEDURA

Una Rete Neurale è una struttura generale, che si deve poter adattare a molti casi differenti. Mediante una procedura di *addestramento* (*training*), la rete viene portata ad assumere l'assetto atto a risolvere il problema specifico che si sta analizzando. In modo molto semplice, la procedura di addestramento avviene tramite la modifica dei pesi associati alle connessioni tra i singoli neuroni, pesi che inizialmente sono assegnati in modo casuale e che successivamente evolvono fino ad una situazione di equilibrio, raggiunto quando la rete ha appreso un concetto oppure è in grado di risolvere, in generale, il problema (verosimilmente ogni componente della rete risolve un'operazione elementare, l'insieme delle quali porta alla soluzione completa).
La procedura di addestramento non è mai semplice. Può capitare, infatti, che essa non termini, se l'architettura della rete oppure l'algoritmo utilizzato per l'addestramento non sono corretti. Al contrario, se l'addestramento va a buon fine, si ottengono dei pesi e dei bias che consentono alla rete neurale di funzionare correttamente fornendo in input qualsiasi ingresso.
Per modificare i pesi sinaptici possono essere usati molti algoritmi diversi (i maggiori vengono descritti nei paragrafi seguenti), i principali dei quali sono di tipo retroattivo, nel senso che il risultato ottenuto al termine di un passo viene utilizzato alla successiva iterazione e si basano, sostanzialmente, sul confronto tra i risultati ottenuti dalla rete e quelli desiderati, e sulla minimizzazione dello scarto tra questi.
A livello procedurale, occorre anzitutto disporre di una serie di *m* vettori di esempio da fornire alla rete come input (*vettori di esempio* o *training set*), uno per volta, in modo da poter calcolare, per ognuno di essi, l'errore ed aggiornare i pesi prima di passare alla lettura del successivo. Terminata la lettura e la modifica dei pesi per tutti i vettori di esempio, si ripete ricorsivamente la procedura fino a quando non si ottenga un'approssimazione soddisfacente.
Come detto, esistono diversi algoritmi di addestramento: il principale prende il nome di *algoritmo a retropropagazione dell'errore* (*error back propagation*), che sarà descritto nel Paragrafo 2.5.3, dopo aver introdotto l'altra importante tipologia di algoritmo, nel Paragrafo 2.5.2.
Dal momento che la rete deve sapere rispondere correttamente a qualsiasi ingresso (proprietà di *generalità della rete* neurale), non è possibile che i soli responsi corretti siano riferiti ai soli vettori del training – set, perché altrimenti si avrebbe una semplice *memoria associativa*, con capacità predittiva nulla, o molto bassa.
A tale proposito, la capacità risolutiva di una rete è tanto più elevata quanto più le connessioni sono presenti in numero elevato, mentre la capacità di generalizzazione si riduce (fenomeno di *overfitting* [Cammarata, 1990]).
Pertanto, oltre ad una buona procedura di addestramento, occorre saper scegliere un'architettura adatta per la rete, in cui il numero di neuroni, connessioni ed ingressi siano i più piccoli possibili: usualmente è sufficiente un solo strato nascosto con funzioni di attivazione sigmoidali. [Maier *et al.*, 2000]
Più la rete è semplice, più veloce è la procedura di ottimizzazione, anche se peggiore può essere la prestazione: occorre trovare un buon compromesso tra le due esigenze, che non può essere lasciato unicamente all'intuizione dell'Analista, motivo per cui sono stati proposti differenti algoritmi (qui solamente citati, se ne trovano ampi riferimenti in [Maier *et al.*, 2000]) che, partendo da una determinata architettura neurale, o la semplificano il più possibile

(*pruning algorihtms*), oppure la espandono, cercando di mantenerne la capacità di generalizzazione e le prestazioni (*constructive algorithms*).

2.5.2 APPRENDIMENTO *FORWARD PROPAGATION*

Questo algoritmo di apprendimento non è retroattivo, dal momento che i pesi vengono assegnati, e fissati, percorrendo la rete dallo strato di ingresso verso quello di uscita, senza tornare indietro. I valori di input alla funzione di attivazione *I* per ogni neurone *j* di uno strato *s*, con $s \neq 1, \{s = 2,3, \dots, n\}$, si calcolano come

$$I_j^s = \sum_i W_{ij} x_i^{s-1} \qquad (2.10),$$

dove *W* rappresenta il peso sinaptico della connessione tra il neurone *i* e quello *j*, mentre *x* rappresenta l'output della funzione di attivazione per il neurone *i* dello strato precedente *s – 1*. Le uscite prodotte dalla funzione di attivazione relativa ai neuroni dello strato *s* saranno calcolate come

$$x_s^j = P\left(I_j^s\right) \qquad (2.11),$$

dove *P* è la funzione di attivazione, già introdotta con le Equazioni (2.2) e (2.9).

2.5.3 APPRENDIMENTO *BACKWARD PROPAGATION*

L'algoritmo di apprendimento a retropropagazione dell'errore è retroattivo, e si colloca all'interno di quella classe di algoritmi chiamati *del primo ordine*, i quali ottimizzano il valore dei pesi facendo ricorso al metodo della discesa del gradiente (una tecnica di ottimizzazione che consente di determinare i minimi locali delle funzioni) e mantengono costanti i valori dei pesi ad ogni iterazione; accanto a questi, esistono anche algoritmi *del secondo ordine*, facenti riferimento al metodo di approssimazione di Newton (è un metodo applicabile su intervalli chiusi e limitati contenenti una sola radice, che consente di determinare l'intersezione della tangente alla curva nell'intervallo individuato con l'asse delle ascisse, e di ripetere ricorsivamente la procedura su intervalli più stretti fino all'equilibrio) ed all'inversa della matrice Hessiana (matrice delle derivate parziali seconde di una funzione in *n* variabili) per assegnare i valori ai pesi.
L'apprendimento a retropropagazione dell'errore prevede, come prima operazione, il calcolo dell'errore δ tra output stimato x_j^n ed output desiderato $\bar{x}_j^n$, relativo ad ogni neurone *j* dello strato di uscita *n*:

$$\delta_j^n = \left(x_j^n - \bar{x}_j^n\right) P'\left(I_j^n\right) \qquad (2.12),$$

dove $P'(\blacksquare)$ rappresenta la derivata prima della funzione di attivazione.
Successivamente vengono aggiornati i pesi delle connessioni tra i neuroni *i* dello strato *n – 1*, immediatamente precedente a quello di output, e *j* dello strato *n*:

$$\Delta W_{ij} = -\mu \delta_j^n x_i^{n-1} \qquad (2.13),$$

dove μ è un coefficiente che tiene conto della sensibilità dell'algoritmo all'errore tra uscita stimata e osservata.
Si procede poi a ritroso per tutti gli altri strati, in modo tale che gli errori di stima delle uscite si propaghino fino allo strato di input (*s = 1*):

$$\begin{cases} \delta_j^s = P'\left(I_j^s\right)\sum_r\left(\delta_r^{s+1}W_{rj}\right) \\ \Delta W_{ji} = -\mu\delta_r^{s+1}x_i^{s-1} \end{cases} \quad (2.14),$$

in cui le variabili assumono il significato già introdotto in precedenza.
Dopo l'immissione nella rete di ognuno degli *m* vettori che compongono il training – set, si procede all'aggiornamento dei pesi delle connessioni al fine di diminuire l'errore (E_k) associato proprio al vettore di esempio in uso (*k*):

$$E_k = \sum_j\left(x_j^n - \bar{x}_j^n\right)^2 \quad (2.15).$$

L'analisi del training – set viene ripetuta ricorsivamente fino a quando l'errore quadratico medio, relativo a tutto il set di esempio, non risulta inferiore ad una certa soglia prefissata, che, con le tecnologie a disposizione oggi, può tranquillamente essere fissata ad un valore molto prossimo a zero. L'errore medio può, pertanto, essere determinato come

$$E_m = \frac{1}{m} \times \sum_{k=1}^{m} E_k \quad (2.16).$$

Essendo questo un metodo del primo ordine, lascia invariato il valore dei pesi all'interno di ogni iterazione.
Sebbene il metodo della backpropagation sia consolidato e ritenuto valido, esso presenta comunque alcuni problemi, ad esempio la sensitività sia alle condizioni iniziali sia ai minimi locali della superficie di errore.

2.5.4 ALGORITMO DI APPRENDIMENTO DI *LEVENBERG – MARQUARDT*

L'algoritmo di apprendimento di Levenberg e Marquardt è un metodo del secondo ordine, dal momento che effettua un'interpolazione non lineare seguendo sia la tecnica di Newton, sia quella della discesa del gradiente. Questo algoritmo è da preferirsi rispetto ad altri, essendo più stabile e più rapido nel ricercare la soluzione, che può essere raggiunta pur partendo da un punto molto lontano. L'utilizzo ottimale di questo algoritmo riguarda la maggior parte delle reti neurali, mentre il metodo della backpropagation (Paragrafo 2.5.3) si rivela più adatto solamente per grandi reti neurali, composte da un gran numero di neuroni e di strati nascosti.
Come quasi tutti i metodi di linearizzazione, anche l'algoritmo di Levenberg – Marquardt è iterativo. In generale la rete neurale può essere scritta come una funzione che relaziona gli ingressi *I* alle uscite *x*, con parametri *W* e θ:

$$x = f(I;\Xi) \quad (2.17),$$

dove Ξ rappresenta la matrice che contiene sia i pesi sia i bias: $\Xi = [W\ \theta]$
I parametri si inizializzano, solitamente, per semplicità, al vettore unitario,

$$\Xi^T = [1,1,\dots,1] \quad (2.18).$$

Ad ogni iterazione esso viene aggiornato, in modo da rendere sempre più piccolo l'errore quadratico medio di scarto:

$$\Xi^{aggiornato} = \Xi + \delta \quad (2.19),$$

che si ottiene linearizzando l'espressione della rete neurale, definita nell'equazione (2.17):

$$f\left(I_i, \Xi_i^{aggiornato}\right) \approx f(I_i, \Xi) + J_i(\delta) \quad (2.20).$$

Nell'equazione (2.20), la matrice *J* rappresenta lo Jacobiano del parametro di aggiornamento, ossia il gradiente.
Al fine di minimizzare l'errore di stima,

$$\min_{\Xi} S = \min_{\Xi} \sum_{i=1}^{m} \left(\bar{x}_i - f(I_i, \Xi)\right)^2 \quad (2.21),$$

occorre porre a zero la derivata di questa rispetto all'aggiornamento,

$$\frac{\partial S}{\partial \delta} = 0 \quad (2.22),$$

per calcolare i valori dei parametri che consentono di minimizzare l'errore di stima.

2.5.5 *EARLY STOPPING*

Qualunque sia l'algoritmo di ricerca del minimo che viene adottato, esso viene implementato all'interno di una procedura di addestramento che viene chiamata *early stopping*. Tutto il set di dati viene suddiviso in tre parti, la prima, più consistente, utilizzata nell'addestramento vero e proprio (*training set*, pari, di norma, al *60%* dei dati totali), la seconda, utilizzata per validare l'addestramento (*validation set*, pari al *20%*), e l'ultima, utilizzata per effettuare un ulteriore test sulla procedura (*testing set*, anch'esso pari al *20%*). Durante l'addestramento, la rete cerca, noti gli ingressi, di riprodurre le uscite modificando pesi e bias, applicando uno degli algoritmi visti in precedenza, e valuta la prestazione calcolando l'errore quadratico medio (*MSE*). Per come operano gli algoritmi di ricerca del minimo, ad ogni passo tale errore, calcolato sul solo set di addestramento, diminuisce. Oltre un certo passo, invece, esso tende ad aumentare sul set di validazione: quando accade ciò, la rete perde di generalità (fenomeno di *overfitting*); bisognerebbe cercare di costruire una rete neurale che minimizzi questo problema, cercando di individuare il numero di neuroni e di strati interni corretto. Comunque, l'addestramento si considera concluso quando si rileva che l'errore quadratico medio sul set di validazione è cresciuto per un certo numero di epoche, come riassunto nella seguente Figura 2.5.

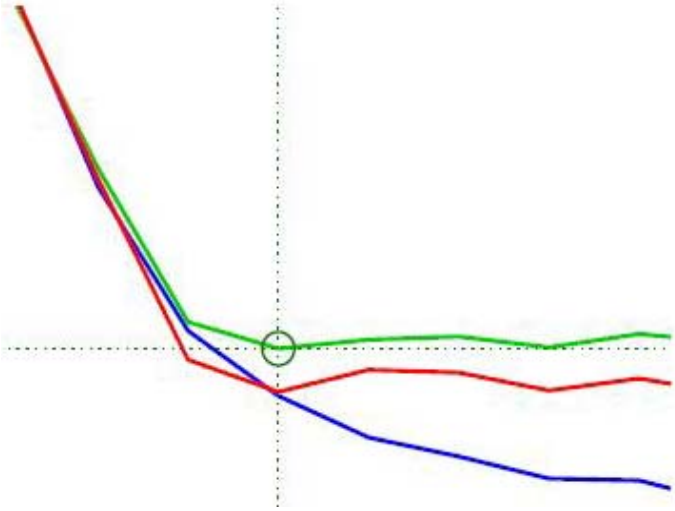

*Figura 2.5 – Grafico dell'errore quadratico medio nella procedura dell'*early stopping*. In blu è rappresentato l'MSE del set di training, in verde quello del set di validazione ed in rosso quello del set di test. L'addestramento si ritiene concluso quando si raggiunge il minimo anche sul set di validazione (da Demuth* et al.*)*

Si cita, sebbene esuli dagli interessi di questo lavoro, che esiste un'altra metodologia per arrestare la procedura di addestramento a risultato ottenuto: il metodo della Regolarizzazione Bayesiana, che fornisce risultati quasi equivalenti a quelli dell'early stopping. [Dung Doan, Liong, 2004]

2.6 APPLICAZIONI PRATICHE DELLE RETI NEURALI

Le reti neurali sono dei modelli sviluppatisi in anni recenti, il cui utilizzo è ancora in espansione. Nonostante questa recente storia, le Reti Neurali si sono diffuse in tutti quei campi dove i modelli analitici hanno fallito oppure dove la conoscenza dei dati sia incerta, insufficiente, o errata.
Tra i principali utilizzi delle reti neurali, si trovano i software *OCR* per il riconoscimento di testo da documenti sottoposti a scansione, oppure le analisi finanziarie [Gallo, 2007], meteorologiche, mediche, o, ancora, sociali. [Di Franco, 1998]
Di maggiore interesse, è l'utilizzo delle reti neurali per individuare e riconoscere parole molto simili tra loro, che vengono registrate e successivamente introdotte nella rete stessa, che, dall'analisi degli spettri di frequenza delle onde sonore, riesce, con una buona approssimazione, ad individuare le parole in modo corretto. [Lang *et al.*, 1989]
Ancora, in anni recenti [Baglietto *et al.*, 2000], le strutture neuronali artificiali hanno trovato uno sviluppo nei confronti dei sistemi di comunicazione, soprattutto quelli complessi, nei quali occorre diramare una grande quantità di informazioni provenienti da più nodi diversi verso altrettanti numerosi punti. Occorre ottimizzare l'acquisizione, la memorizzazione e la trasmissione delle comunicazioni, operazioni molto complicate e controllate da equazioni fortemente non lineari, spesso da sviluppare su un orizzonte temporale molto lungo, quando non infinito. Se per la gestione di questi sistemi si adottassero i classici schemi della Programmazione Dinamica, ciascun Decisore potrebbe considerare solo una minima parte dell'insieme dei dati, quella che transita all'interno della sua area di competenza, e dovrebbe trascurare tutto il resto, generando inevitabilmente analisi imprecise, che si ripercuoterebbero a tutti gli altri analisti. Mediante uno schema risolutivo neurale, invece, è possibile tenere in considerazione tutta la mole di dati che interessano il sistema, e gestirli nel loro insieme.
Le applicazioni principali, comunque, delle reti neurali, anche perché collegate a quanto si andrà a svolgere nel presente lavoro di Tesi, riguardano lo studio ed il controllo dei sistemi idrici, dal momento che, ci si rese conto fin dalle prime utilizzazioni, era possibile risolvere tutti quei problemi inerenti le risorse idriche, che si erano incontrati utilizzando i modelli tradizionali, quelli analitici.
Uno dei maggiori vantaggi dell'utilizzo delle Reti nei confronti dei più tradizionali modelli analitici riguarda la forma della funzione distribuzione di probabilità dei dati di input: se per garantire una stima efficiente nel caso di alcuni tra i più comuni modelli analitici (gli *ARX*, mentre decade nei modelli fisicamente basati), gli ingressi devono essere distribuiti normalmente, ciò non è più vero se si fa ricorso alle strutture neurali, dal momento che i dati di input possono assumere una distribuzione qualsiasi.
Inoltre, quando tra i dati di ingresso alla rete si utilizzano anche gli afflussi al sistema, è possibile costruire un buon previsore di questi ultimi, se l'architettura del modello prevede più di uno strato nascosto.
Le strutture neurali trovano, come detto, un ampio campo di utilizzo nella previsione dei valori assunti dalle variabili che quantificano le risorse idriche quando, come avviene di sovente, si debbano prevedere quantità da input incerti o stocastici. Fin dai primi sviluppi delle Reti Neurali per questi scopi si sono scelte le architetture feedforward, ma oggi vengono sempre più utilizzate quelle ricorrenti. Infatti, se con le reti feedforward è necessario che i sistemi dinamici siano trattati esplicitamente, con quelle ricorrenti alcune proprietà dinamiche

possono essere implicite, quindi non devono essere descritte nell'architettura del modello; inoltre, l'utilizzo di strutture neurali ricorrenti consente sia di trattare problemi definiti su un orizzonte di progetto mobile (assai frequente nel campo delle risorse idriche), non trattabile con architetture feedforward, sia di avere buone prestazioni quando vi sia un notevole rumore nei dati di ingresso (*noisy data*).
Di contro, tuttavia, le strutture ricorrenti possono incontrare alcune difficoltà nella ricerca della soluzione quando i dati da processare abbiano correlazioni molto estese nel tempo.
Comunque, le reti neurali diventano assai più vantaggiose se si guarda alla complessità computazionale del problema, dal momento che, all'aumentare del numero di stati del sistema il tempo di calcolo aumenta esponenzialmente nel caso della Programmazione Dinamica, linearmente altrimenti. Anche in un sistema piccolo può risultare vantaggioso utilizzare il modello neurale, perché il numero di stati può essere funzione del tipo di analisi che si desidera condurre. In casi pratici, comunque, a meno che non si tratti di reti idriche relativamente piccole, create per scopi dimostrativi o didattici, il numero di serbatoi e canali è sufficientemente elevato tale da rendere maggiormente conveniente l'utilizzo delle architetture neurali in luogo di altre strutture modellistiche (basti pensare, in Italia, ai sistemi fluviali del Ticino, dell'Adda, del Po o del Tevere, per le cui analisi non è possibile limitarsi alle sole aste fluviali, ma occorre includere anche la maggior parte degli affluenti e dei bacini che su di essi si innestano; i sistemi fluviali italiani sono piuttosto piccoli, se si intende analizzare reti più grandi, come possono essere il Nilo, lo Zambesi, il Rio delle Amazzoni o il Fiume Giallo, il numero di ingressi e di uscite tende ad incrementarsi ulteriormente).
Ciò mostra come si renda necessario individuare un modello, o un insieme di modelli, che fornisca un buon compromesso tra la qualità delle prestazioni erogate e la rapidità con cui esse vengono raggiunte. Le Reti Neurali, pur essendo inferiori in termini assoluti di prestazioni ad un qualsiasi algoritmo non lineare di Programmazione Dinamica, si ritiene che possano ben rispondere a questi requisiti, riuscendo comunque ad erogare delle prestazioni soddisfacenti in un tempo relativamente piccolo, che si mantenga sostanzialmente invariato qualunque sia la dimensione del sistema che cercano di descrivere.
Altri esempi applicativi delle Reti in tematiche ambientali legate ai bacini idrici, riguardano l'utilizzo che è possibile fare di questi modelli per prevedere l'eutrofizzazione algale. I modelli normalmente utilizzati per descrivere tale fenomeno, sono costruiti facendo ricorso ad equazioni lineari, che non riescono a riprodurre perfettamente le relazioni tra nutrienti disciolti nell'acqua e produzione di clorofilla o crescita, trascurando anche alcuni importanti fenomeni che generano eutrofizzazione, come, ad esempio, la struttura dell'ecosistema. Mediante le Reti Neurali è possibile descrivere bene queste relazioni, ottenendo dei risultati soddisfacenti. [Karul *et al.*, 2000]
Peraltro, l'utilizzo delle Reti Neurali per modellizzare la gestione di un sistema idrico con l'obiettivo di riceverne il massimo beneficio possibile, non è un campo del tutto nuovo, essendo già stato sperimentato, in modo abbastanza soddisfacente per simulazioni dei modelli idrologici ed idrodinamici per limitati periodi di tempo, per un bacino del fiume Orinoco, in Venezuela. [Solomatine, Avila Torres, 1996]

3

PRESENTAZIONE DEL PROBLEMA ED APPROCCI RISOLUTIVI

In questo Capitolo si introduce il problema specifico delle reti di serbatoi idrici, ponendo l'attenzione sulle modalità operative, sulle problematiche e su quali sono gli obiettivi.

3.1 IMPOSTAZIONE DEL PROBLEMA

3.1.1 OTTIMIZZAZIONE NEURALE

Vista l'importanza che hanno nell'ambiente circostante, è importante riuscire a gestire nel modo migliore possibile i sistemi idrici. Scopo di ciò è disporre, ad ogni istante temporale, dei livelli opportuni (ossia tali da garantire il fabbisogno di acqua al successivo istante, ma anche la sicurezza dei litorali e delle sponde) all'interno dei serbatoi, dei rilasci adeguati alle esigenze, delle portate circolanti nei canali coerenti e fisicamente realizzabili. Questo si rende necessario al fine di massimizzare l'utilità totale derivata dalla vendita dell'energia elettrica prodotta, corrispondente all'unico indicatore considerato nella presente trattazione, sebbene, in realtà, la gestione dei sistemi reali deve soddisfare moltissimi portatori, i quali vogliono ottenere il massimo beneficio possibile per il loro interesse ad ogni passo temporale.
Tuttavia, per lo scopo di questa sperimentazione, d'interesse è massimizzare solamente l'utilità totale, obiettivo per il quale si rende necessario determinare un orizzonte progettuale finito.
Un confronto tra diversi algoritmi di ottimizzazione è stato condotto in un precedente lavoro di Tesi [Taddio, Togni, 1993], che aveva come riferimento fisico la rete del Fiume Zambesi, la medesima utilizzata in questa trattazione, e descritta nel Capitolo 4.
È bene, a questo punto, richiamare un importante concetto, dal quale non si può mai prescindere quando si ha a che fare con i sistemi ambientali in generale (dunque non solamente con quelli idrici): il concetto di "gestione ottima" non può essere sviluppato in senso letterale, dal momento che non esiste mai una politica perfetta in assoluto. Può esistere, ed è quella che si cerca di individuare solitamente, una politica migliore rispetto ai risultati della gestione storica, oppure possono esistere politiche che si avvicinano all'*utopia*, che corrisponde al punto, nello spazio degli obiettivi, in cui ognuno dei Portatori riesce ad ottimizzare il proprio, senza precludere agli altri di poter ottimizzare quelli di loro interesse: questo punto (l'utopia) non può mai essere fisicamente raggiunto, perché l'ottimizzazione di un determinato target, inevitabilmente causa insoddisfazione per altri punti di vista.
Bisogna altresì tenere in conto che è possibile, quando si conoscono determinati modelli e si dispone di un certo livello tecnologico, riuscire a determinare la migliore gestione mai ottenuta fino ad allora (dove "migliore", è da intendersi limitatamente alla distanza dal punto

di utopia), ma è molto probabile che in futuro, quando sia le conoscenze sia le tecnologie saranno progredite, il risultato ottenuto sarà considerato non buono, perché sarà stata individuata qualche politica che fornisce una prestazione ancora più vicina all'utopia.
Riallacciandosi a quanto scritto prima, appare allora chiaro che, disponendo di un modello in grado di apprendere, come una Rete Neurale, esso sarà in grado di individuare una politica tanto migliore quanto più ampio e vasto sarà il campo di addestramento, ossia la casistica degli input che possono interessare il sistema.
Essendo peraltro la rete in grado di imparare dalle prestazioni passate, se l'addestramento riesce ad essere molto ampio, è più probabile che di fronte ad una situazione nuova, sconosciuta, ad esempio afflussi particolarmente intensi o scarsi, il modello riesca a rispondere in modo corretto, rispetto al caso in cui si conduca un addestramento povero o incompleto.

3.1.2 SVILUPPO DEL PROBLEMA

Il problema che si tratta in questo lavoro si compone di diverse parti.
La prima fase, necessaria proprio per garantire che l'addestramento sia efficace (presupponendo, dunque, un algoritmo *supervised*, come descritto nel Paragrafo 2.2.4), consiste nell'individuazione di un gran numero di serie di afflussi, e di rilasci corrispondenti. Come sempre, quando si ha a che fare con i sistemi naturali, i dati reali sono molto pochi, corrispondenti alle realizzazioni dei fenomeni che si sono nel tempo effettivamente verificate. Queste originano una sola serie, assolutamente insufficiente per un addestramento anche discreto di una Rete Neurale.
Occorre, pertanto, costruire un numero molto elevato di serie sintetiche, che costituiscono i "vettori di esempio" (Paragrafo 2.5.1): questa fase, piuttosto ampia e laboriosa, è trattata separatamente nel Capitolo 5.
Indipendente dalla generazione degli afflussi, è la ricerca dell'architettura della rete neurale, che deve rispondere alle caratteristiche di generalità e semplicità esposte sempre nel Paragrafo 2.2.4.
Parallelamente alla ricerca dell'architettura ottimale per la rete, occorre procedere con l'addestramento di questa, dal momento che è possibile non riuscire ad addestrare correttamente una rete con una certa architettura, mentre una struttura differente può generare prestazioni molto migliori. Come si capisce, queste due fasi sono da eseguire insieme, ricorsivamente, fino ad individuare un buon compromesso che, tuttavia, non infici le caratteristiche di generalità e semplicità. Solitamente, poiché riesce ad avere buone prestazioni pur mantenendosi generale, si utilizza un'architettura con un solo strato nascosto, ma con un numero di neuroni da determinare a seconda del problema in analisi.
Per valutare l'addestramento, ovviamente, si effettua un confronto tra i risultati ottenuti da una certa architettura con quelli prodotti da un altro processo di ottimizzazione, nella fattispecie l'algoritmo Relax, [Bertsekas, Tseng, 1985] [Bertsekas *et al.*, 1989] ampiamente testato e validato, e comunemente utilizzato (vista l'importanza che riveste per il presente lavoro, se ne offre una descrizione dettagliata nell'Appendice A3).
Esiste, comunque, un'importante differenza tra come agisce Relax e, invece, come operano le reti neurali: il primo algoritmo effettua un'ottimizzazione assumendo, che tutti gli ingressi al sistema (per esempio lo scenario degli afflussi) siano noti su tutto l'orizzonte temporale considerato, mentre la rete neurale opera in senza conoscere il valore assunto dagli ingressi lungo tutto l'orizzonte progettuale.
L'algoritmo Relax si richiama all'interno di un applicativo DOS, Aquafun (il cui utilizzo è descritto nell'Appendice A1), scritto con lo scopo principale di confrontare le prestazioni della rete idrica ottenute dall'ottimizzazione con lo stesso Relax rispetto a quelle ottenute con altri algoritmi (Netflo e Spath, meno buoni a livello di prestazioni per il caso dello Zambesi)

[Taddio, Togni, 1993], ma adatto anche per risolvere un Problema di Ottimizzazione qualsiasi con uno dei tre algoritmi.
In particolare, Aquafun richiede come input i valori degli afflussi su tutto l'orizzonte temporale, la struttura dei serbatoi e i volumi di invaso iniziale e finale, oltre agli estremi di volume per ogni periodo, l'evaporazione, la struttura dei canali e delle funzioni di utilità, producendo come output il valore di invaso ad ogni periodo, la portata circolante nei canali ad ogni periodo, ed il valore delle utilità, per ogni canale e complessiva, suddivise per periodo e globali su tutto l'orizzonte di progetto.
La Rete Neurale, invece, può richiedere come dati di input l'insieme di afflussi e valori di invaso, eventualmente anche un indicatore del tempo, producendo in uscita i rilasci (ma è possibile costruire reti con input più semplici, ad esempio i soli invasi).
Prima di procedere a dettagliare quanto sinteticamente esposto in questo paragrafo, la Figura 3.1 a pagina seguente riporta un diagramma di flusso della procedura che si intende seguire.
Il software Matlab, usato insieme al suo analogo open – source, Octave, disponibile su piattaforma Cygwin e Cygwin – X, per l'adattamento a Windows Vista, offre, oltre alla possibilità di costruire funzioni per l'automatizzazione delle procedure, anche dei *tools* per la costruzione e l'utilizzo di Reti Neurali, e semplifica enormemente l'esportazione dei risultati su file di testo, molto più versatili da utilizzare.
Per poter costruire i vettori di addestramento, occorre rielaborare, adattandoli, i file testuali necessari ad Aquafun: ancora una volta, Matlab od Octave si rivelano utili per questi scopi.

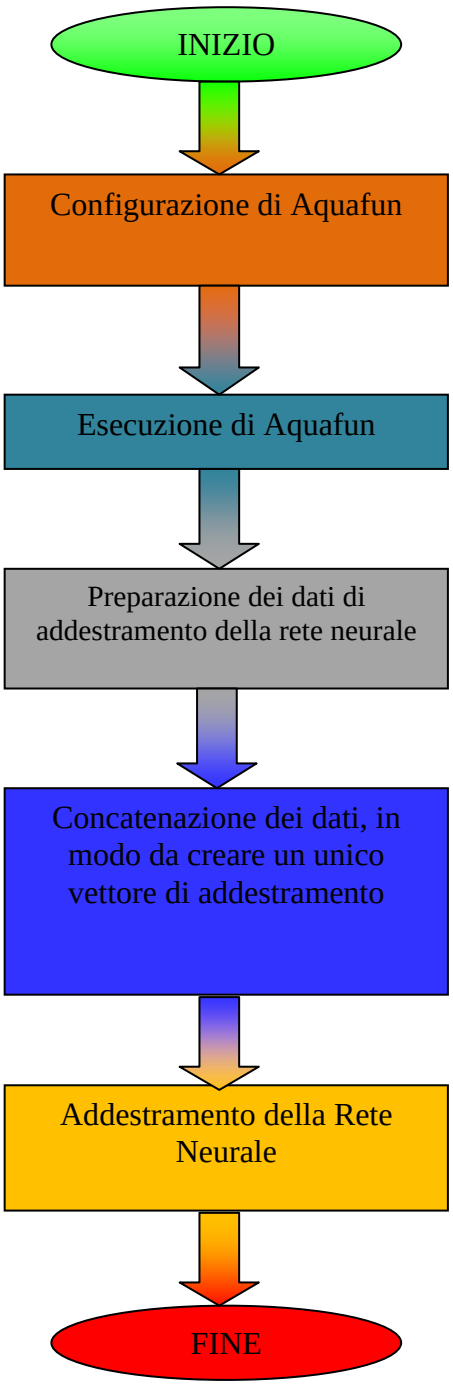

Figura 3.1 – Diagramma di flusso della procedura da seguire per l'ottimizzazione del problema di allocazione attraverso l'utilizzo congiunto delle Reti Neurali e dell'ottimizzatore lineare Aquafun

3.2 SCELTA DEL MIGLIOR MODELLO NEURALE

L'architettura neurale vera e propria, definitiva, ossia quanti strati nascosti inserire e quanti neuroni per ciascuno di essi implementare, deve essere scelta durante l'addestramento, in modo da ottenere le migliori prestazioni possibili.
Anzitutto, occorre scegliere se sviluppare una rete neurale per ciascuno dei serbatoi presenti nella rete idrica, o se, invece, costruire un'unica architettura neurale per tutto il sistema idrico in esame.
È ovvio che questa decisione va presa all'inizio, preliminarmente alla raccolta ed alla rielaborazione dei dati necessari all'addestramento, perché a seconda del modello neurale che si sceglie, la matrice di addestramento risulterà molto diversa.
Nel caso in cui si scelga di costruire una rete neurale per ogni serbatoio, la forma della matrice di addestramento può mutare, a seconda che si scelga di adottare un'*informazione decentralizzata* oppure *centralizzata*. Nel primo caso, si assume che ogni serbatoio sia indipendente dagli altri, per cui il comportamento di ognuno non è minimamente influenzato dallo stato e dal comportamento degli altri. Questa ipotesi, tuttavia, è piuttosto irrealistica nei sistemi idrici reali, per cui si adotta un'informazione di tipo centralizzato assumendo, come in effetti avviene, che il comportamento di ogni serbatoio sia correlato a quello di tutti gli altri.
Il differente percorso che si può seguire è quello di costruire una sola rete neurale per tutto il sistema, costruendo, pertanto, anche un'unica matrice di addestramento per tutto il sistema.
Nel presente lavoro di Tesi, comunque, si sono esplorate più strade: a priori, l'idea migliore appare quella di costruire un'unica Rete Neurale per tutto il sistema.
Due sono le ragioni che hanno portato a questa conclusione: la prima risponde a quella che è la reale gestione del sistema idrico dello Zambesi, il quale è controllato da un unico Decisore, che si occupa di tutti i serbatoi nel loro insieme; la seconda è un motivo puramente logistico, corrispondente ad una scelta di semplificazione di tutta la procedura, sia quella di costruzione vera e propria della rete, sia il training della stessa. Se, infatti, si fosse deciso di progettare una struttura neurale per ognuno dei serbatoi, si sarebbero dovute costruire, addestrare e validare numerose reti, complicando il lavoro in misura eccessiva.
La scelta di costruire una sola rete risponde meglio all'essenziale requisito di generalità, anche se potrebbe necessitare di una durata di addestramento più elevata, dato il maggior numero di dati che devono essere forniti all'architettura.
L'impegno computazionale nell'elaborare una singola struttura per tutto il sistema in luogo di una per ciascun serbatoio, potrebbe non essere molto dissimile, perché, una volta definita la struttura, si deve procedere ad un solo addestramento, ma, di contro, i dati da trattare sono in numero maggiore.
Per quanto riguarda le prestazioni, la rete neurale unica dovrebbe portare a risultati migliori, poiché opera con una maggiore quantità di dati, ed anche la velocità risolutiva di questo modello dovrebbe essere più elevata rispetto a quello che lavora sui singoli serbatoi (se si considera il tempo totale, comprensivo delle simulazioni e dell'estrazione dei risultati).
Stabilita la struttura dell'architettura neurale, devono essere effettuate alcune considerazioni sulla costruzione della matrice di addestramento, composta da un insieme di valori di input, e dalle uscite dell'ottimizzatore lineare che occorrono per calcolare l'errore di stima dei risultati della stessa rete.
Nei dati di input devono essere inclusi, per ognuno dei periodi sui quali si effettua la simulazione, i valori dei volumi degli invasi dei singoli serbatoi presenti nella rete, ed i valori degli afflussi, suddivisi anche questi per serbatoio, in modo da avere una descrizione puntuale di tutti gli ingressi al modello
Se si decidesse di implementare una struttura neurale che calcola i rilasci sulla base della precedente equazione di bilancio, sarebbe necessario considerare grandezze riferite a diversi istanti temporali, precisamente gli invasi al tempo t e gli afflussi al tempo $t+1$, ipotizzandole

note entrambe allo stesso istante, ossia *t*. Questo non è ovviamente possibile, perché gli afflussi relativi a *t+1* sono quelli che si verificano nell'intervallo $[t, t+1)$, e sono quindi deterministicamente conosciuti solamente nell'estremo destro dell'intervallo temporale.
Per considerarli in un qualsiasi momento precedente occorre introdurre un previsore, che complicherebbe molto la struttura della rete neurale, dal momento che andrebbe ottimizzato anch'esso. D'altro canto, l'utilizzo di un ottimizzatore lineare come Aquafun presuppone che ad ogni istante si conosca il valore dell'invaso (ottenuto dalla simulazione al passo prima), e tutta la serie di afflussi che insiste sul sistema (relativo, evidentemente, al precedente step temporale, per questo noto al momento di interesse), in modo da poter risolvere l'equazione di bilancio (3.1) appena presentata. Un ottimizzatore lineare non prevede, infatti, dei previsori per stimare gli afflussi futuri. Ai fini dell'addestramento della rete neurale, pertanto, si considerano o i soli valori di invaso *s*, deterministicamente noti al tempo *t*, oppure la coppia di dati

$$[s_t \quad a_{[t-1,t]}] \qquad (3.1),$$

ossia l'insieme dei volumi di invaso e degli afflussi occorsi nel periodo precedente *[t-1, t]*, e quindi noti a *t* (gli stessi dati di ottimizzazione lineare), cui eventualmente aggiungere un contatore temporale per tenere in conto della ciclicità del sistema.

Comunque, quanto esposto sinora è frutto solamente di ipotesi, che sono pertanto da verificare: uno dei possibili sviluppi futuri di questo lavoro di Tesi potrebbe essere comparare la velocità di addestramento del modello unico e di quello per serbatoi, verificando se davvero la differenza è rilevante, o se invece trascurabile; accanto a questo, si potrebbe valutare anche se il modello unico produce risultati sensibilmente differenti dal modello per singoli invasi.
In fase di addestramento vero e proprio, occorrerà affinare la struttura del modello, scegliendo proprio il numero di neuroni.
In un passaggio successivo, al fine di migliorare la precisione del modello, deve essere sicuramente presa in considerazione l'ipotesi di introdurre un previsore nell'architettura neurale, in modo tale da poter disporre dei valori di afflusso a_{t+1} insieme a quelli di invaso s_t.

3.3 DEFINIZIONE DEGLI INPUT ALLA RETE NEURALE

3.3.1 COSTRUZIONE DELLA MATRICE *I* DI INPUT

Accanto al problema di quali dati di input considerare nella matrice di addestramento, occorre altresì considerare che l'istante iniziale e finale, e quelli immediatamente prossimi ad essi sono soggetti ad effetti di bordo, in cui vengono meno sia la correlazione sia i legami tra i valori consecutivi; inoltre, qualora si decidesse di utilizzare un previsore, sussiste il problema che il vettore di invasi relativo al primo istante non trova corrispondenze negli afflussi, e lo stesso avviene per i valori di afflusso relativi all'ultimo istante. Pertanto si includono nella matrice di input alla rete neurale solamente i periodi centrali, tralasciando un certo numero di istanti all'inizio ed alla fine dell'orizzonte di progetto. Il numero di periodi da considerare non può essere fissato a priori, ma sarà da includere, e valutare, nella procedura di addestramento.
A scopo chiarificatore, nella seguente Figura 3.2, si spiega come si ottiene la matrice di input *I*, che avrà dimensione *[d, m]*, dove *d* è il numero di periodi che si prendono in considerazione (una frazione della lunghezza dell'orizzonte di progetto *h*). La figura rappresenta il caso più complesso di matrice di addestramento, quella in cui i volumi di invaso sono riferiti al tempo *t* e gli afflussi, invece, al tempo *t+1*. Ogni colonna m della matrice è formata da una coppia di vettori, quello di afflusso e quello di invaso.

Si tenga presente che, e questo ha validità generale, ad ogni colonna della matrice di input deve essere associato un neurone dello strato di ingresso alla rete neurale: ognuno di questi è indicato con *j*.

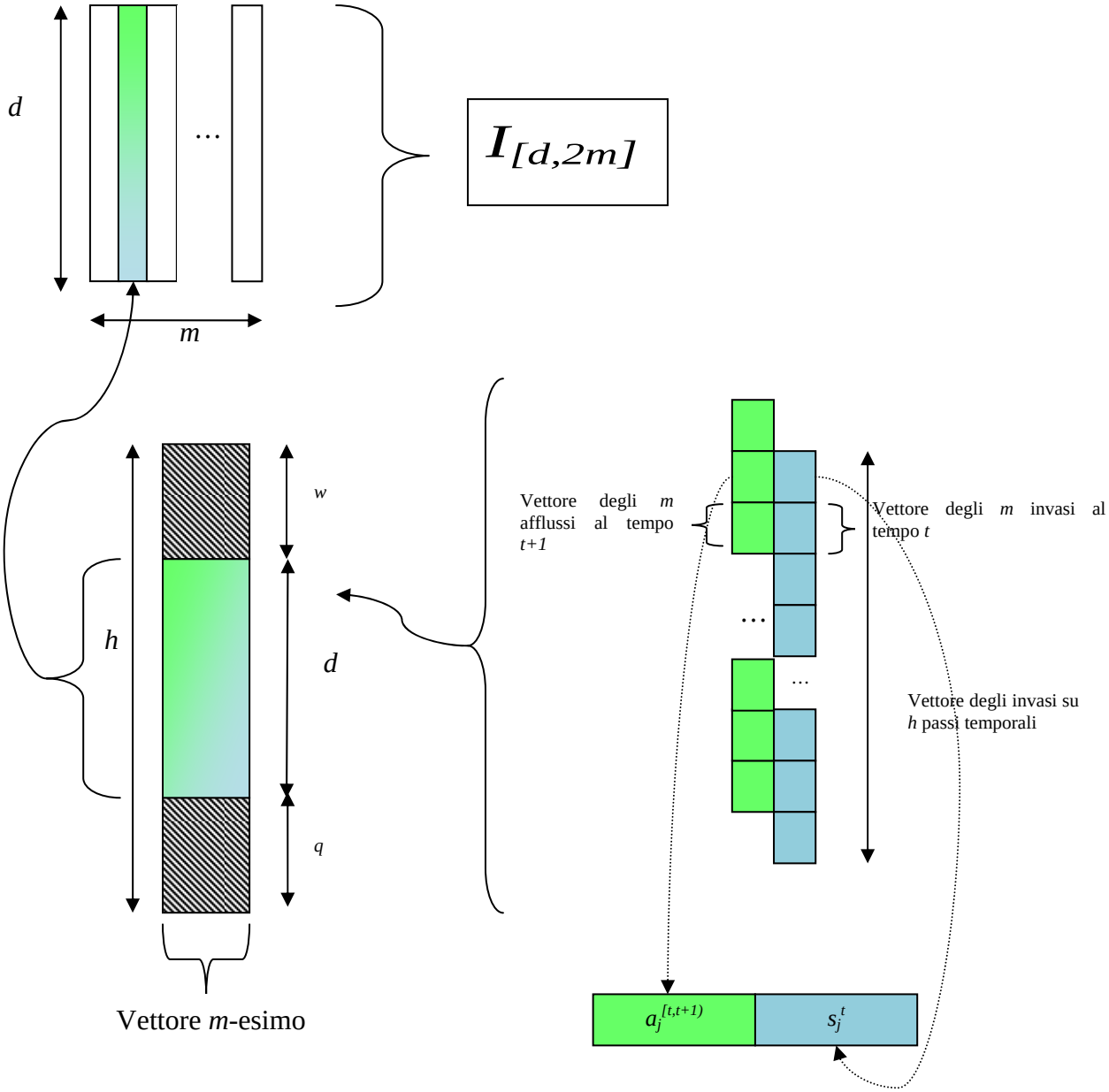

Figura 3.2 – Creazione della matrice di input I alla rete neurale: partendo dai singoli vettori di invaso (in azzurro) e di afflusso (in verde), si procede al loro affiancamento, quindi si considera un numero limitato di periodi; affiancando gli m vettori di afflussi + invasi si ottiene la matrice di input

3.3.2 MATRICE DEI CONTROLLI DI OTTIMIZZAZIONE

Per quanto riguarda l'insieme dei controlli che la rete neurale deve produrre in output, esso deve essere, dal punto di vista dimensionale, identico a quello prodotto dall'ottimizzatore lineare Aquafun.
Non necessariamente la dimensione del vettore di output deve essere la stessa di quello di ingresso, dal momento che potrebbe verificarsi l'eventualità che uno o più serbatoi rilascino in più canali distinti, per ognuno dei quali occorre risolvere il problema di gestione. Il numero di periodi per cui calcolare i controlli è invece limitato alla stessa durata degli ingressi, identificata già con *d*.
Pertanto, la matrice di ottimizzazione lineare $O^{confronto}$ sarà della forma di quella indicata nella seguente Figura 3.3.

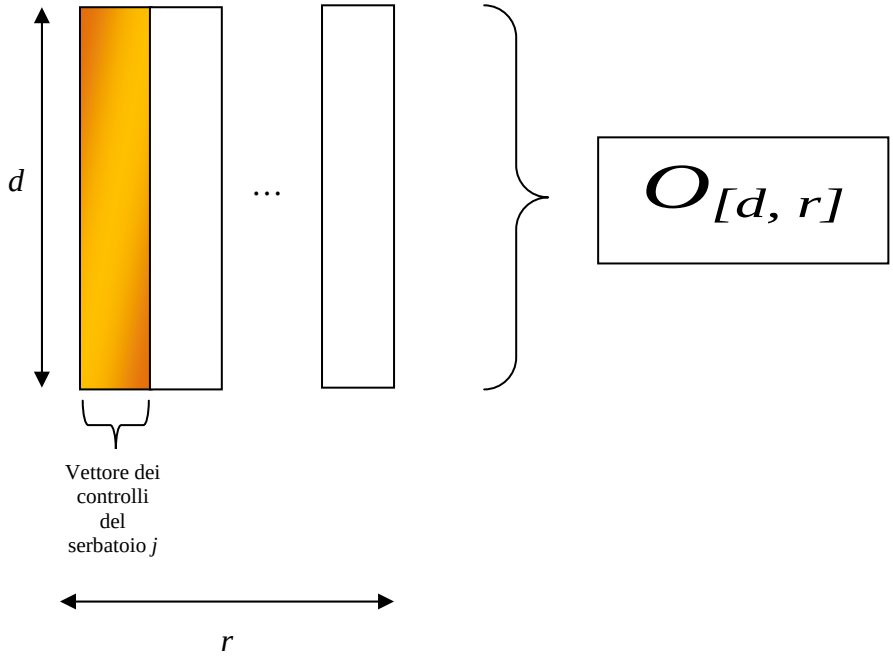

Figura 3.3 – Costruzione della matrice di ottimizzazione lineare

In analogia con la simbologia già introdotta, la matrice di uscita sarà definita da *d* righe, corrispondenti alla durata della simulazione, e *r* colonne, corrispondenti al numero di controlli totali. In generale, vale la relazione

$$m \neq r \qquad (3.2).$$

Questa differenza dimensionale non è tuttavia importante, perché la rete neurale può essere costruita come si vuole, dimensionando liberamente il vettore di ingresso e quello di uscite, perché tutte le relazioni sono implementate automaticamente all'interno della rete stessa.
Sebbene quella descritta nel presente paragrafo sia la matrice da utilizzare per il confronto dei risultati, ciò che deve produrre la rete neurale è esattamente analoga, sia in termini dimensionali sia in termini semantici. Per questo, la rete neurale deve disporre di un numero di neuroni nello strato di uscita pari ad *r*.

3.3.3 CONCATENAZIONE DEI VETTORI: MATRICE DI ADDESTRAMENTO *A*

Per poter procedere con l'addestramento della rete neurale occorre concatenare tutti i dati che sono stati fino ad ora ottenuti, al fine di disporre di una matrice di addestramento da introdurre nella struttura.
Ricapitolando:

1) Non si considerano tutti i periodi *h* dell'orizzonte di progetto, ma solamente la porzione centrale, lunga *d* periodi, al fine di tralasciare gli effetti di bordo relativi alla correlazione;
2) Si estraggono i valori di invaso relativi ad ognuno dei serbatoi (*m*) presenti nella rete idrica, per ognuno dei periodi considerati. Ad ogni serbatoio si fa corrispondere un vettore;
3) Si estraggono i valori di afflusso, per ogni serbatoio, relativi ad ogni periodo, e si costruiscono tanti vettori quanti sono i serbatoi (*m*);
4) Si affiancano i valori di invaso a quelli di afflusso, in modo tale che gli invasi relativi al periodo *t* abbiano una corrispondenza negli afflussi relativi al periodo successivo, *t + 1*, per poter risolvere l'equazione di bilancio, introdotta nel capitolo 1;
5) Si estraggono, sempre per i soli periodi *d* considerati, i rilasci relativi ad ognuno dei serbatoi, costruendo un vettore per ognuno dei controlli prodotti. In generale, il numero di vettori di controlli (*r*) può differire da quello di invasi o afflussi (*d*);
6) Si affiancano i vettori di controllo alle coppie invasi – afflussi, già costituite con i passaggi precedenti.

La matrice di addestramento (definita da *A*) prodotta, pertanto, viene costruita come mostrato nello schema riportato nella seguente Figura 3.4.

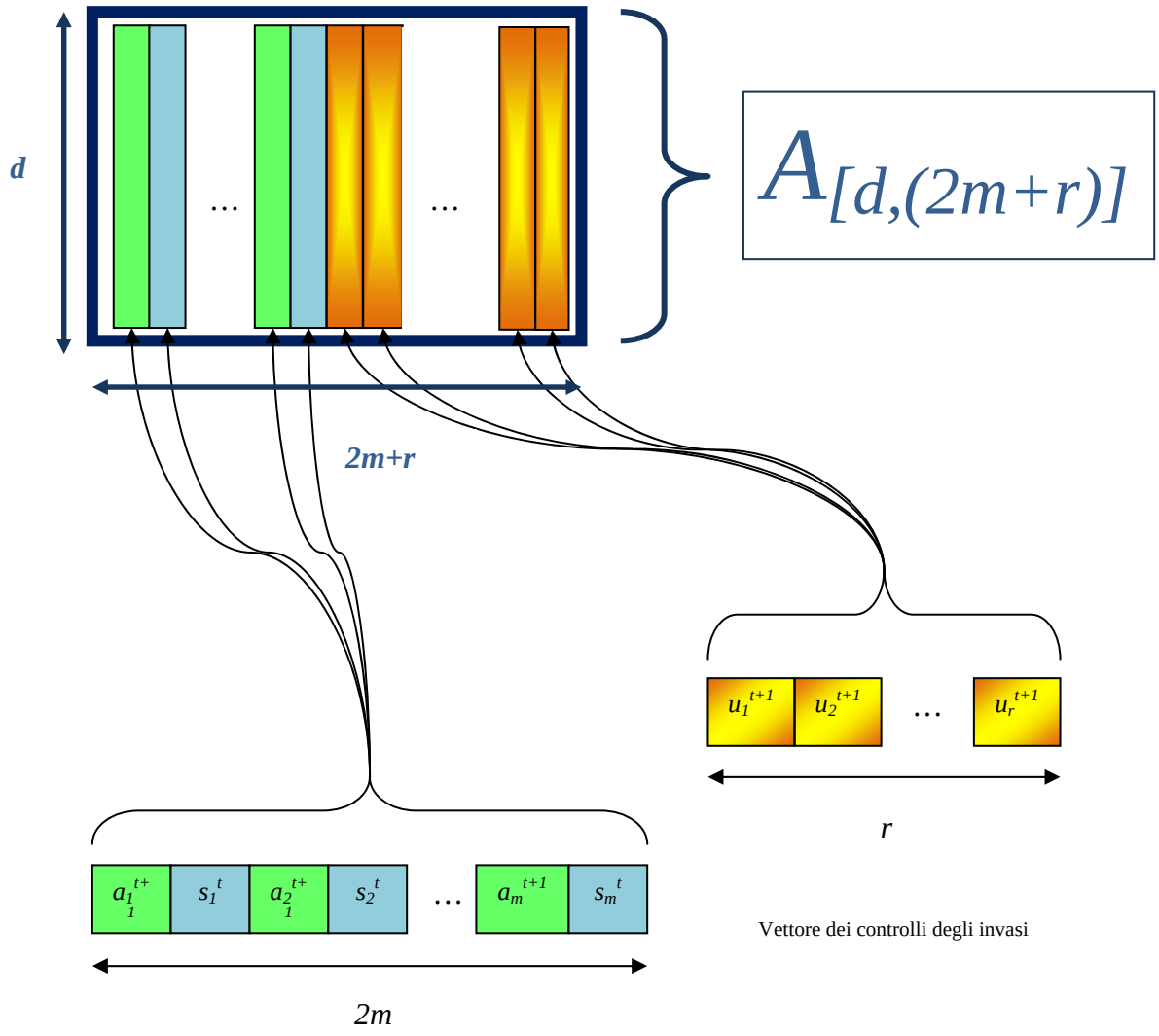

Figura 3.4 – Costruzione della matrice di addestramento dell'architettura neurale

Il modo più semplice ed immediato per costruire la matrice sarebbe stato quello di affiancare tutti i vettori relativi ad un singolo serbatoio, vale a dire afflusso, invaso e controllo (prendendo a riferimento la Figura 3.4 soprastante, il vettore verde, quello azzurro e quello giallo – arancione), ripetendo questa tripletta per tutti i nodi di interesse della rete idrica. Tuttavia, si è scelto di non agire in questo modo per due motivi: il primo perché si è preferito scindere in modo chiaro i dati necessari all'addestramento vero e proprio (ossia afflussi e invasi), da quelli utilizzati invece per la valutazione della prestazione della rete neurale costruita (le sequenze dei controlli, per l'appunto); il secondo motivo per cui le due sottomatrici sono state separate fa riferimento al diverso numero di colonne di input da quelle di controllo.

3.3.4 SEMPLIFICAZIONE DELLA MATRICE IN ASSENZA DI PREVISORE

Quando la matrice di addestramento è costituita unicamente dagli invasi e dai controlli di ottimizzazione lineare, essa avrà la struttura rappresentata nella seguente Figura 3.5.

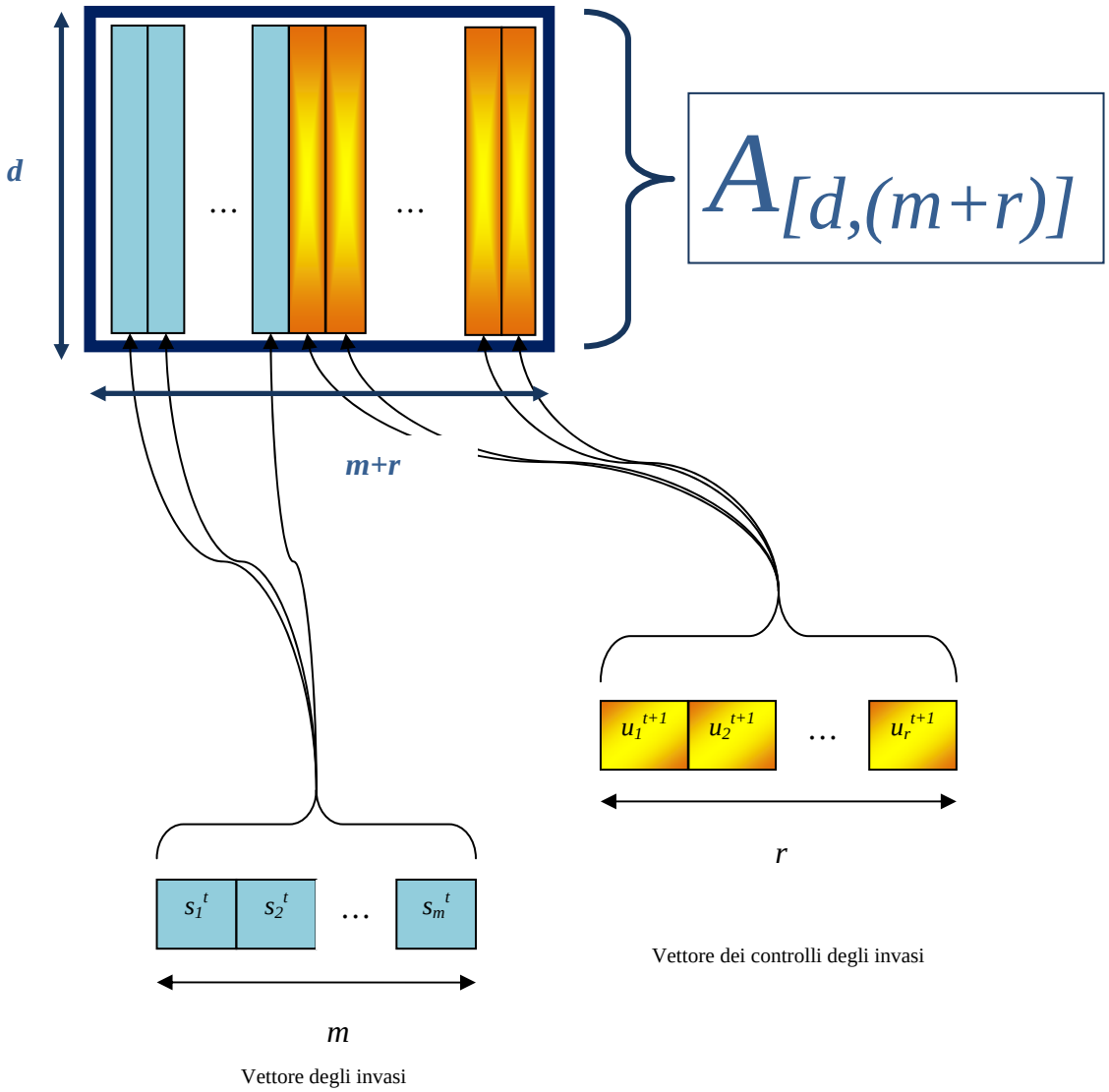

Figura 3.5 – Struttura della matrice di addestramento nel caso più semplice, in cui si considerano i soli volumi di invaso (e i controlli di ottimizzazione lineare per la valutazione delle prestazioni)

Questa sarà la struttura che si adotterà inizialmente, e se si otterranno già dei buoni risultati, non sarà ulteriormente complicata. La matrice di addestramento così ottenuta sarà $A_{[(m+r),d]}$, dove ognuna delle colonne *m* è costituita da un solo vettore, quello di invaso.
Qualora, però, le prestazioni ottenute da reti addestrate con la semplice matrice introdotta nella Figura 3.5 non fossero soddisfacenti, occorre aggiungere, tra i dati di input, gli afflussi noti al tempo *t*, costruendo quindi una matrice come quella che si ottiene nella seguente Figura 3.6.

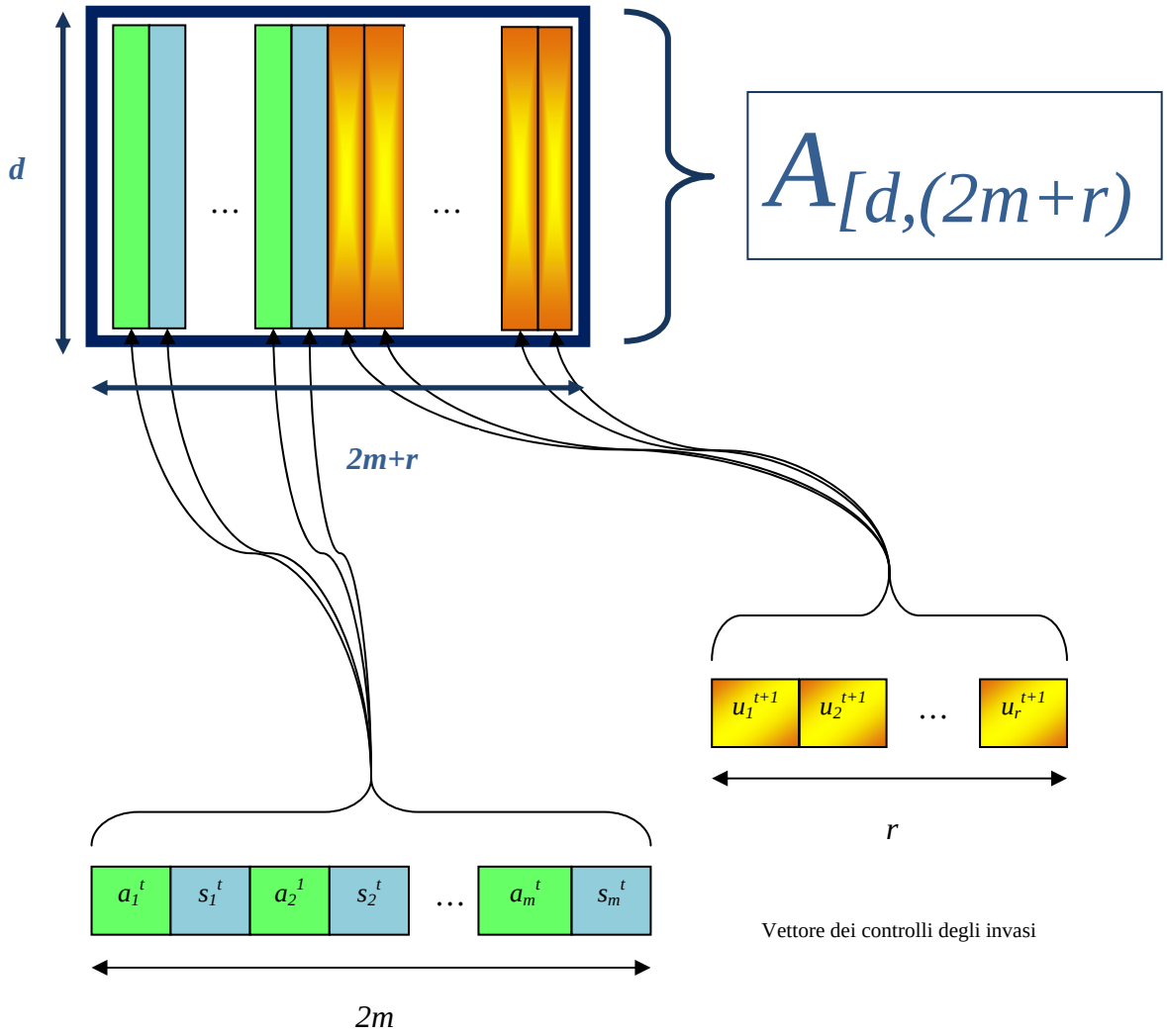

Figura 3.6 – Matrice di addestramento con invasi ed afflussi relativi allo stesso periodo, t

Operativamente, viene costruita un'unica matrice di addestramento (quella generale), da cui poi si estraggono, volta per volta, i dati che interessano.

3.4 ADDESTRAMENTO E VALUTAZIONE DELLA RETE

Al fine di valutare la prestazione ottenuta dalla rete neurale con una certa architettura ed assegnati valori dei pesi, si confronta la matrice dei controlli prodotti dalla rete con quella prodotta dall'ottimizzatore lineare Aquafun. I valori di quest'ultimo saranno definiti *osservazioni*, indicandone il vettore con il simbolo $\bar{R}$, mentre le uscite della rete saranno le *stime*, indicate con $\hat{O}$. La valutazione di queste stime si effettua, semplicemente, ricercando il valore dei parametri che rende minimo l'errore quadratico medio della differenze tra i due valori (stima ai *minimi quadrati*, *MSE*), calcolabile come

$$MSE = E_o = \sum_{i=1}^{r} \left(\hat{O}_i - \bar{R}_i\right)^2 \qquad (3.3),$$

in cui ovviamente *r* è il numero degli elementi della matrice di output e *i* indica il singolo elemento di questa.
Applicando l'algoritmo di retro propagazione dell'errore oppure il metodo di stima di Levenberg e Marquardt (Paragrafi 2.5.3 e 2.5.4), è possibile, in modo automatico, valutare quanto si discosta la stima dall'osservazione, e modificare, aggiornandoli, i valori dei parametri, al fine proprio di ridurre il valore *MSE* calcolato con l'equazione (3. 3).
Il grosso vantaggio nell'addestrare una rete con questa procedura è l'automatismo, determinato dal fatto che in input alla rete si forniscono già, per come è stata costruita la matrice *A* di addestramento, i valori delle osservazioni, per cui l'aggiornamento dei parametri in modo iterativo viene effettuato rapidamente dal software che implementa la rete neurale stessa.
Può accadere, tuttavia, che una determinata architettura non produca un addestramento soddisfacente (tipicamente quando gli errori *MSE* si mantengono troppo alti). In questo caso bisogna procedere con una modifica della struttura della Rete Neurale, e ripetere la .procedura di training. La modifica strutturale alla rete può essere eseguita solamente dall'Analista, il quale, di solito, o cambia il numero di neuroni, o modifica gli strati nascosti.
Aumentare il numero degli strati nascosti appesantisce molto la struttura della rete neurale, rendendo anche molto pesante sia l'addestramento, sia l'utilizzo del modello. Si tende, pertanto, ad aumentare la quantità di neuroni mantenendo costante il numero di stati interni, dal momento che, di solito, i neuroni con cui si parte sono troppo pochi per riuscire ad addestrare correttamente la rete con i dati a disposizione.
Ovviamente, quanto più si riesce a rendere automatica la procedura, tanto maggiori saranno i vantaggi conseguenti: facendo ricorso al software Matlab, o all'analogo Octave su piattaforma Cygwin, si riesce ad implementare automaticamente l'esecuzione di Relax e la valutazione delle prestazioni della rete unitamente all'addestramento del modello stesso, riducendo considerevolmente il tempo di calcolo.

4

INTRODUZIONE AL BACINO DELLO ZAMBESI

Nel capitolo si descrive il bacino del Fiume Zambesi, tracciandone le linee generali geografiche e socio – economiche. Si dettagliano maggiormente alcune zone di interesse. Si descrivono il modello adottato e le sue caratteristiche peculiari, in accordo con le possibilità offerte dai software utilizzati per l'ottimizzazione.

4.1 IL FIUME ZAMBESI

4.1.1 INTRODUZIONE

Il Fiume Zambesi, il quarto di tutta l'Africa, è lungo *2574* km, e dalle sorgenti in Zambia (vicino a Mwinilunga, a circa *1500* m di quota sul livello medio del mare, si snoda tra Angola, Botswana, Malawi, Namibia, Tanzania, Zambia, Zimbabwe e Mozambico, dove si getta nell'Oceano Indiano (il fiume è il principale, per dimensioni, che si getta in questo Oceano). Il suo bacino idrografico, benché abbia un'estensione di *1500000* km^2, è pari a circa la metà di quello del Nilo, il primo fiume dell'Africa. Dal punto di vista geografico, i confini del bacino sono individuati tra *24*° e *38*° E e tra *12*° e *20*° S (Figura 4.1 a pagina seguente).
Ciò che si nota subito osservando la figura, è la vastità del territorio del bacino e la sua "estensione" su più Stati, in una realtà sicuramente non facile, quale è quella africana.
Di interesse, non solo turistico ma anche ai fini idroelettrici, necessari quindi alla popolazione per vivere, sono le Cascate Vittoria (in Zambia), tra le più grandi del mondo. Esistono poi altre cascate di notevole interesse, anche se forse meno famose delle Vittoria: sono le Cascate Ngonye (sempre in Zambia Occidentale), e Chavuma (che segnano una porzione del confine tra lo Zambia e l'Angola).
Climatologicamente, la parte settentrionale del bacino ha un clima tipicamente tropicale, caratterizzato da mesi estivi molto piovosi, fino a *1400* mm annui, mentre la parte settentrionale è più arida, e le precipitazioni scendono a *700* mm annui. Collocandosi nell'emisfero australe, i mesi estivi piovosi si collocano tra Novembre ed Aprile (si veda Capitolo 5).
Infine, una curiosità: nonostante il fiume sia lungo più di *2500* km, è attraversato da soli cinque ponti, ad indicare la vastità dei territori attraversati e la contemporanea scarsità di urbanizzazione che si incontra lungo il suo corso.

Figura 4.1 – Il bacino del Fiume Zambesi (rielaborazione personale da Google Earth)

Il fiume è utilizzato a scopi idroelettrici, e in effetti la ricerca della politica ottima di gestione degli impianti è lo scopo di questo lavoro di Tesi. Le principali centrali sono due, una sulla Diga di Kariba, che fornisce energia sia allo Zambia sia allo Stato dello Zimbabwe, e la seconda sulla Diga di Cabora Bassa, per la fornitura di energia alla Repubblica Sudafricana. Sulle cascate Vittoria, infine, esiste una più piccola centrale, che, tuttavia data la sua dimensione, esula dall'analisi che qui ci si propone di condurre.

4.1.2 IL CORSO DEL FIUME

Il corso del Fiume Zambesi può essere suddiviso in tre parti, oltre la sorgente: lo Zambesi Superiore (*Upper Zambesi*), fino alle Cascate Vittoria, lo Zambesi Medio (*Middle Zambesi*), dalle Cascate Vittoria fino alle rapide di Cabora Bassa, e Basso Zambesi (*Lower Zambesi*), da queste rapide fino alla foce nell'Oceano Indiano.

La sorgente

Nello Zambia Nord – Occidentale, a circa *1500* m di quota, in mezzo alle foreste, da una zona paludosa si diparte un corso d'acqua. Esiste uno spartiacque naturale ad Est, tra il bacino del Fiume Congo e quello dello Zambesi, costituito da una fascia montagnosa che declina bruscamente in altezza sia a nord sia a sud, ma si estende longitudinalmente tra Est ed Ovest, tra le latitudini *11*° S e *12*° S.

Lo Zambesi superiore

Dopo aver percorso circa *240* km in direzione sud - ovest, il fiume Zambesi devia verso sud per andare ad unirsi a molti altri affluenti. Nei pressi della città di Kakengi, il fiume subisce un allargamento (da *100* a *350* m) per andare a formare delle rapide, che a sud della città terminano nelle Cascate Chavuma (una panoramica è rappresentata nella seguente Figura 4.2).

Figura 4.2 – Le Cascate Chavuma sul fiume Zambesi (da www.panoramio.com, come le successive)

Ancora in Zambia, il fiume diventa pianeggiante e si entra in una regione dove le inondazioni regolarmente allagano il terreno. Proseguendo, il fiume piega verso sud - est, dove per un lungo tratto non si hanno affluenti dalla sponda orientale. Subito prima di incontrare un grosso affluente, il Cuando, lo Zambesi forma le Cascate Ngonve, e poi delle rapide.
Grazie alla confluenza con il Cuando, il letto del fiume si allarga molto, si fa meno profondo e la velocità delle acque si abbassa, preparando, mentre vira ad est, il salto delle Cascate Vittoria (una suggestiva immagine è rappresentata nella seguente Figura 4.3, mentre alcuni dati tecnici si possono ritrovare nella Tabella 4.1) , prima di gettarsi nell'Altopiano Centrale Africano.

Figura 4.3 – Le Cascate Vittoria (fonte citata)

ALTEZZA SALTO [m]	107
PERCORSO TOTALE [m]	1737
PORTATA MEDIA [m^3/s]	1088
PORTATA DI SFIORO [m^3/s]	7079

Tabella 4.1 – Dati tecnici delle Cascate Vittoria (elaborazione personale dal Sito Web/www.world-waterfalls.com/home.php)

È da notare come, dopo un salto iniziale che porta il livello dai *1500* m della sorgente ai *1100* m nei pressi della città di Kakengi (*350* km dopo la sorgente), la quota si mantiene pressappoco costante, dal momento che fino alle Cascate Vittoria si scende di soli *180* metri.

Il medio Zambesi

Dopo le Cascate Vittoria si parla di Medio Zambesi. Qui il fiume si incanala in strette gole basaltiche alte *250* metri circa, larghe tra *20* e *60* metri, che causano la formazione di rapide ed un dislivello di altri *250* metri.
Senza dubbio di interesse per questo lavoro è menzionare il Lago Kariba - anche se sarebbe più opportuno definirlo serbatoio, dal momento che è regolato -, che nacque nel *1959* a seguito del completamento della Diga Kariba (Figure 4.4 e 4.5, rispettivamente per la diga ed il lago). Grazie a questo, viene prodotta energia idroelettrica per lo Zambia e lo Zimbabwe. Superato il salto della diga, il fiume piega verso nord, fino alla confluenza con il fiume Kafue, dove vira ad est. Qui, dopo essersi unito al Luagwa, il fiume Zambesi si getta nel Lago Cahora Bassa (Cabora Bassa, una cui fotografia da satellite è visibile nella seguente Figura 4.6), dove ufficialmente si fa terminare lo Zambesi Medio. Anche il lago Cahora è un serbatoio, dal momento che viene regolato da una diga (Cabora Dam, in Figura 4.7), la cui costruzione è stata ultimata nel *1974*.

Figura 4.4 – La Diga di Kariba (da www.panoramio.com/photos/original/1437950.jpg)

Figura 4.5 – Il Lago di Kariba (fonte citata)

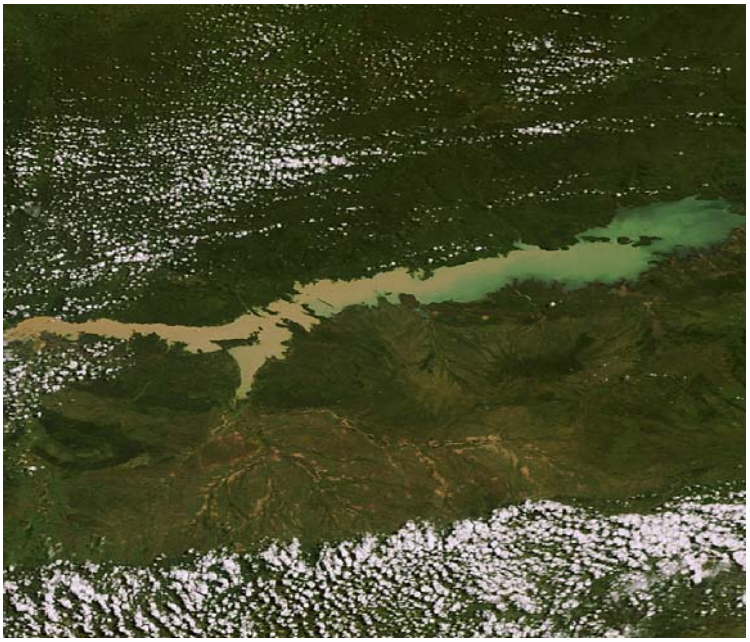

Figura 4.6 – Foto da satellite del Lago Cahora (fonte citata)

Figura 4.7 – La Diga di Cahora (fonte citata)

Zambesi inferiore

È questo l'ultimo tratto del fiume, il cui inizio si fa coincidere con l'uscita dal Lago Cabora e il termine corrisponde alla foce nell'Oceano Indiano, per una lunghezza di circa *650* km. Questa parte è piuttosto pianeggiante, pertanto è navigabile, anche se, essendo soggette alla variabilità tipica delle zone tropicali, con una stagione secca ed una umida, le acque possono, durante appunto la stagione secca, essere molto basse.

Dopo aver attraversato la Gola Lupata, il fiume si allarga smisuratamente, arrivando ad un'estensione che può raggiungere anche gli *8* km. Relativamente vicino alla foce, il fiume si divide in molte ramificazioni (*Milambe*, *Kongone*, *Luabo*, *Timbwe*, normalmente ostruite da banchi sabbiosi), che si riuniscono durante la stagione delle piogge, quando le acque si rialzano.

A *150* km dalla foce c'è l'incontro con l'ultimo affluente, il fiume Shire, che reca le acque del Lago Malawi.

A conclusione di questa breve descrizione introduttiva, è bene sottolineare come, in seguito alla costruzione delle dighe di Kariba e Cabora Bassa, il delta si sia ridotto di circa la metà.

4.1.3 ASPETTI BIOLOGICI

Il Fiume Zambesi, soprattutto nelle vicinanze del delta, ospita moltissime specie animali, tra cui ippopotami, coccodrilli, varani, ma anche specie avicole come aironi, pellicani ed altre.

È da notare come la vegetazione boscosa presente lungo le sponde permette a specie di grosse dimensioni di proliferare: tra queste occorre annoverare bufali, zebre, giraffe ed elefanti.

È da segnalare, tuttavia, che la costruzione delle dighe di Kariba e Cabora Bassa ha modificato anche notevolmente la climatologia del luogo, diminuendo progressivamente il numero delle alluvioni, causando quindi una riduzione - talvolta anche considerevole - delle dimensioni degli habitat del luogo, e, come naturale conseguenza, una riduzione delle popolazioni di grandi animali.

Meno soggetti ai cambiamenti climatologici sono le specie ittiche, che si mantengono presenti in grosse quantità nel fiume, ma anche nei laghi interni.

4.1.4 CENNI DI ECONOMIA

Su tutto il bacino del fiume Zambesi gravita una popolazione non molto numerosa, per l'estensione della zona: si tratta infatti di circa *32* milioni di persone, su un'area di *1500000* km^2 (un veloce confronto, abbastanza significativo dal punto di vista numerico, può essere fatto con l'Italia, dove su una superficie di *301338* km^2, vivono poco più di *60* milioni di abitanti.

Quindi in Italia la densità è $199{,}112\ \frac{ab}{km^2}$ mentre nel bacino dello Zambesi è $13{,}333\ \frac{ab}{km^2}$.

La quasi totalità della popolazione (*80*% circa) vive intorno al fiume, dal quale ricava l'acqua necessaria per l'attività agricola, la principale fonte di reddito locale. Accanto ad essa, è abbastanza sviluppata la pesca, soprattutto intensiva, che fornisce reddito anche per chi non vive nei pressi del fiume, ma viene da molto lontano per praticarla. Intorno alle città di Mongu e Livingstone (si veda la seguente Figura 4.8), si pratica la pesca sportiva e turistica, per specie esotiche che vengono oltretutto esportate negli acquari.

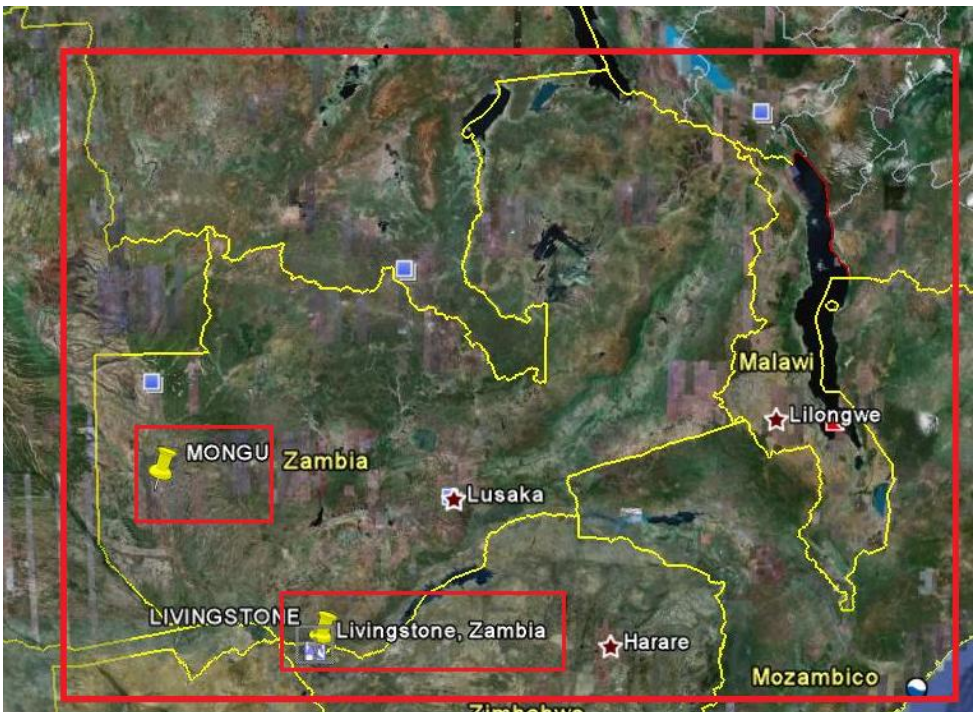

Figura 4.8 –Localizzazione delle città di Mongu e Livingstone in Zambia (rielaborazione personale da Google Earth)

Non esistono tuttavia solo queste attività: la zona è infatti molto ricca di giacimenti minerari, di combustibili fossili e di carbone, che danno origine alle attività estrattive.
Buona parte della popolazione è inoltre occupata nella manutenzione delle dighe e degli impianti idroelettrici presenti nel bacino.
Infine, non manca il turismo: oltre alle Cascate Vittoria, vengono visitati il Lago Kariba e le Paludi di Mana.

4.1.5 TRASPORTI

Nonostante sia un collegamento molto importante tra i diversi paesi del bacino, il corso del fiume è spesso costituito da rapide, per cui non è possibile effettuare lunghe traversate, ma può essere attraversato sia a piedi sia in automobile, tramite traghetti, in moltissimi punti. Tuttavia, dal momento che le zone interne vengono inondate assai facilmente, per brevi tratti il fiume è da preferirsi rispetto alle strade di terra. Lo Zambesi rappresenta poi l'unico punto di accesso per moltissimi villaggi, che possono essere raggiunti solamente tramite imbarcazioni.
Come detto in precedenza, nonostante la sua lunghezza, lo Zambesi è attraversato da soli cinque ponti, uno dei quali può essere percorso solamente a piedi. Il principale e più antico ponte (del *1905*, lungo *250* metri ad un'altezza di *125* metri) è quello sulle cascate Vittoria (una foto è riportata nella seguente Figura 4.9), che fu in origine pensato come parte di una ferrovia, mai realizzata (Cairo – Città del Capo).

Figura 4.9 – Il ponte sulle Cascate Vittoria (da www.wikipedia.it)

4.1.6 IL COMPLESSO IDROELETTRICO

Il Fiume Zambesi offre numerose risorse alle popolazioni che vivono intorno ad esso, ma l'uso principale delle sue acque è senza dubbio quello idroelettrico, ovverosia la produzione di energia per soddisfare il fabbisogno di energia elettrica delle nazioni che si affacciano sul suo bacino.
A tale proposito, è bene, a questo punto riepilogare come si articola l'insieme degli impianti per la produzione di energia idroelettrica che sfruttano le acque dello Zambesi. Per inquadrare il sistema si faccia riferimento alle seguenti Figure 4.10 – 4.11. Nella prima immagine si ha un quadro d'insieme del bacino, con l'indicazione delle principali infrastrutture ed attrazioni che vi sono ubicate, mentre nella seconda immagine si possono vedere dove sono localizzati i principali impianti per la produzione idroelettrica.

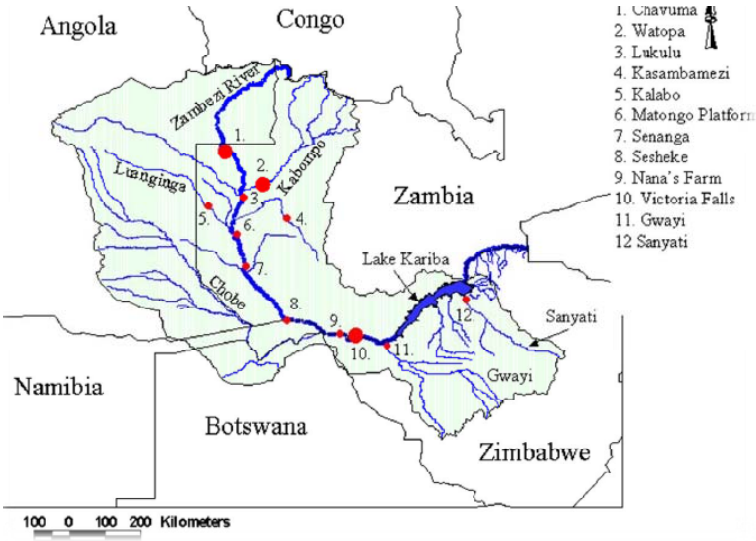

Figura 4.10 – Visione d'insieme del bacino dello Zambesi con le principali risorse/infrastrutture (dal Sito Internet della ZWA)

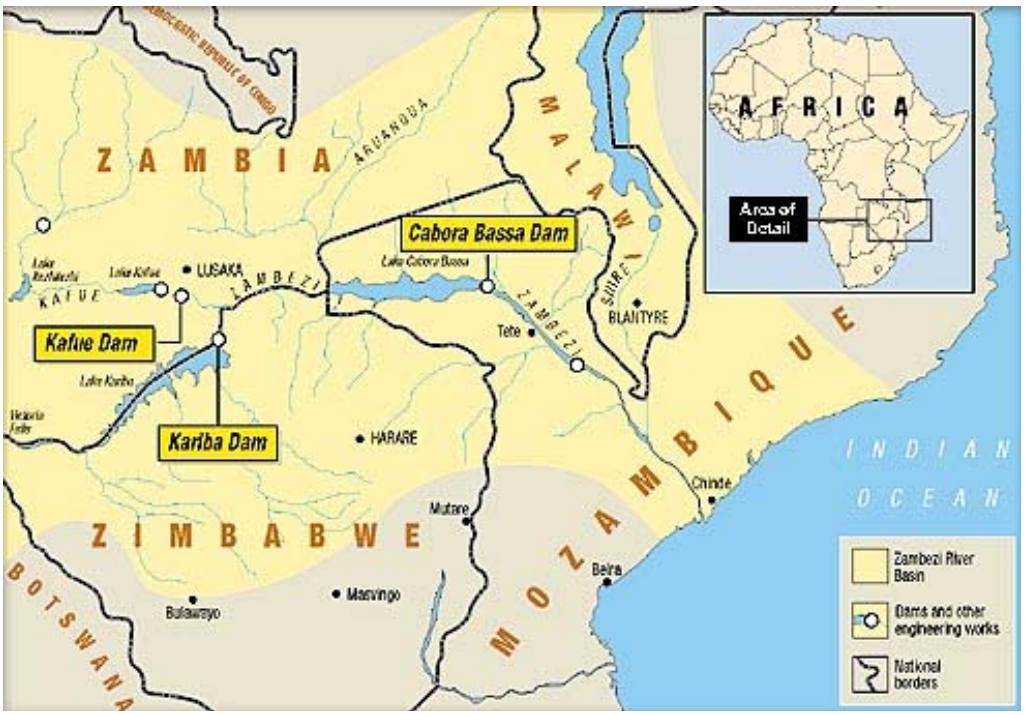

Figura 4.11 – Mappa dei principali bacini (dal Sito Internet della ZWA)

Di seguito si riportano, per ogni nazione, i principali sbarramenti e le centrali per la produzione di energia elettrica associate. L'ordine di presentazione segue l'andamento geografico del fiume.

Malawi

In questa nazione si trova l'unico lago naturale della regione, il Lago Nyassa, lungo il corso del fiume Shire, a valle del quale si trovano tre impianti: Nkula Falls A con una potenza di *24* MW, Nkula Falls B con una potenza di *80* MW e, a valle rispetto a questi, Tedzani Falls con *40* MW installati. Questi impianti, tuttavia, pur facendo parte del sistema, non saranno nel seguito considerati perché di potenza decisamente modesta rispetto al totale installato.

Zambia

A valle della Diga di Kariba è ubicato il primo impianto, denominato North Bank, capace di *600* MW; il secondo impianto si trova a monte di Victoria Falls, che funziona grazie ad una centrale ad acqua fluente. Questo impianto ha una capacità di *108* MW, che, essendo modesto, non sarà considerato per gli scopi di questo lavoro. Infine, l'impianto più importante è situato lungo il corso del Kafue e sfrutta un salto naturale di *400* metri, che, partendo dal piccolo serbatoio di Kafue Gorge, consente di produrre *900* MW.

La scelta di realizzare un piccolo serbatoio a monte di uno con capacità di invaso risponde al requisito di ottimizzazione della risorsa idrica disponibile: in effetti, se si fosse scelto di aumentare l'acqua invasata, si sarebbe inondata una zona paludosa (Kafue Flats), che avrebbe causato un aumento della superficie rispetto al volume, il che, considerando le latitudini cui si fa riferimento (come scritto nel paragrafo 4.1.1), si sarebbe tradotto in un aumento vertiginoso dell'evapotraspirazione. Per garantire comunque una sufficiente riserva idrica è stato realizzato un serbatoio più in alto, denominato Itezhitezhi (Itezhi nel seguito), a monte delle Kafue Flats che ne garantisce il corretto approvvigionamento idrico. Tuttavia, pur essendo riusciti a ridurre l'evapotraspirazione, si va incontro ad un altro inconveniente, ossia il tempo necessario al trasferimento dal serbatoio Itezhitezhi a quello di Kafue, che, dal

momento che occorre attraversare le paludi, è pari a circa due mesi.

Zimbabwe

L'impianto di South Bank è situato sulla sponda del lago Kariba appartenente allo Zimbabwe, ed ha una capacità di *666* MW; contestualmente a questo, con analogo salto motore, opera l'impianto di North Bank (in Zambia), interconnesso al precedente e con una potenza di *600* MW. I due impianti, pertanto, saranno considerati un unico complesso con potenza di *1266* MW.

Mozambico

In questa Nazione si trovano gli impianti tra i più imponenti installati nel bacino dello Zambesi. Si trovano infatti gli impianti di Cabora Bassa, con *2057* MW di potenza installata. Tuttavia, a causa delle continuative e ripetute controversie politiche nella regione, che hanno portato ad una vera e propria guerra civile, la produzione effettiva realizzata è sempre stata molto inferiore, irrisoria rispetto alla capacità. Lungo questa direzione esistono notevoli capacità di sviluppo.

Di seguito si riporta uno schema della rete che si analizza (seguente Figura 4.12), ed una tabella (Tabella 4.2) che riassume le localizzazioni dei serbatoi e delle relative centrali idroelettriche.
Nella successiva Tabella 4.3 si riportano, invece, le caratteristiche tecniche degli impianti principali sullo Zambesi.

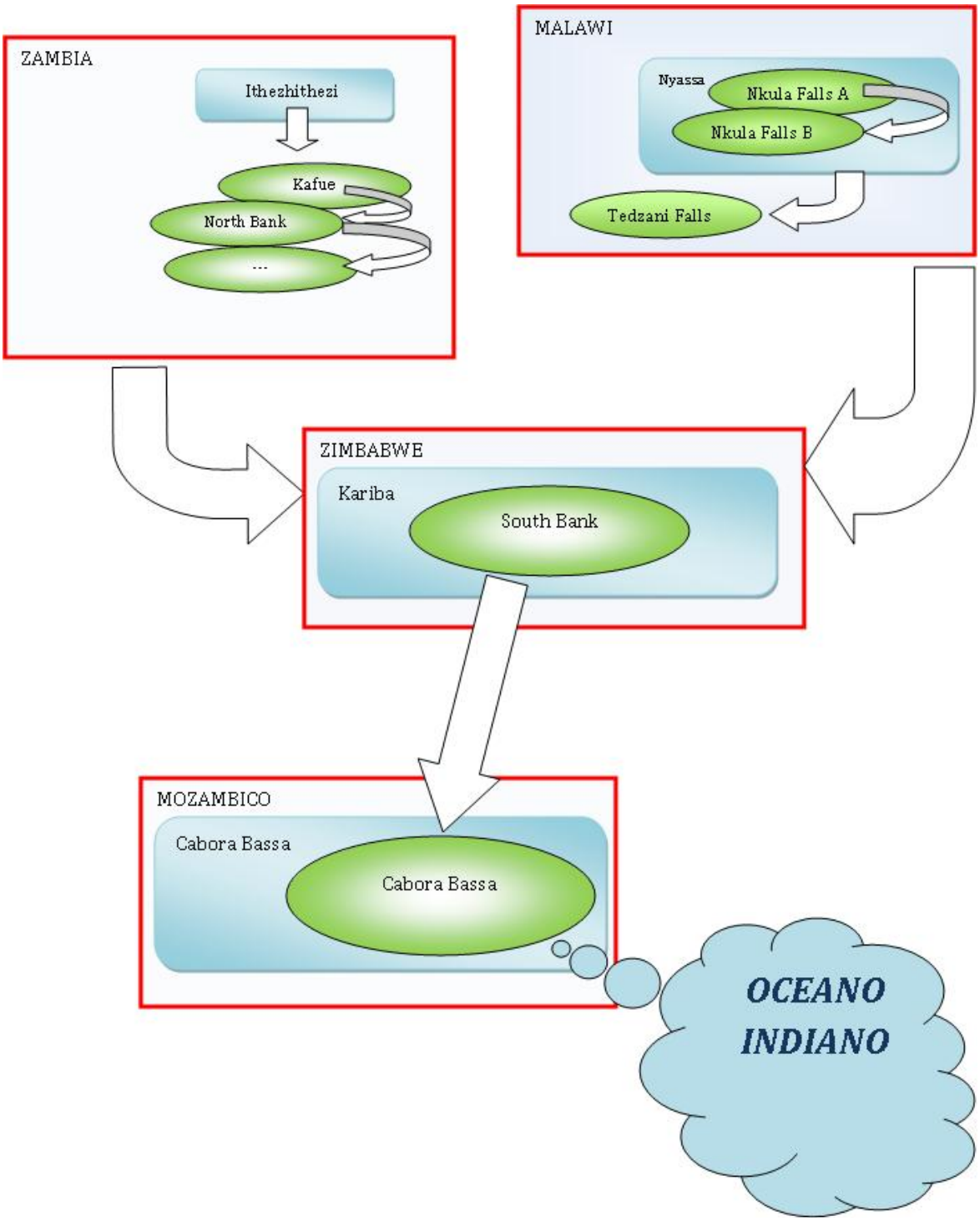

Figura 4.12 – Schema topologico della rete analizzata (gli ovali individuano gli impianti, i rettangoli a bordo smussato i serbatoi, infine i rettangoli a bordo rosso le nazioni di appartenenza)

STATO	SERBATOIO	IMPIANTO
Malawi	Nyassa (lago)	Nkula Falls A
		Nkula Falls B
	-	Tedzani Falls
Zambia	Itezhitezhi	-
	Kafue Gorge	Kafue
		North Bank
		-no name-
Zimbabwe	Kariba	South Bank
Mozambico	Cabora Bassa	Cabora Bassa
POTENZA TOTALE EROGATA [MW]		4475
POTENZA TOTALE CONSIDERATA [MW]		*4223*

Tabella 4.2 – Caratteristiche delle centrali idroelettriche incontrate (le caselle evidenziate in grigio non sono prese in considerazione nella'analisi del sistema perché troppo poco significative)

	SERBATOI			
	ITHEZHITEZHI	KAFUE GORGE	KARIBA	CABORA BASSA
CAPACITA' RILASCIO[1] [m^3/s]	4200	4250	9500	13950
CAPACITA' STANDARD [10^9 m^3]	5	0,7	44	60
CAPACITA' MINIMA [10^9 m^3]	-	-	116	12,5
LIVELLO MASSIMO [m.a.s.l.]	1029,5	976,6	488,5	330,5
LIVELLO MINIMO [m.a.s.l.]	1006	972	475,5	295

	IMPIANTI				
			NORTH BANK	SOUTH BANK	
NUMERO DI TURBINE	-	6	4	6	5
POTENZA INSTALLATA[2] [MW]	-	150	150	111	415
PORTATA TURBINABILE[3] [m3/s]	-	42	200	140	452
MASSIMO SALTO UTILE [m]	-	397	108	110	128

[1] E' da intendersi l'insieme di paratie più sfioratori
[2] [3] Da intendersi per singola turbina

Tabella 4.3 – Dettaglio delle caratteristiche tecniche degli impianti installati nel Bacino del Fiume Zambesi

4.1.7 DETTAGLIO DELLE SEZIONI SIGNIFICATIVE

Nella seguente Figura 4.13 si riporta una visione d'insieme della regione dello Zambesi, nella quale è possibile individuare chiaramente i confini di Stato (**-----**), la rete idrica (linee nere continue), e le zone paludose (linee sovrapposte trasversalmente).

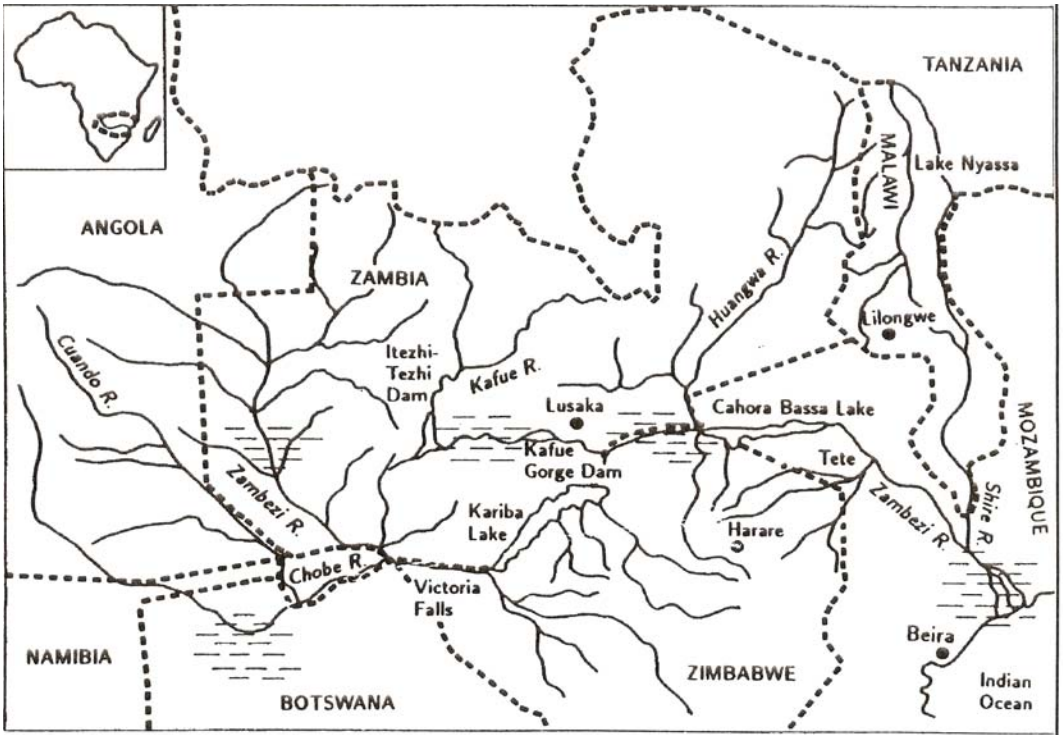

Figura 4.13 – La regione dello Zambesi, con in evidenza i confini di Stato, la rete idrica principale e le zone paludose, di cui l'area è molto ricca (da Taddio, Togni, op. cit.*)*

Di seguito si individuano le principali caratteristiche di ognuna delle macrosezioni che compongono il territorio del fiume Zambesi.

La sezione di Kafue

Il Fiume Kafue costituisce uno dei principali affluenti allo Zambesi, con un bacino, interamente situato nel territorio dello Zambia, pari a *155000* km^2.

Idrologicamente il bacino si compone di tre parti principali, la prima delle quali, che si estende dalla Regione mineraria del Copperbelt fino al serbatoio di Itezhitezhi (pari a circa *106000* km^2, ovvero il *68,4*% dell'intera area), è molto significativa per gli afflussi, essendo qui le precipitazioni particolarmente intense da dicembre a marzo. Il serbatoio di Itezhitezhi è molto piccolo, ed ha l'unico scopo di alimentare il sottostante bacino di Kafue che ha una capacità limitata per poter sfruttare interamente il salto motore di *400* m. Un evidente svantaggio del serbatoio di controllo di Itezhitezhi è che i flussi idrici uscenti da esso giungono al serbatoio principale di Kafue con due mesi di ritardo, ma è possibile determinare, mediante dati registrati nelle stazioni, delle serie di afflussi al primo invaso. Nella seguente Figura 4.14 si riporta il dettaglio della prima zona della regione di Kafue.

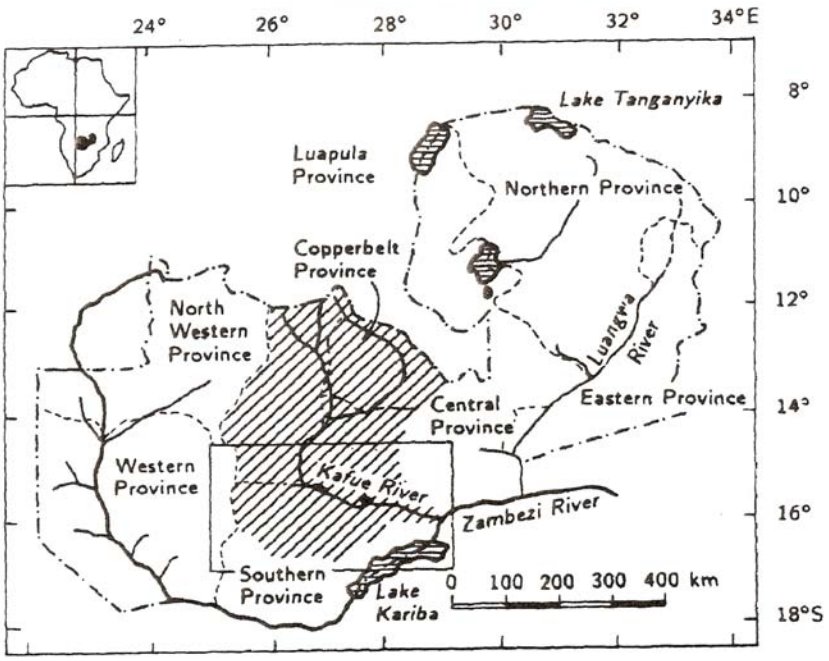

Figura 4.14 – La sezione di Kafue dalla Provincia del Copperbelt al serbatoio di Itezhitezhi (da Taddio, Togni, op. cit.*)*

La seconda zona, estesa per 45000 km^2 (29%) parte da Itezhitezhi e raggiunge il serbatoio di Kafue Gorge, formando una zona paludosa denominata Kafue Flats. In quest'area si ha un'elevata evapotraspirazione, che genera una perdita di acqua troppo alta se il serbatoio di Kafue viene completamente riempito. La pronunciata evaporazione rende di fatto impossibile stimare gli afflussi alla zona, dovuti agli affluenti minori, e pertanto questi vengono stimati partendo da quelli di Itezhitezhi, assumendo uguali i coefficienti di deflusso. Nella figura 4.15, che segue, un dettaglio dell'area.

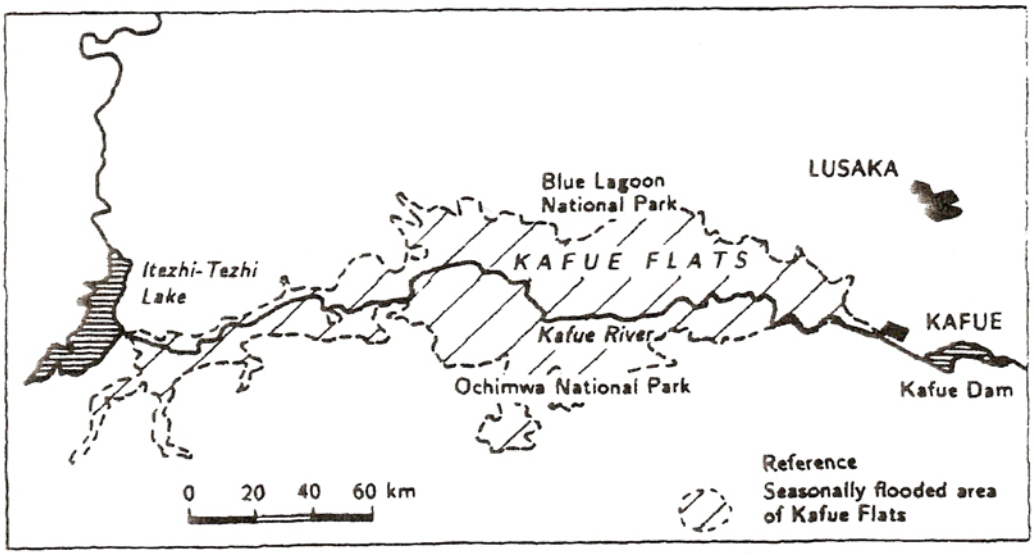

Figura 4.15 – La parte meridionale della sezione di Kafue, con il lago Kafue e le "Kafue Flats" (da Taddio, Togni, op. cit.*)*

Oltre ai problemi prettamente idrologici, le zone paludose di Kafue Flat presentano aspetti biologici ed ecologici di cui non si può non tenere conto: infatti gli ecosistemi terrestri ed acquatici ivi presenti sono altamente correlati, per i cicli di inondazione e siccità. Quando si è nella stagione secca, gli animali si nutrono della vegetazione che affiora dalle acque, mentre essa ricresce rapidamente durante le inondazioni. In questo

periodo, altresì, molti pesci si riproducono, trovando nelle zone alluvionate un habitat più esteso.
Si evince pertanto come non sia possibile trascurare gli aspetti ambientali (così importanti tali da aver richiesto un deflusso minimo vitale – *DMV* – da Itezhitezhi, pari a *300* m^3/s e *15* m^3/s da Kafue Gorge), anche se essi sono spesso in contrasto con l'obiettivo di massimizzazione della produzione di energia elettrica. Il fatto, anch'esso non trascurabile, che la sussistenza delle popolazioni della zona sia dovuta prevalentemente alle attività di pesca, chiarisce come sia necessaria una nuova politica di gestione del bacino, politica che si cerca tramite l'utilizzo delle reti neurali nel presente lavoro di Tesi.

La sezione di Kariba

La sezione di Kariba interessa una superficie pari a circa un terzo di tutta l'area complessiva, dunque si possono riconoscere almeno due grosse macro – aree: il *bacino superiore*, dalle sorgenti fino alle Cascate Vittoria, e il *bacino inferiore*, dalle cascate fino a Kariba, una cui visione d'insieme è offerta nella Figura 4.16.
Nella parte superiore del bacino, che si estende su una superficie di circa *507200* km^2, non sono presenti affluenti o apporti particolari di acqua, pertanto è relativamente semplice ricostruire serie di portate che possano essere indicative dell'andamento dei regimi idrici. Si deve inoltre tenere conto del fatto che il ritardo del trasporto idrico da una sezione alla successiva è trascurabile. Differente è il discorso per quanto riguarda la parte inferiore del bacino (che insiste su un'area di circa *156000* km^2), dove ci sono numerosi affluenti secondari e corsi d'acqua, che, spesso, privi di stazioni di misura, sfociano direttamente nel lago e quindi rendono impossibile una valutazione degli afflussi. Il modo più semplice ed immediato, anche se conduce ad inevitabili approssimazioni, per calcolare tali afflussi, consiste nell'applicare l'equazione di continuità del serbatoio.
È da ricordare la presenza della zona paludosa, a valle di Kariba, denominata Mana Pools, che segue le medesime ciclicità biologiche di Kafue Flats.

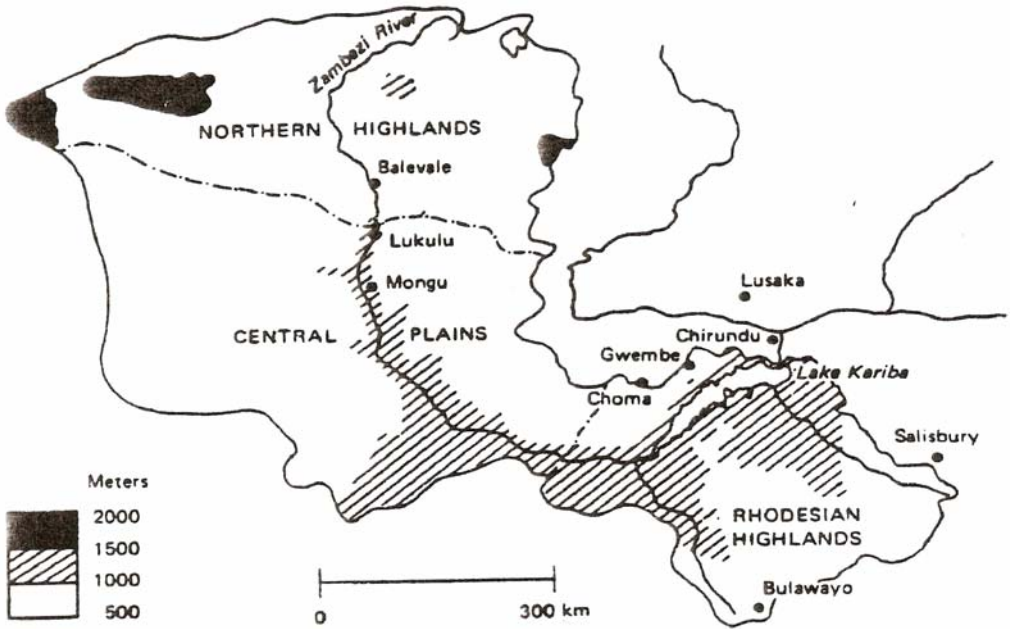

Figura 4.16 – Visione d'insieme della sezione di Kariba (da Taddio, Togni, op. cit.*)*

La sezione Cabora Bassa

Il serbatoio di Cabora Bassa riceve le acque sia da Kariba sia da Kafue Gorge, pertanto dispone di un bacino di dimensioni assai considerevoli, quantificabili in *11000000 km*^{2}. Il ritardo temporale del deflusso idrico tra Chirundu e Cabora è non superiore ai quattro – cinque giorni, pertanto ampiamente trascurabile se si considerano, come nel caso qui studiato, dati mensili. Anche in questa sezione (una cui visione d'insieme è offerta nella Figura 4.17) è presente un piccolo affluente (Huangwa), che rende difficile calcolare gli afflussi, dal momento che non esistono stazioni di rilevamento.

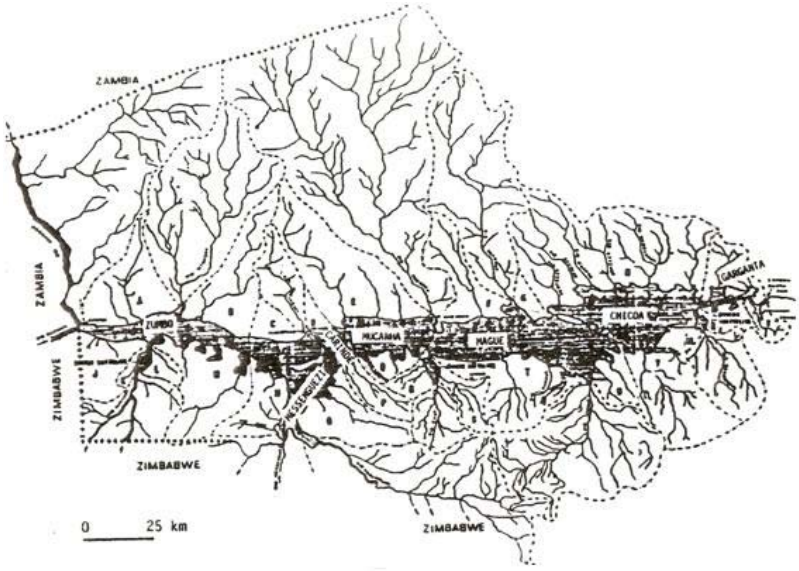

Figura 4.17- Visione d'insieme dell'area di Cabora Bassa, sezione terminale dello Zambesi (da Taddio, Togni, op. cit.*)*

4.2 L'AUTORITÀ DI BACINO (*ZRA*)

4.2.1 DESCRIZIONE

La gestione degli aspetti idroelettrici spetta ad un'apposita Autorità di Bacino (*Zambesi River Authority – ZRA*), costituita collegialmente dal Ministro delle Finanze e dal Ministro dell'Energia della Repubblica di Zambia e della Repubblica di Zimbabwe, per un totale, dunque, di quattro persone.
Tale autorità venne istituita ufficialmente il *1 ottobre 1987*, come una Corporazione, grazie all'azione congiunta dei due Parlamenti di Zambia e Zimbabwe, in seguito alla ricostituzione dell'Autorità per l'Energia dell'Africa Centrale (*Central African Power Corporation*). Legalmente, l'Autorità appartiene in parti eguali ai due Governi, che ne condividono la gestione.

4.2.2 FUNZIONI TECNICHE

L'Autorità di Bacino (nel seguito, citata come ZRA) è l'organismo che si occupa non solo della gestione idroelettrica degli impianti, ma anche di numerosi altri aspetti che entrano in gioco in un bacino assai esteso quale è quello dello Zambesi. Di seguito viene riportato un elenco in cui sono individuate le principali funzioni riconosciute alla ZRA:

1) Occuparsi del funzionamento, del monitoraggio e della manutenzione del Complesso di Kariba (*Kariba Complex*), dove, con questa dicitura, sono da intendersi:

    - la diga di Kariba (*Kariba Dam*),
    - il serbatoio,
    - le stazioni di rilevamento presenti in loco,
    - ogni altra opera presente ed autorizzata dall'Autorità di Bacino;

2) Valutare la necessità di costruire nuove dighe sul Lago Kariba, e sottoporne il progetto al Consiglio;
3) Dopo aver ricevuto il nulla osta dallo stesso Consiglio, occuparsi della costruzione, del mantenimento e del funzionamento delle altre dighe eventualmente costruite;
4) Raccogliere dati sul funzionamento del sistema sia idrico sia ambientale del Fiume Zambesi, al fine di ricercare una politica di funzionamento migliore di quella attualmente in vigore, e sottoporla all'attenzione degli Stati contraenti;
5) Regolare i livelli del Lago Kariba e di tutti gli altri serbatoi presenti sul bacino;
6) Assicurarsi che le acque in gestione vengano usate nel modo più efficace e più efficiente possibile, al fine di
7) garantire un corretto invio dell'Energia elettrica prodotta ai due Stati contraenti, ovverosia Zambia e Zimbabwe.

4.3 IL MODELLO DELLA RETE FLUVIALE

4.3.1 STRUTTURA DELLA RETE

Prima di descrivere il modello della rete, occorre premettere che, ai fini di semplificare la trattazione, lo schema modellistico adottato per il presente lavoro di Tesi differisce dallo schema del bacino completo, che, sulla base delle descrizioni riportate ai paragrafi precedenti, si struttura secondo quanto riportato nella seguente Figura 4.18.

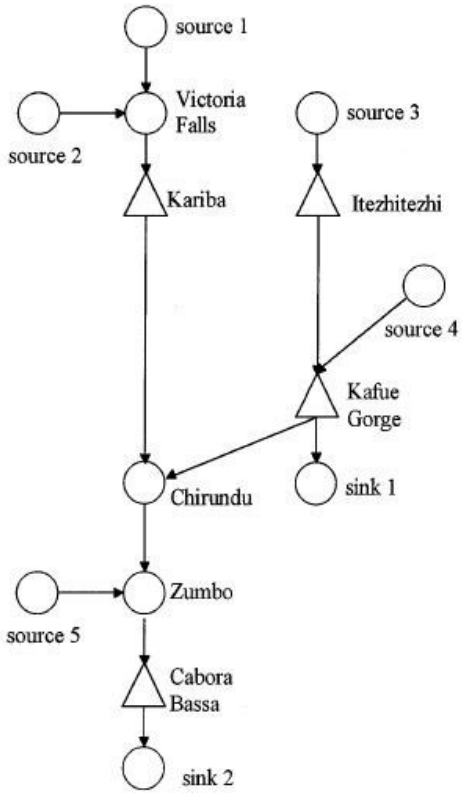

Figura 4.18 – Schema completo della rete del bacino del Fiume Zambesi

Invece, per gli scopi sperimentali che qui ci si propone, il modello è stato notevolmente semplificato.
La rete studiata si compone, nel rispetto sia della fisica del sistema sia della struttura delle reti di serbatoi descritti nel Capitolo 1, di quattro nodi che consentono l'accumulo di risorsa (corrispondenti ai serbatoi: Ithezhitezhi, indicato con *IT*, Kafue, indicato con *KF*, Cabora Bassa, *CB* e Kariba, *KA*), di dieci canali di collegamento tra i nodi, di tre "nodi fittizi" o di passaggio, che rappresentano la sorgente (da dove si generano gli afflussi), il pozzo (ove si raccolgono le uscite dei serbatoi, ed un nodo di confluenza, identificato con il serbatoio di Chirundu (*CH*). In esso si considerano solamente flussi di passaggio, quindi non ci possono essere accumuli di risorsa.

Nel computo dei canali devono essere considerati sia quelli reali, fisicamente esistenti, che collegano i vari serbatoi (*IT – KF*, *KF – CH*, *KA – CH*, *CH – CB*), sia quelli fittizi, che collegano la sorgente ai serbatoi e questi al pozzo (nello specifico, solamente i serbatoi di Kafue e Cabora sono direttamente collegati con il pozzo, perché gli altri due bacini rilasciano flussi che rimangono interni alla rete, essendo immessi rispettivamente in Kafue, se escono da Ithezhitezhi, o in Chirundu, se escono da Kariba). Per rendere più chiara la struttura della rete, se ne riporta il grafo nella seguente Figura 4.19.

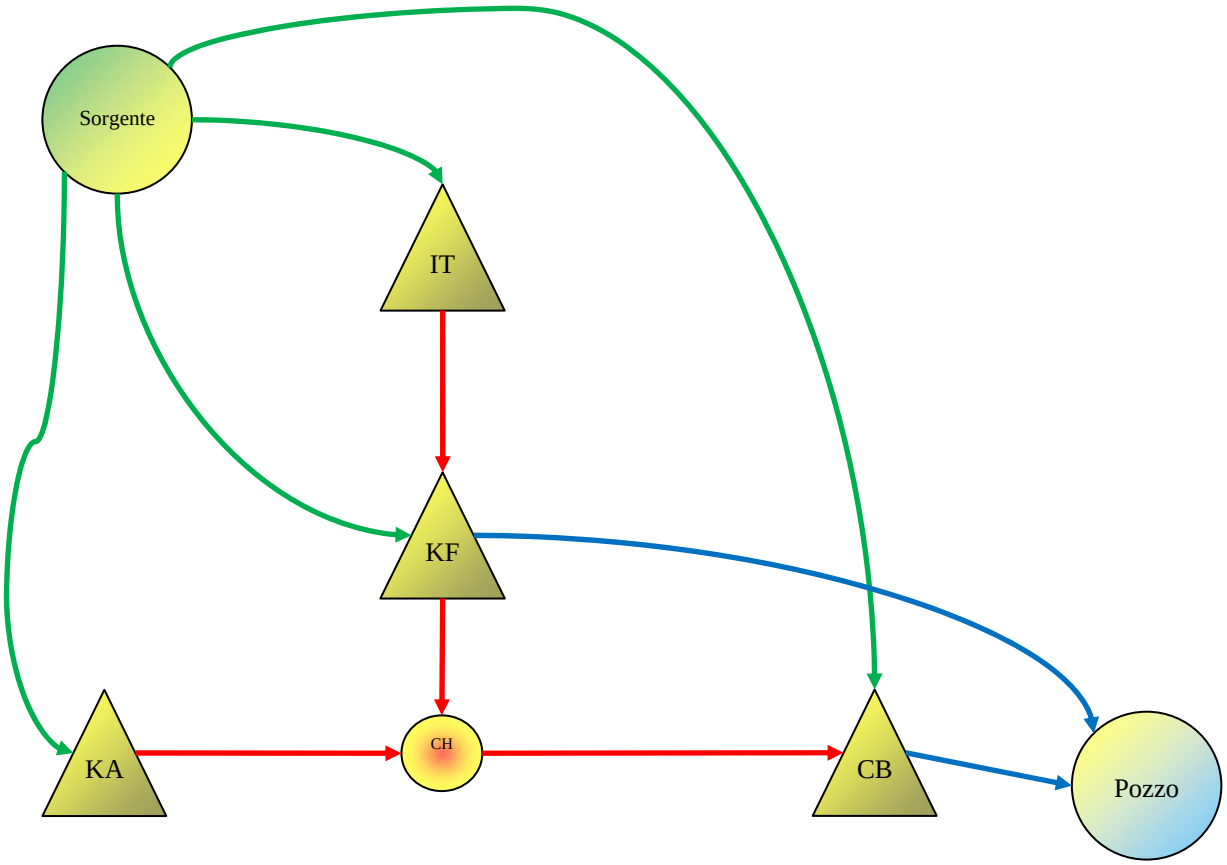

Figura 4.19 – Struttura della rete usata per modellizzare il Bacino del Fiume Zambesi; in verde i canali fittizi che collegano la sorgente ai serbatoi, in rosso i canali reali, in azzurro gli altri canali fittizi di collegamento tra serbatoi è pozzi

La rete, così costituita, può essere implementata nel software di ottimizzazione Aquafun, e può anche essere replicata per tutti i periodi di interesse della simulazione, creando delle "copie" identiche, collegate tra loro da ulteriori canali fittizi temporali (non riportati in Figura 4.18, per non appesantire troppo lo schema).
Le procedure di simulazione devono essere eseguite con numerose serie di dati di ingresso, al fine di tarare e validare correttamente i modelli di ottimizzazione. Per quanto riguarda l'analisi degli input, sia quelli storicamente disponibili (*144* periodi a passo mensile), sia quelli sintetici creati successivamente, si rimanda al successivo Capitolo 5.

4.3.2 APPORTI, PERDITE E RITARDI DI TRASPORTO NELLA RETE

I valori degli afflussi al serbatoio di Cabora ed Ithezhitezhi sono tabulati [Southern Africa Development Coordination Conference], quelli al serbatoio di Kafue sono calcolati in una percentuale del *31%* di quelli di Ithezhitezhi, mentre gli afflussi al serbatoio di Kariba sono ottenuti facendo ricorso a modelli basati su equazioni di bilancio volumetrico. Tuttavia, per gli scopi che interessano il presente lavoro, sarà sufficiente ottenerli per differenza tra il totale mensile e quelli inviati agli altri.
Un altro punto importante riguarda le perdite che interessano i serbatoi della rete. Queste perdite derivano sostanzialmente dall'evaporazione, che viene definita come funzione di un assegnato tasso medio mensile, indipendente dalla reale superficie del lago nel periodo considerato (ipotesi assai semplificativa, ma dovuta alle limitazioni del software Aquafun), ma legato alle oscillazioni delle altezze del pelo libero e dei volumi idrici contenuti.
I tassi netti medi di evaporazione, identificati con $\eta_b^{(t)}$, sono ottenuti per differenza tra il tasso di evaporazione effettivo e la precipitazione sul lago, e sono riportati nella seguente Tabella 4.4.

periodo	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*
[mese]	O	N	D	G	F	M	A	M	G	L	A	S
IT	0,19	0,05	-0,06	-0,09	-0,08	0,01	0,11	0,12	0,09	0,12	0,14	0,17
KF	0,232	0,101	0,02	-0,011	-0,013	0,079	0,141	0,144	0,108	0,144	0,168	0,204
KA	0,1672	0,1158	-0,0395	-0,0368	-0,0669	0,0074	0,091	0,1113	0,0994	0,0865	0,1113	0,1467
CB	0,1711	0,1244	-0,0043	-0,0095	-0,0305	0,0404	0,0989	0,1119	0,1011	0,0866	0,113	0,1471

Tabella 4.4 – Tassi netti medi di evaporazione, in metri (da Taddio, Togni, op. cit.*)*

Noti i tassi, è possibile calcolare i valori di perdita evaporativa:

$$e_b^{(mese)} = \frac{u_b^{(mese)} - l_b^{(mese)}}{\overline{h_b^{(mese)}} - \underline{h_b^{(mese)}}} \times \eta_b^{(mese)} \qquad (4.1),$$

dove u e l rappresentano i volumi minimo e massimo del serbatoio b al mese considerato, mentre $\overline{h}$ e $\underline{h}$ rappresentano le quote di pelo libero massima e minima. Queste grandezze vengono richiamate, per questioni di praticità, nella seguente Tabella 4.5.

	h_b min [m s.l.m.]	$u_b^{(t)}$ [Mm^3]	h_b max [m s.l.m.]	$l_b^{(t)}$ [Mm^3]
IT	1006	700	1029,5	5624
KF	972	140	976,6	1040
KA	475,5	50	488,5	64800
CB	295	4000	330,5	68000

Tabella 4.5 – Quote di pelo libero minime e massime, volumi di invaso minimi e massimi dei serbatoi della rete

Note tutte le grandezze, le perdite evaporative per ogni mese sono calcolate applicando l'Equazione (4.4), la quale ha come ipotesi fondamentale la considerazione dell'invaso come cilindrico, condizione che potrebbe causare problemi di sovrastima o sottostima. Un'ulteriore considerazione deve, a questo punto, essere fatta: le perdite evaporative relative al serbatoio di Kafue sono date dalla somma dei valori dovuti sia a Kafue Gorge, sia a Kafue Flats. Detti valori sono riportati nella seguente Tabella 4.6, che tiene conto anche dell'ultima osservazione.

[mese]	O	N	D	G	F	M	A	M	G	L	A	S
I	40	10	0	0	0	2	23	25	19	25	29	36
KF	277	121	24	0	0	94	169	169	129	172	201	244
KA	833	577	0	0	0	37	453	554	495	431	554	731
CB	308	224	0	0	0	73	178	202	182	156	204	265

Tabella 4.6 – Perdite evaporative complessive medie mensili, in [m^3]

Osservando la precedente Tabella 4.6, si notano alcuni valori nulli, prevalentemente concentrati nei mesi estivi, tra dicembre, gennaio e febbraio: essi sono stati assegnati quando le precipitazioni superavano le perdite nette mensili $\eta_b^{(t)}$, causa che avrebbe indotto perdite totali negative, non gestibili mediante Aquafun.

Prima di procedere, occorre definire se esistono dei ritardi di trasporto significativi lungo i canali: essendo adottato un passo mensile, l'unico tratto in cui i ritardi di trasporto assumono valori significativamente degni di nota, è quello che congiunge Itezhitezhi a Kafue, cui è assegnato un valore pari a due periodi, a significare che l'acqua uscente da Ithezhitezhi raggiunge il serbatoio di Kafue dopo due mesi.

4.3.3 DEFINIZIONE DELLE FUNZIONI DI UTILITÀ

Le funzioni di utilità definite per la rete del Fiume Zambesi rispondono ai criteri generali già individuati nel Capitolo 1, e fanno riferimento unicamente al beneficio ottenuto dalla produzione di energia elettrica, funzione della quantità di portata turbinata dai singoli impianti, presenti sui serbatoi di Kariba, Kafue e Cabora Bassa. Esse si associano, tuttavia, rispettando le caratteristiche di Aquafun, ai canali uscenti dai sopraccitati serbatoi.
In questa sede si richiama solamente la modalità costruttiva di dette funzioni, per i dettagli tecnici sulla loro definizione si rimanda a [Taddio, Togni, 1993].
Per la definizione della massima utilità di un impianto si considera un mese della durata di *31* giorni; se si definisce con W_i^{31} la massima portata turbinabile dall'impianto *i*, espressa in *[Mm^3]*, e con *Pi* il valore della potenza dell'impianto *i* in *[kW]*, si calcola la funzione di utilità secondo una legge esponenziale:

$$U_i^{31}(f_i) = P_i \times \left(1 - \exp\left(\frac{-f_i}{\omega_i}\right)\right) \qquad (4.2),$$

in cui

$$\omega_i = \frac{W_i^{31}}{5} \qquad (4.3)$$

è la *costante di tempo dell'esponenziale* e f_i il flusso che viene turbinato, in metri cubi al secondo.
Al fine di costruire la linearizzazione a tratti necessaria per la definizione in Aquafun, il flusso turbinato è stato suddiviso, in modo arbitrario, in tratti che corrispondono al *50*%, *70*%, *80*%, *90*%, *95*%, *99*% e *100*% del totale, in modo da migliorare la linearizzazione avvicinandosi al totale.
Le funzioni ottenute sono riportate nella seguente Tabella 4.7.

Portata	Utilità marginale
2196	529
878	75
439	34
439	21
220	14
176	11
44	9
25445	0
Kariba	

Portata	Utilità marginale
337	2450
135	347
68	159
67	97
34	65
27	52
7	43
11383	0
Kafue Gorge	

Portata	Utilità marginale
3026	635
1211	76
605	41
605	25
303	17
242	13
61	12
37364	0
Cabora Bassa	

Tabella 4.7 – Funzioni di utilità linearizzate per i serbatoi

I valori delle portate turbinabili per i mesi di durata differente sono ottenuti semplicemente come frazione dell'utilità dei mesi con *31* giorni.

5

ANALISI DEGLI AFFLUSSI E SERIE SINTETICHE

Per testare la rete neurale occorre disporre di numerose serie di afflussi e di rilasci: quando non si disponga di serie reali, occorre generarne di nuove in modo sintetico. Il presente capitolo è volto alla descrizione ed all'analisi di tutte le serie di afflussi utilizzate per testare la rete.

5.1. INTRODUZIONE

Per poter addestrare la rete, al fine di "istruirla" con tutte le casistiche che si potrebbero verificare, quindi è necessario disporre di una elevata quantità di dati
Purtroppo, come quasi sempre avviene, i dati a disposizione sono in numero molto esiguo, dal momento che i fenomeni si verificano una sola volta, e non è possibile ripetere esperimenti sui sistemi reali.
Occorre pertanto, partendo dalle rilevazioni effettuate negli anni, costruire delle nuove serie di dati, dette *serie sintetiche*, che abbiano la fondamentale caratteristica di riprodurre il fenomeno realmente verificatosi, ma che siano da questo diversi. Lo scopo di questa operazione è disporre di ulteriori serie di dati, che siano in qualche modo "realistici", in modo da poter essere utilizzati per effettuare sperimentazioni su una rappresentazione modellistica del sistema reale: in buona sostanza, devono poter rispondere alla semplice domanda: *"Che cosa sarebbe accaduto al sistema se in natura si fosse verificato questo fenomeno anziché quello che è realmente successo?"*.

5.2. SERIE STORICHE DI AFFLUSSI

5.2.1 ANALISI DI BASE

I dati reali effettivamente disponibili sono gli afflussi al sistema idrico dello Zambesi tra l'Ottobre del *1930* ed il Settembre del *1942*. Nella Tabella 5.1, a pagina seguente, si riportano gli afflussi, raggruppati per anno e per mese. I valori rappresentano una media dei milioni di metri cubi affluiti su tutto il bacino lungo tutto l'arco temporale mensile. Da una prima semplice analisi dei valori si evince come i mesi più piovosi siano febbraio o marzo, a seconda dell'anno preso in considerazione, mentre quello meno piovoso sia novembre, e come l'anno in cui si sono verificati gli afflussi più elevati sia stato l'anno *X* (tra Ottobre *1939* e Settembre *1940*), mentre l'anno meno carico sia stato invece il *XII* (Ottobre *1941* – Settembre *1942*).
Una prima rappresentazione dell'andamento della serie di afflussi è stata ottenuta plottando i dati a disposizione, come riportato nel seguente Grafico 4.1.

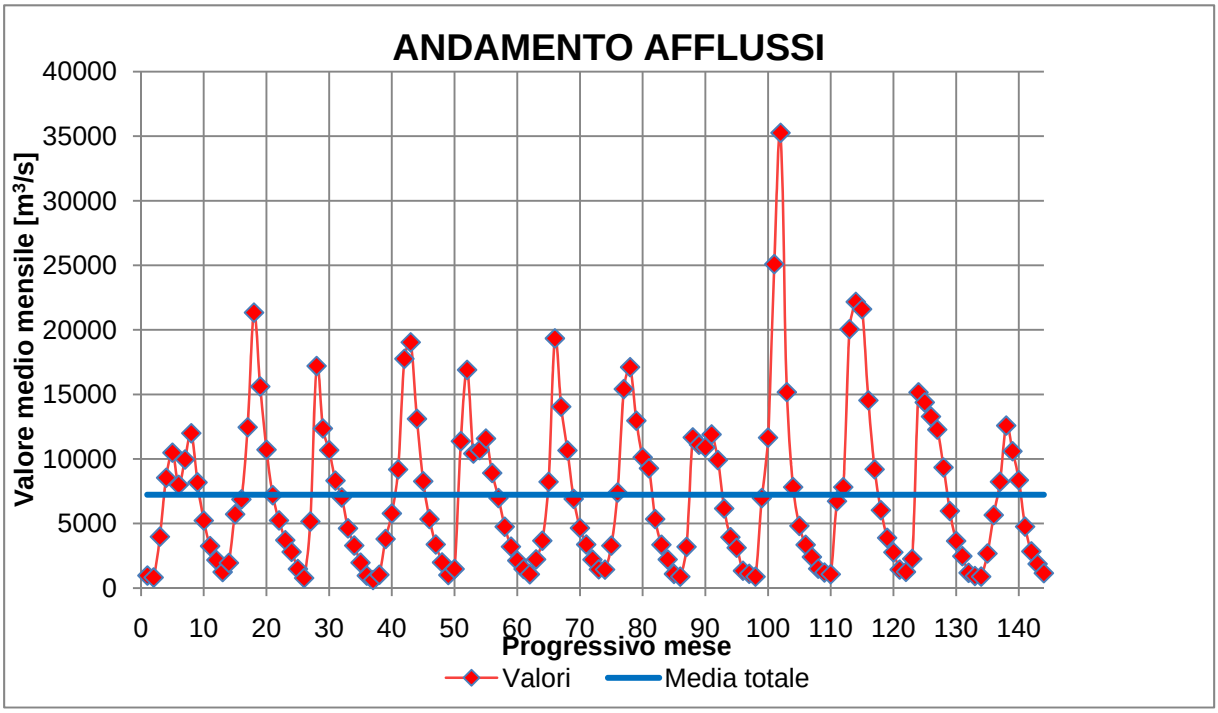

Grafico 5.1 –Andamento della serie originale di afflussi, con media e varianza dei dati graficati insieme

Il grafico avvalora quanto detto in precedenza, relativamente alla massima precipitazione nell'anno *X* e al mese di Febbraio. Si noti in azzurro la media dei valori.

[m^3/s*mese]	ANNO I	ANNO II	ANNO III	ANNO IV	ANNO V	ANNO VI	ANNO VII	ANNO VIII	ANNO IX	ANNO X	ANNO XI	ANNO XII	*MEDIA*	*SOMMA*
OTTOBRE	974	1250	1483	626	1014	1486	1455	1101	1110	1207	1440	941	*1174*	*14087*
NOVEMBRE	824	1956	786	1035	1486	1076	1438	891	886	1072	1258	895	*1134*	*13603*
DICEMBRE	3984	5725	5158	3808	11382	2227	3290	3198	6979	6735	2262	2682	*4786*	*57430*
GENNAIO	8552	6876	17205	5785	16903	3671	7402	11674	11654	7822	15163	5660	*9864*	*118367*
FEBBRAIO	10491	12459	12365	9190	10423	8234	15421	11108	25093	20057	14390	8238	*13122*	*157469*
MARZO	7999	21343	10689	17764	10695	19344	17112	10904	35265	22175	13288	12588	*16597*	*199166*
APRILE	9966	15616	8327	19041	11586	14050	12969	11905	15176	21606	12277	10602	*13593*	*163121*
MAGGIO	12000	10720	7027	13109	8916	10662	10130	9918	7824	14544	9347	8364	*10213*	*122561*
GIUGNO	8187	7199	4632	8278	6978	6911	9279	6171	4822	9203	5974	4764	*6867*	*82398*
LUGLIO	5234	5246	3300	5347	4756	4658	5348	3943	3355	6037	3651	2844	*4477*	*53719*
AGOSTO	3255	3713	1972	3385	3212	3379	3362	3128	2414	3894	2476	1875	*3005*	*36065*
SETTEMBRE	2180	2805	969	1991	2131	2211	2219	1348	1503	2776	1187	1160	*1873*	*22480*
MEDIA	*6137*	*7909*	*6159*	*7447*	*7457*	*6492*	*7452*	*6274*	*9673*	*9761*	*6893*	*5051*	media	7225
SOMMA	*73646*	*94908*	*73913*	*89359*	*89482*	*77909*	*89425*	*75289*	*116081*	*117128*	*82713*	*60613*	varianza	36523039

Tabella 5.1 – Tabella degli afflussi, in verde l'anno più piovoso, in rosso quello con meno precipitazioni (fonte: Taddio, Togni, op. cit.*)*

5.2.2 ULTERIORI ANALISI PECULIARI

Mediante il software Matlab (per il dettaglio delle funzioni utilizzate si rimanda all'Appendice A2) sono state effettuate delle analisi più approfondite della medesima serie di afflusso, volta a comprendere meglio come essa è strutturata, e quali possono essere le sue peculiarità di cui tenere conto, nella generazione di nuove serie sintetiche.

Andamenti annuali

La prima analisi svolta ha prodotto come risultato un grafico (Grafico 5.2 che segue) in cui si riportano le medie e le varianze relative ai singoli anni (Anno I, Anno II, …, Anno XII).

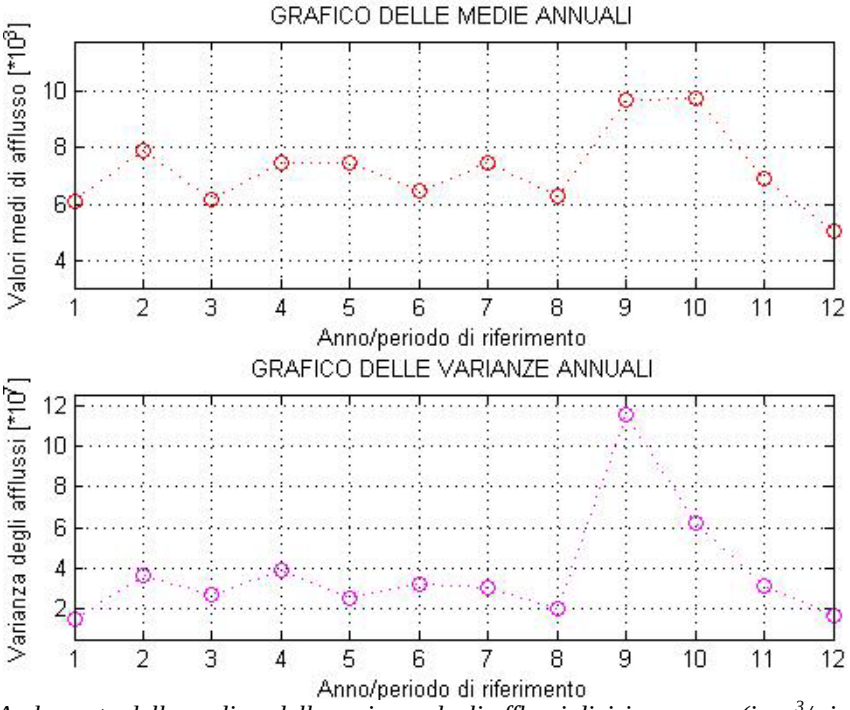

Grafico 5.2 – Andamento delle medie e delle varianze degli afflussi divisi per anno (in m^3/s i valori medi, in m^6/s^2 le varianze)

Ciò conferma quanto detto in precedenza: sull'anno più piovoso e il meno piovoso. I restanti anni presentano afflussi relativamente costanti, dal momento che i loro valori medi sono molto simili tra loro, ed oscillano tra *6000* e *8000* metri cubi. L'analisi delle varianze può essere una verifica del fatto che la serie presa in considerazione è quella effettivamente verificatasi in natura: per ognuno degli anni considerati, infatti, i valori delle varianze sono sempre tra loro differenti, e questo è un valido indice di valutazione della significatività della serie.

Andamenti mensili

La seconda analisi svolta sulla serie di afflussi ha avuto come scopo la produzione di un grafico in cui si sono riportati gli andamenti medi (e le relative varianze) degli afflussi raggruppati per mese (essendo *12* periodi gli anni a disposizione, ogni serie mensile conterrà *12* dati). I risultati sono riportati nel seguente Grafico 5.3.

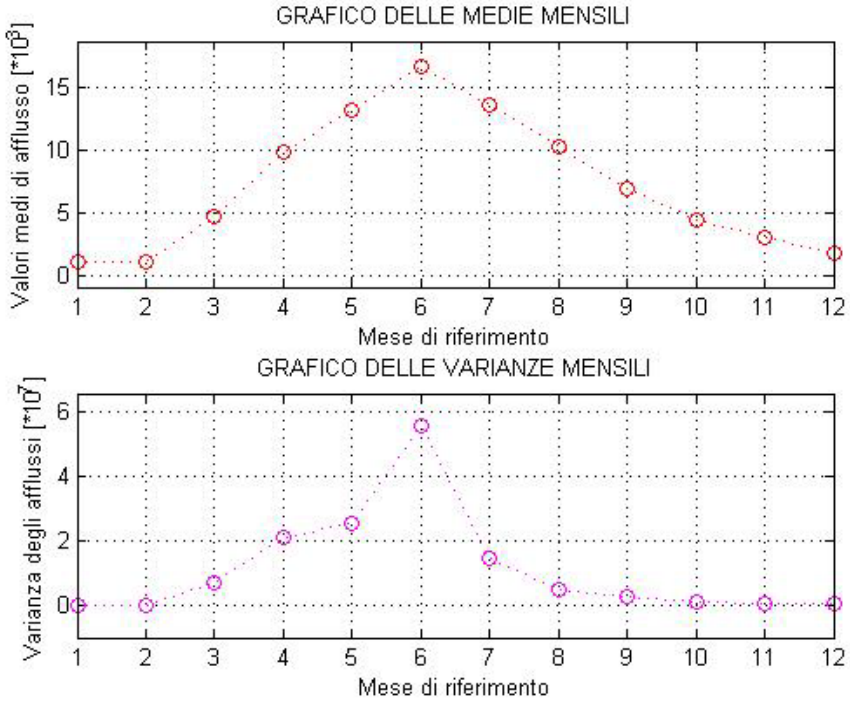

Grafico 5.3 – Andamento delle medie e delle varianze degli afflussi divisi per mese, espressi in metri cubi al secondo (il mese 1 corrisponde a Ottobre, il mese 12 a Settembre)

Anche in questi grafici si trova conferma di quanto affermato in precedenza: il mese in cui mediamente si verificano gli afflussi più elevati è marzo, seguito, a pari distanza da febbraio e aprile, mentre gli afflussi meno consistenti si hanno tra settembre, ottobre e novembre (valore minimo). L'andamento dei valori medi consente di trovare una conferma del clima locale (tra il *12*° ed il *20*° parallelo Sud, come spiegato nel Capitolo 4): si vede la presenza di una stagione secca (tra luglio e dicembre, stagione primaverile ed estiva), e la presenza di una, invece, umida (nei mesi autunnali – invernali, tra gennaio e giugno), con il picco in marzo.

Andamenti su finestra mobile

L'ultima, più interessante, analisi svolta ha permesso di calcolare i valori medi e le varianze degli afflussi, considerati ogni volta in numero pari a *13*, utilizzando una finestra mobile di ampiezza *12*, contenente quindi un semestre precedente al valore centrale ed un semestre successivo. Questa analisi è stata utile per capire la correlazione tra i dati, nonché le ciclicità e le sottoperiodicità interne al sistema.

Al fine di poter effettuare il conteggio del valore medio e della varianza, senza dover alterare l'ampiezza della finestra di analisi, sono stati esclusi i primi e gli ultimi sei valori di afflusso, in modo tale che sia il primo dato analizzato avesse sei valori che lo precedevano, e, allo stesso modo, l'ultimo dato preso in considerazione ne avesse sei posteriori.

I risultati sono stati, al solito, riportati in due grafici distinti per non appesantire troppo la visualizzazione (Grafico 5.4.a e Grafico 5.4.b).

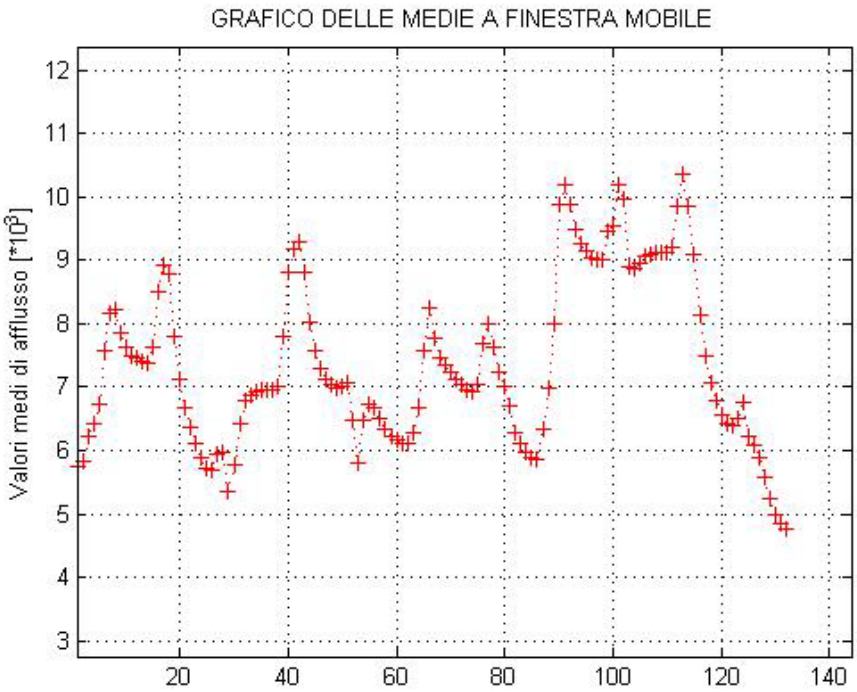

Grafico 5.4.a – Rappresentazione delle medie su intervalli a finestra mobile, in metri cubi per secondo

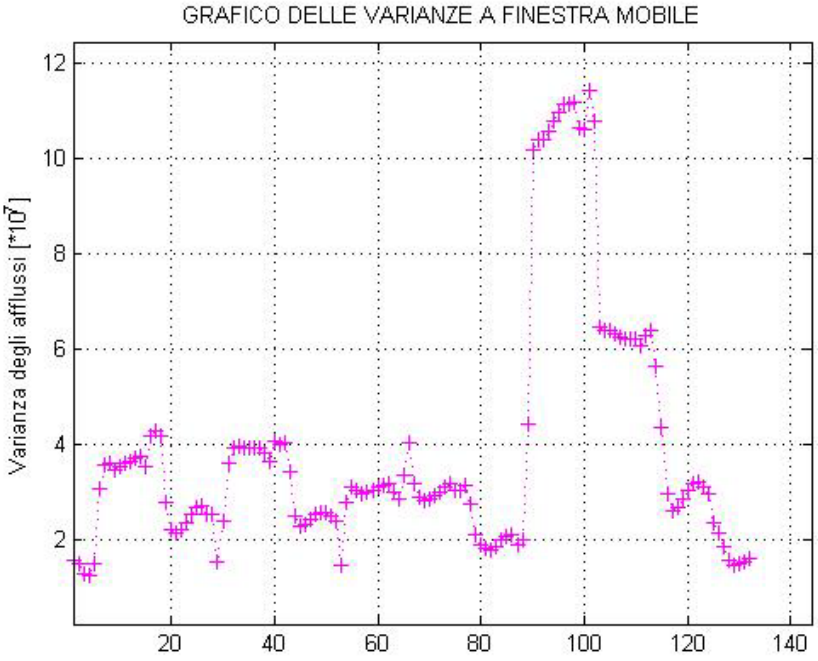

Grafico 5.4.b – Rappresentazione delle varianze su intervalli a finestra mobile

Analizzando il primo grafico, si rileva un'oscillazione dei valori medi, che seguono un andamento ciclico, con i minimi relativi ai periodi più secchi e i massimi relativi ai periodi più umidi. Avvicinandosi ai passi finali, si nota, tra gli step *90* e *120*, un sensibile incremento delle medie: questo è dovuto al fatto che vengono presi in considerazione i dati relativi agli anni più piovosi, all'interno dei quali i mesi meno piovosi hanno un apporto di acqua spesso molto prossimo ai mesi in cui negli altri anni si hanno le precipitazioni maggiori.
Nell'andamento complessivo dei dati non è possibile, inoltre, riconoscere con precisione un andamento ciclico.

5.3 SERIE SINTETICHE

Quelli analizzati e discussi in precedenza sono i soli dati che sono storicamente misurati. Come detto, per tarare e validare la rete neurale, occorre disporre di numerosissime serie di dati che coprano quanta più parte possibile della casistica che la natura può manifestare: occorre pertanto generare delle nuove serie, denominate *serie sintetiche*, che abbiano assumano dei valori realistici.

5.3.1 SERIE SINTETICHE COSTRUITE IN PASSATO

In un precedente lavoro di Tesi di Laurea [Onida, 2002], erano già state generate numerose serie sintetiche di afflussi. La costruzione delle serie artificiali per il sopraccitato lavoro è avvenuta seguendo due strade, differenti per metodologia ma uguali dal punto di vista dei risultati:

1) La prima scelta è stata quella di partire dalla serie originale (descritta al precedente paragrafo), e generare da questa altre successioni di dati, semplicemente aggiungendo o togliendo una percentuale (da *-70%* a *+200%*, con un intervallo del *10%*). Si sono così ottenute *27* serie di dati, diverse da quella originale di partenza nei valori ma non nella struttura;
2) La seconda strada seguita è consistita nella costruzione di serie artificiali che rappresentassero gli afflussi riferiti a particolari situazioni climatologiche, appositamente realizzate; nella fattispecie, un periodo secco (serie "Dry"), uno umido ("Wet"), e periodi con apporto idrico casuale (serie "Random 1", "Random 2" e "Random 3"). Per ottenere tutte queste serie si è realizzato un nucleo di base, della durata generalmente di quattro periodi, ripetuta uguale per tre volte. Terminata questa prima fase "di costruzione" delle serie, si è aumentato il numero di dati di afflusso applicando le stesse variazioni percentuali descritte nella fase precedente, a tutti i valori sintetici ottenuti.

Di seguito, si interpretano brevemente i risultati ottenuti.
In particolare, analizzando i dati ottenuti applicando la prima strategia messa in evidenza, si evince che gli input sono sì formalmente differenti gli uni dagli altri, ma i risultati prodotti da Aquafun sono di fatto tutti uguali tra loro, a meno di un valore costante. Ciò perché il software di ottimizzazione implementa al suo interno equazioni lineari, pertanto se i dati di input sono sempre gli stessi traslati di un valore costante (quello che avviene aggiungendo o togliendo la percentuale dalla serie originale), anche i dati di output saranno sempre i medesimi, traslati della stessa costante dei valori in ingresso (proprio per la linearità dell'algoritmo risolutivo).
Relativamente, invece, alle serie sintetiche ottenute applicando la seconda strada, si evince, fin da una preliminare lettura dei valori di afflusso, che la serie "Dry" e la serie "Wet" sono esattamente uguali, e questo potrebbe essere indice o di un errore di salvataggio e trasmissione dei dati su supporti di memorizzazione, o di un errore procedurale di chi ha costruito la serie. Più in dettaglio, se si osserva il seguente Grafico 5.5, relativo alla serie Dry, si evince come essa sia stata costruita replicando quattro volte lo stesso schema, ottenendo dei valori fisicamente poco realistici.

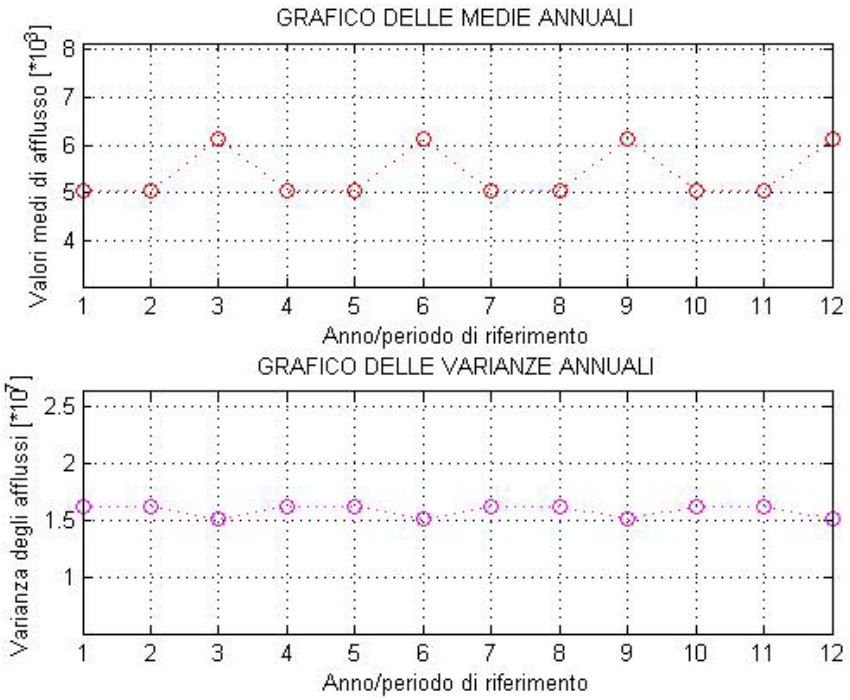

Grafico 5.5 – Valori di medie e varianze annuali per la serie Dry

Il grafico precedente è solo un esempio, che mette in luce come tali serie storiche siano inadatte per gli scopi del presente lavoro.
Infatti, sebbene non sia strettamente necessario, per tarare e validare la rete neurale, disporre di serie fisicamente sensate, è preferibile utilizzare afflussi sintetici che siano realistici.
Si è pertanto deciso di costruire ex – novo delle serie di afflussi artificiali che, basandosi sugli unici afflussi effettivamente verificatisi in natura (raccolti nella serie denominata "Standard"), ed utilizzando le statistiche di questa, come media, varianza e covarianza, riesca a riprodurne l'andamento, pur rappresentando una casistica di eventi molto più ampia e riconducibile a differenti situazioni meteo climatiche reali.
Le modalità procedurali e i risultati ottenuti sono dettagliati nel paragrafo seguente.

5.3.2 GENERAZIONE DELLE SERIE SINTETICHE *AR(0)*

Per generare degli afflussi sintetici si possono seguire diverse modalità, la più semplice delle quali prevede di costruire un modello auto – regressivo di ordine *0* (*AR(0)*), in cui non c'è alcuna correlazione temporale tra valori consecutivi nel tempo. Ciò non è sensato, dal punto di vista fisico, ma ai fini pratici non è importante, dal momento che la rete neurale deve poter essere addestrata con serie qualsiasi di dati, e deve fornire una politica il più possibile vicino a quella ottenuta con ottimizzazione lineare. Al fine, comunque, di mantenere il range di valori non troppo distante dall'intervallo reale in cui cadono gli afflussi, discussi nel Paragrafo 5.2, i valori casuali estratti sono stati collegati alle statistiche della serie storica di dati, che per semplicità vengono richiamate nella seguente Tabella 5.2.

MEDIA [m^3/s*mese]	7225
VARIANZA [m^6/s^2*$mese^2$]	36523039
DEVIAZIONE STANDARD [m3/s*mese]	6043

Tabella 5.2 – Statistiche della serie storica di afflussi

Il modello auto regressivo di ordine *0* utilizzato utilizza un'equazione del tipo:

$$a_t^{sintetico} = \overline{a_{serie\ storica}} + k \times \sigma_{serie\ storica} \times (0{,}5 - r) \qquad (5.1),$$

dalla quale si comprende come è ottenuto il valore di afflusso artificiale: la base è il valore medio della serie storica, $\overline{a_{serie\ storica}}$, cui viene aggiunto un termine, dipendente dalla deviazione standard $\sigma_{serie\ storica}$, moltiplicato per un coefficiente *k*, intero, compreso tra *1* e *4*, con la funzione di disperdere o raggruppare i valori della serie intorno alla media (tanto più è alto, tanto più la serie casuale risulterà dispersa intorno alla media, tanto più grandi in modulo saranno i valori). Per garantire che i valori casuali della serie storica oscillino intorno alla media, collocandosi sia al di sotto sia al di sopra di essa, e non siano soltanto maggiori, occorre introdurre un fattore, funzione del casuale estratto, *r*. Detto numero casuale è un reale compreso tra *0* e *1*, per cui il termine

$$(0{,}5 - r)$$

consente di creare termini positivi, quando $r < 0{,}5$, termini negativi, se $r > 0{,}5$, ma anche termini esattamente uguali al valore medio della serie storica, quando $r = 0{,}5$ (l'ultima opzione ha, tuttavia, una probabilità quasi nulla di verificarsi).
Al fine di evitare afflussi negativi, impossibili da gestire, è stato preso il valore massimo tra *0* e il risultato dell'equazione (5.2).
Una serie sintetica creata in questo modo garantisce un buona disomogeneità nei valori, dal momento che, su un gran numero di estrazioni casuali di valori per *r*, circa la metà di essi è inferiore a *0,5*, laddove, ovviamente l'altra metà è superiore.
Al fine di limitare l'intervallo di valori che possono essere presi in considerazione, si è deciso di valutare le singole serie sintetiche create, eliminando quelle il cui valore medio si discostasse troppo dal valore medio della serie storica disponibile: essendo quest'ultimo uguale a $7225 \frac{m^3}{s \times mese}$, si è deciso di considerare valide solamente le serie il cui valor medio sia uguale a quello storico diminuito della metà, oppure aumentato della stessa quantità, approssimando per difetto. Pertanto si sono accettate le serie con:

$$3000 \leq \mu_{serie\ sintetica} \leq 10000 \left[\frac{m^3}{sec \times mese}\right] \qquad (5.2).$$

La scelta di ricorrere unicamente a semplici modelli per la generazione delle serie sintetiche è stata dettata unicamente dalla necessità di riuscire a produrre un'enorme quantità di valori con un algoritmo rapido.
Non essendo importante il rispetto della fisica del sistema, si è preferito utilizzare un algoritmo semplice, che consentisse di costruire un gran numero di serie sintetiche, piuttosto che ricorrere ad un algoritmo più complesso che rappresentasse correttamente l'andamento degli afflussi, non essendo questo lo scopo del presente lavoro. Se si confrontano, in scala semilogaritmica, gli andamenti della serie storica (Grafico 5.6) e di una delle serie sintetiche create (la *"ARZ0512"*, Grafico 5.7), si osserva come la seconda non rispecchi l'andamento della prima, producendo, molto probabilmente, in sede di simulazione, dei risultati che non possono essere utilizzati nella gestione del sistema, perché inaccettabili dal punto di vista fisico.

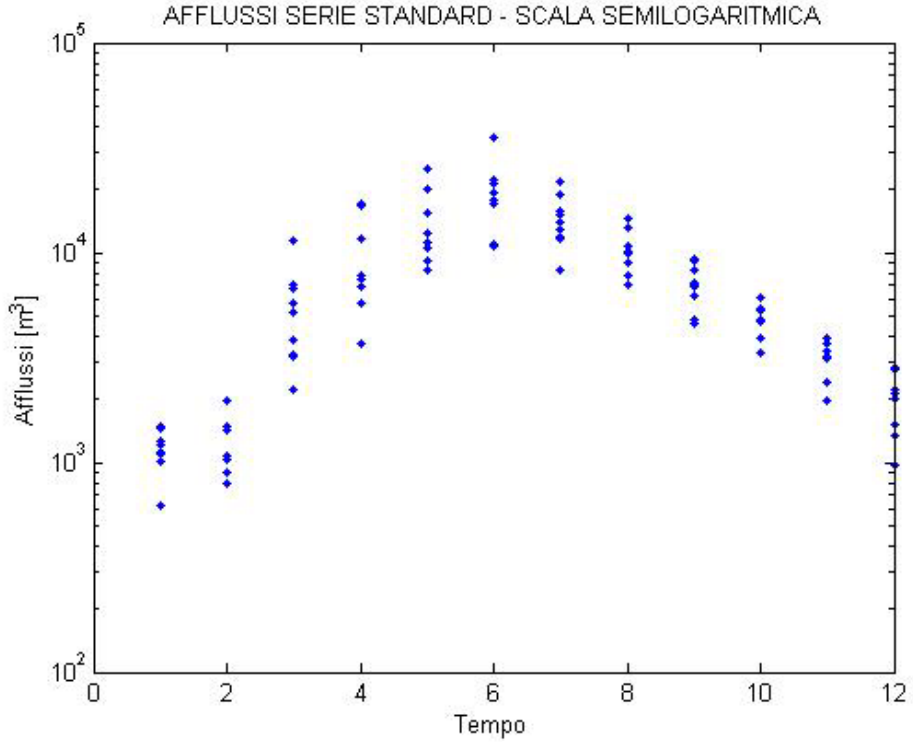

Grafico 5.6 – Rappresentazione in scala semilogaritmica degli afflussi della serie storica dei dati

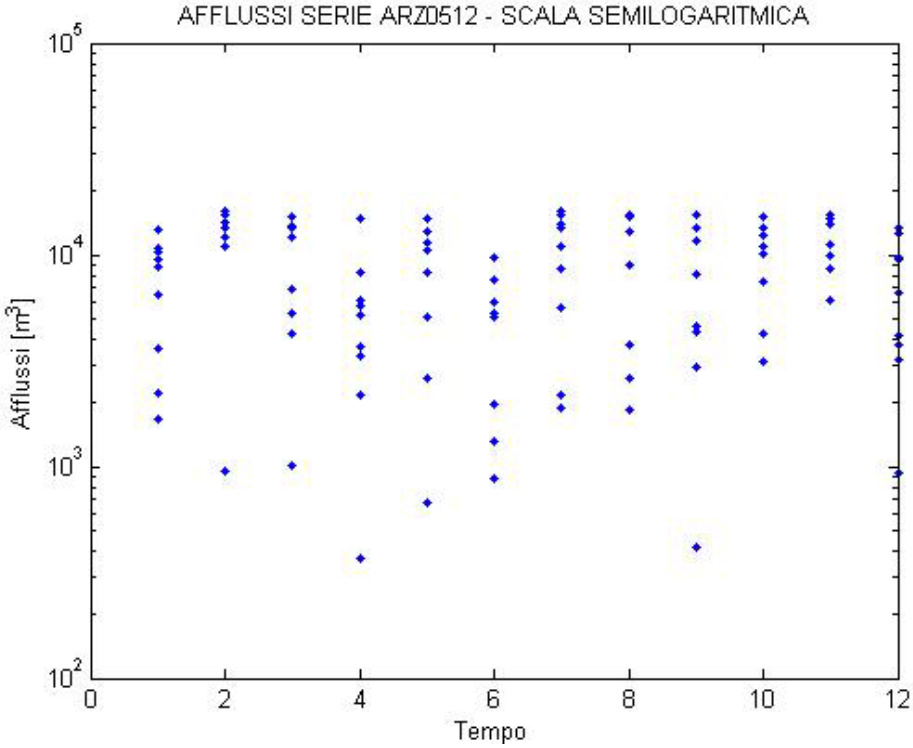

Grafico 5.7 – Rappresentazione, in scala semilogaritmica di una serie di afflussi sintetici creati con il modello AR(0)

Ritenendo opportuno disporre di almeno *1000* serie di afflusso, dal punto di vista operativo, si è deciso di generare *800* serie sintetiche con un modello di ordine *0*, suddividendo equamente tra di esse il valore del coefficiente *k* introdotto con l'equazione (5.2), precisamente per ogni valore di *k* (*1*, *2*, *3*, *4*) sono state create *200* diverse serie. Non necessariamente tutte sono poi state considerate valide, dal momento che quelle che non rispettavano il vincolo sul valor medio sono state escluse in partenza. Al fine di completare l'insieme dei dati necessari, si

sono create altre serie, in numero minore, con un modello leggermente più complicato, di forma simile ad un auto – regressivo di ordine *1*.

5.3.3 GENERAZIONE DI ALTRE SERIE SINTETICHE

Alla base di un modello auto – regressivo di ordine *1* (*AR(1)*) c'è il legame tra due consecutivi valori della serie, che, nel caso di una serie di afflussi, fanno riferimento al dato al tempo *t* ed al precedente, al tempo *t – 1*.
Il modello adottato per generare le ulteriori serie sintetiche si richiama ad un auto regressivo di primo ordine, dal momento che si presenta nella forma:

$$a_t^{sintetico} = \overline{a_{storici}} + \left(k_1 + k_2 \times a_{t-1}^{storico}\right) \qquad (5.3),$$

dove k_2 è un coefficiente casuale ($0 \leq k_2 \leq 1$), che esprime la forza del legame tra i due valori di afflusso, mentre k_1 è un altro fattore casuale (estratto da una distribuzione normale gaussiana) che rappresenta il disturbo stocastico presente in ogni istante di simulazione. Il legame è tra il valore attuale sintetico ed il precedente storico, e non, come dovrebbe essere in un *AR(1)* vero e proprio, con il precedente sintetico. Detti coefficienti mutano ad ogni passo temporale, in modo tale da garantire la massima generalità possibile. Per mantenere la serie sintetica creata il più possibile vicino al valor medio della serie storica di partenza, ad ogni generazione casuale viene aggiunto tale valore medio. Inoltre, il primo valore della serie è costruito aggiungendo al valore medio della serie di partenza un casuale estratto da una gaussiana, motivo per cui il primo valore sintetico non si discosterà mai eccessivamente da quello storico.
Anche queste serie sintetiche sono state mantenute unicamente quando il loro valor medio non si discostava troppo dalla statistica della serie storica a disposizione, utilizzando gli stessi criteri esposti nell'equazione (5.2) relativa al modello auto – regressivo di ordine *0*.
Nelle analisi condotte in questo lavoro, per semplificare la trattazione ci si riferirà alle serie storiche prodotte con il modello presentato in questo paragrafo con la dicitura "AR(1)", sapendo tuttavia che non si ha a che fare con un auto regressivo del primo ordine.
Come si evince dal seguente Grafico 5.8, in cui è rappresentato l'andamento di una serie sintetica generata con questo secondo modello (la *"ARO0001"*, in scala semilogaritmica), esso rispecchia meglio la struttura della serie storica, già discussa in precedenza.

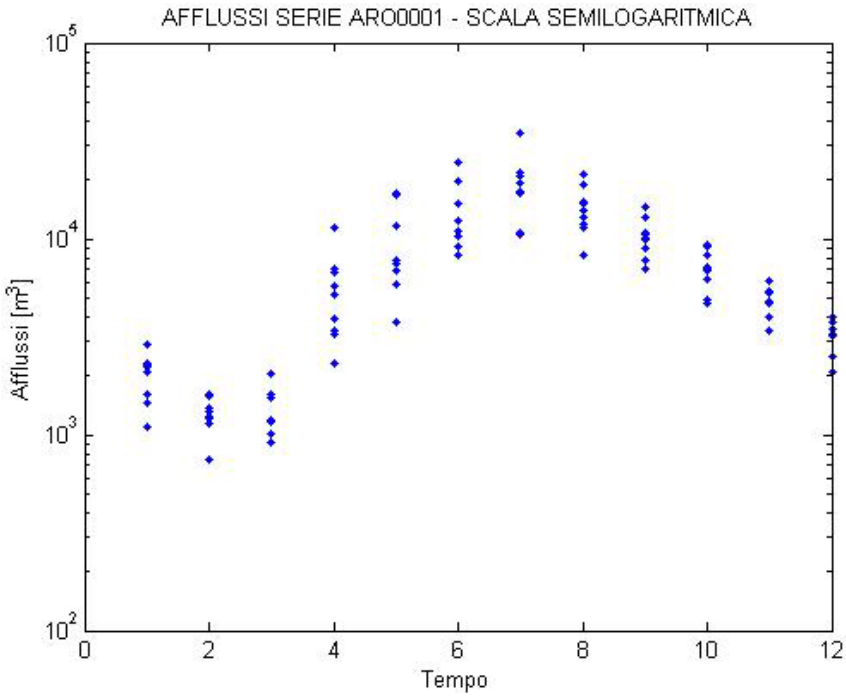

Grafico 5.8 –Rappresentazione in scala semilogaritmica di una serie sintetica ottenuta con il modello simile ad un auto regressivo di primo ordine

Si è detto che non è importante che le serie artificiali rispecchino la struttura degli afflussi storicamente verificatisi, perché le reti neurali possono, e devono, essere addestrate con qualunque serie di dati. Tuttavia, quando si intende includere nella rete neurale la ciclicità del sistema, non è possibile utilizzare dati costruiti in modo completamente casuale: per queste sperimentazioni, allora, le reti neurali verranno addestrate solamente con le serie sintetiche costruite come quella presentata in questa sezione.

6

ELABORAZIONE DEI DATI ED ANALISI DEI RISULTATI

Nel presente capitolo si descrivono le procedure adottate per operare con le Reti Neurali partendo dai dati di afflusso, invaso e controllo utilizzati con l'ottimizzatore lineare. Da esso, infatti, occorre estrarre tutti i vettori necessari alla costruzione della matrice di addestramento, alla taratura ed alla validazione della Rete Neurale, per la quale occorre anche scegliere la struttura nascosta migliore.

6.1 OPERAZIONI PRELIMINARI ALLA COSTRUZIONE DELLA MATRICE DI ADDESTRAMENTO

6.1.1 CREAZIONE DEI FILE DI INPUT AD AQUAFUN

Per poter procedere con l'ottimizzazione della rete neurale occorre disporre dei dati provenienti dall'ottimizzazione lineare: alcuni, come ad esempio i vincoli di volume sui serbatoi, la durata della simulazione stessa e la struttura della rete idrica, si mantengono costanti per tutte le analisi che vengono condotte; altri, quali gli afflussi, devono necessariamente variare di volta in volta, per garantire la più ampia generalità possibile.
Data la loro importanza e peculiarità, alla generazione di serie sintetiche di afflussi, necessarie per supplire alla scarsità di dati storici, è stato dedicato un intera sezione di questo lavoro, il Capitolo 5.
Oltre alla generazione dei valori, occorre effettuare una serie di operazioni di contorno, necessarie a preparare i file che costituiscono gli input ad Aquafun.
Questa operazione è necessaria per evitare di inserire a mano nel programma tutte le reti da simulare: è possibile utilizzare una procedura Matlab che, rapidamente, crea tanti file uguali tra loro, cui è assegnato un nome opportuno.
Contestualmente alla generazione delle serie sintetiche di afflusso viene creato un file testuale nel formato idoneo ad Aquafun, che contiene, oltre ai dati numerici, anche informazioni di intestazione necessarie, al software per capire quali dati sta analizzando al momento.
È importante, per riuscire a distinguere ogni simulazione dall'altra, assegnare alle reti un nome che le identifichi in modo univoco.

6.1.2 RIPARTIZIONE DEGLI AFFLUSSI AD OGNI SERBATOIO

Dal momento che occorre specificare, ad ogni passo temporale, qual è la quota parte di afflusso che è diretta ad uno specifico serbatoio, si è scelto di assegnare un nome che richiama il modello usato per la generazione degli afflussi ed un numero progressivo della serie, ad esempio: "*ARZ0345.NT2*" identifica una delle serie create con il modello auto regressivo di ordine *0*. Lo stesso nome, ovviamente, viene assegnato, con opportuno dominio, anche agli altri file di input. L'unico vincolo imposto al nome riguarda il numero massimo di caratteri che esso deve avere, pari ad *8*, dominio escluso, per poter essere inserito in Aquafun senza problemi.
In seguito alla generazione delle serie si è proceduto, utilizzando come rapporti di suddivisione quelli già definiti in passato (per tutte le serie create, si considerano sempre le medesime proporzioni, "rapporti.txt" [Taddio, Togni, 1993] [Onida, 2002]), a calcolare la quota parte di afflusso diretta, in ogni periodo, a ciascuno degli invasi.

6.1.3 ESTRAZIONE DEI VETTORI DI CONTROLLO E DI INVASO

Ottenuti, finalmente, tutti i dati di input necessari all'ottimizzatore lineare, è possibile lanciare le varie simulazioni, relative al solo algoritmo Relax, i cui risultati vengono riportati in un nuovo file di testo, con nome uguale all'ingresso.
Da questo si possono estrarre i vettori dei volumi di invaso, e i vettori dei controlli, che vengono salvati in appositi file.
Per poter avviare le simulazioni, devono essere inserite delle condizioni al contorno, sia all'inizio, sia alla fine: le prime rappresentano i volumi di invaso iniziali, identificabili con s_0, e devono essere definite le condizioni di chiusura, che, essendo il problema su orizzonte finito, rappresentano la situazione finale del sistema, come esso si configura al termine della simulazione di *144* periodi. Tuttavia, questi ultimi valori non sono presi in considerazione, in quanto necessari solamente all'ottimizzatore lineare per avere un vincolo sul quale poter calcolare il bilancio ed ottimizzare l'utilità.
Per quanto riguarda i valori iniziali di invaso e in generale le condizioni al contorno, esse sono uguali per tutte le reti prese in analisi, essendo uniforme la struttura delle stesse.

6.1.4 CONTROLLO DI UNIFORMITÀ

Prima di costruire la matrice di addestramento alla rete neurale, occorre verificare che i vettori siano costruiti in modo uniforme, cioè i diversi dati relativi ai vari serbatoi siano elencati sempre allo stesso modo, ossia, per ogni periodo, sia nel vettore di afflusso, sia in quello dei livelli di invaso, sia in quello dei controlli. Nella fattispecie, dopo l'estrazione dei diversi valori, i dati relativi agli afflussi, agli invasi e quelli relativi ai controlli, devono essere ordinati tutti allo stesso modo. Rese uniformi le serie temporali, è possibile costruire le matrici di addestramento, ognuna delle quali sarà relativa ad una specifica rete simulata.

Uno schema dettagliato della procedura qui introdotta è riportato nella seguente Figura 6.1.

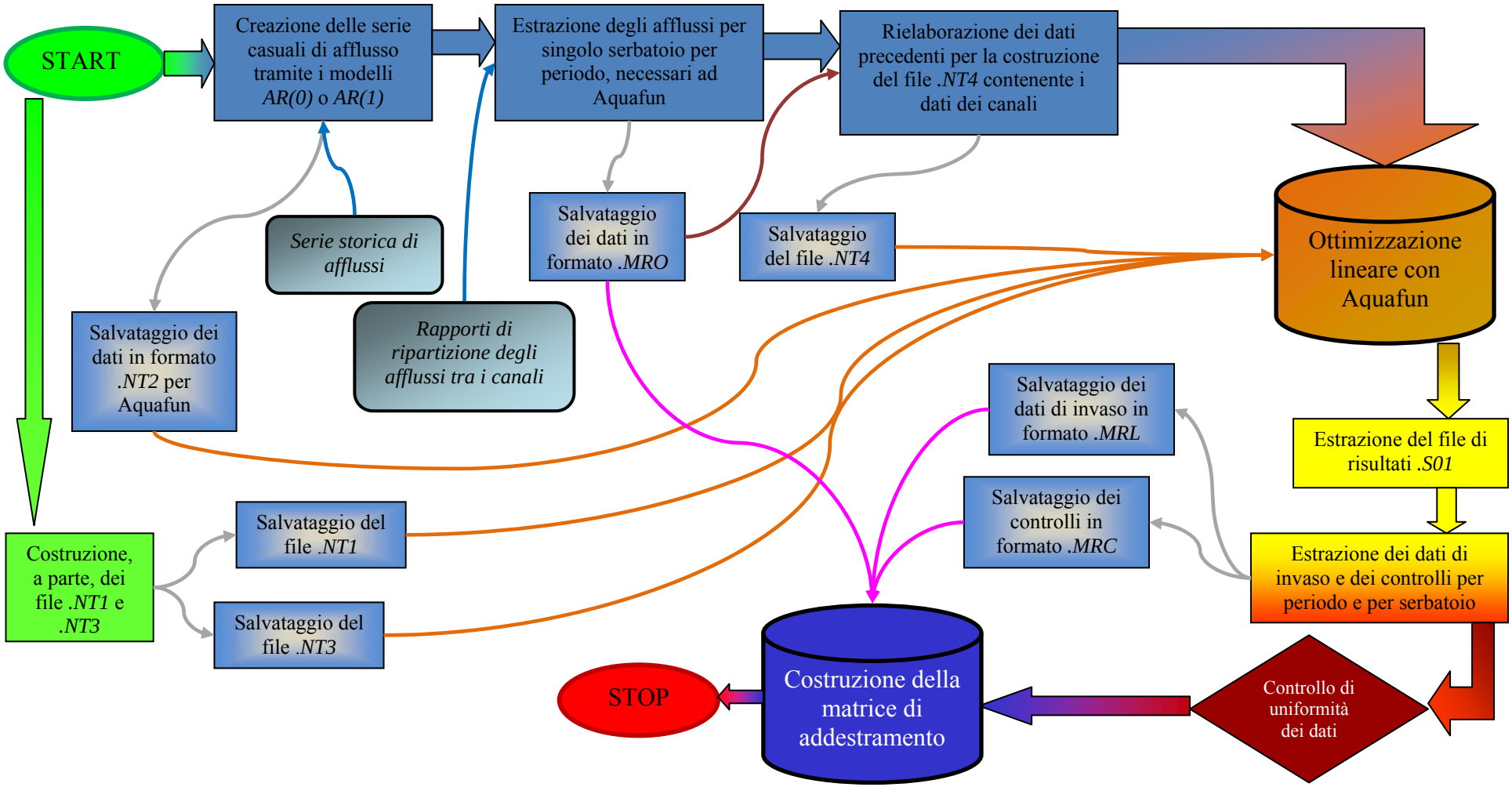

Figura 6.1 – Diagramma di flusso delle operazioni preliminari alla costruzione della matrice di addestramento alla rete neurale

6.2 DATI DI TARATURA E DATI DI VALIDAZIONE

Le serie sintetiche costruite costituiscono l'insieme totale dei dati a disposizione. Una consistente parte di essi deve essere tenuta come insieme di taratura, o calibrazione, mentre la restante parte come insieme di validazione, necessario a verificare che la procedura di addestramento sia andata a buon fine. Ad ogni modo, la verifica ultima sulla bontà della rete deve essere effettuata utilizzando i dati di afflusso storici, quelli effettivamente realizzatisi tra il *1930* ed il *1942*, per i quali si dispongono i valori reali.
Usualmente, la percentuale di dati che si mantiene come insieme di calibrazione è pari all'*80*% del totale, la restante parte (*20%*) si utilizza invece in validazione.
Disponendo di *1000* reti sintetiche, *800* di esse vengono utilizzate in taratura, e le altre *200* in validazione.
Secondo quanto già descritto nel Capitolo 5, sono state generate *200* serie sintetiche utilizzando un modello che si richiama ad un AR(1), e *800* con un modello AR(0), divise in gruppi di *200* secondo il valore assegnato al coefficiente moltiplicativo del valor medio della serie originale. Al fine di garantire un'opportuna eterogeneità nei dati, sono state estratte casualmente le reti che costituiscono l'insieme di validazione (essendo cinque i gruppi sintetici, sono stati estratti *40* elementi da ognuno di essi). Tutte le altre, ottenute per differenza, andranno a formare l'insieme di calibrazione.
Definiti i dati, è possibile procedere a costruire le matrici di input e di controllo alla rete secondo le procedure individuate in precedenza.

6.3 MATRICE DI ADDESTRAMENTO

6.3.1 ASSEMBLAGGIO DEI DATI

La matrice di addestramento, definita nel Capitolo 3, Paragrafo 3.3, ha una struttura del tipo

$$A = [d, (2m + r)] \qquad (6.1),$$

in cui d rappresenta il numero di periodi che si tengono in considerazione, m il numero di serbatoi della rete e r il numero di controlli di ottimizzazione.
Il numero totale di periodi che vengono presi in esame è inferiore alla durata totale dell'orizzonte di progetto, perché se ne escludono alcuni per elidere i possibili effetti di bordo e l'assenza di correlazione tra i dati.
La lunghezza totale dell'orizzonte di progetto, definita con h, è pari a *144* periodi.
Come ipotesi di lavoro, si escludono *12* periodi all'inizio della simulazione e *12* alla fine, in modo tale da lavorare solamente con i *120* periodi centrali, sicuramente privi di effetti di bordo (tra i quali rivestono una particolare importanza i valori estremi dei livelli dei serbatoi, assegnati in Aquafun indipendentemente dalla serie sintetica: essi sono evidentemente uguali per tutte le reti idriche analizzate, e, pertanto, poco significativi). Qualora, tuttavia, non si ottengano risultati soddisfacenti con serie di questa lunghezza, occorrerà provvedere a modificare il numero di periodi da escludere, dapprima riducendo ulteriormente il numero di periodi presi in considerazione, ma se questo intervallo dovesse rimpicciolirsi troppo, si ridurranno le finestre escluse per effetti di bordo.
Il numero di serbatoi, pari a *4*, e il numero di controlli, pari a *5* rimangono fissi in ogni simulazione, essendo data la struttura fisica del sistema. Pertanto:

$$\begin{cases} m = 4 \\ r = 5 \\ d_{start} = 120 \end{cases} \quad (6.2)$$

Considerando ogni vettore di afflusso, ogni vettore dei valori di invaso ed ogni vettore di rilasci come una colonna della matrice, ciò che si ottiene, alla fine, è una tabella con *120* righe (i periodi considerati), e *13* colonne (*4* di afflussi, *4* di invasi e *5* di rilasci).
In particolare, le prime otto colonne saranno utilizzate come dati di input, le ultime cinque per valutare la bontà della rete creata, tramite l'algoritmo di retro – propagazione dell'errore oppure quello di Levenberg e Marquardt.

6.3.2 ESTRAZIONE DELLE SOTTOMATRICI DI INTERESSE

Come spiegato nel Capitolo 3, non tutti i dati di input descritti in precedenza devono necessariamente essere utilizzati per addestrare la rete neurale.
La più semplice struttura possibile è quella che considera in input solamente i vettori di invaso, mentre mantiene invariato il numero di controlli, comunque necessari a valutare la prestazione fornita dalla rete.
Dalla matrice *A* completa vengono estratti allora, solamente i dati relativi agli invasi ed ai rilasci, creando la sottomatrice

$$A_{invasi} = [d, (m + r)] \quad (6.3).$$

In essa, quindi, non si considerano i dati di afflusso. Un'esemplificazione di questa struttura è riportata nella seguente Figura 6.2.

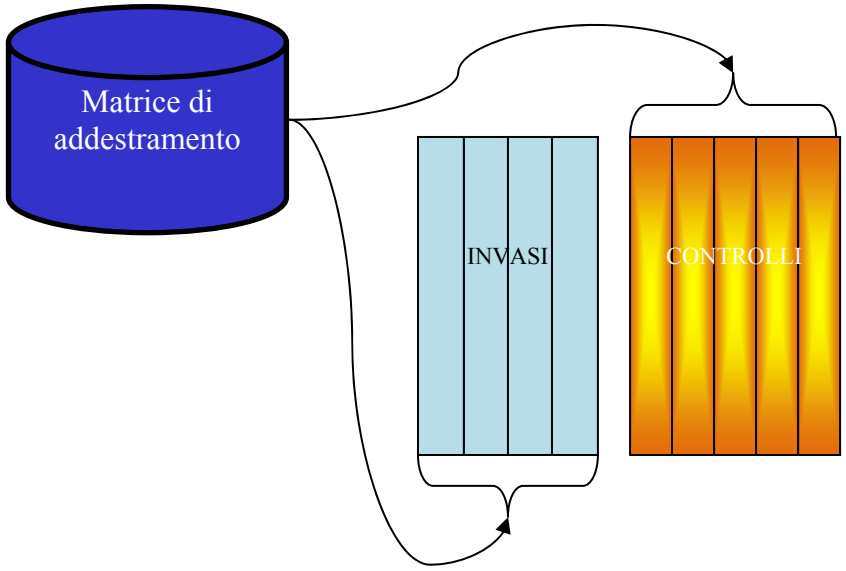

Figura 6.2 – La più semplice matrice di addestramento possibile, in cui non si considerano gli afflussi

In particolare, in riferimento ai dati a disposizione, la matrice di addestramento sarà costituita dall'incolonnamento dei valori relativi a tutte le *800* reti del sottoinsieme di training. Di ciascuna di esse, come detto, si considerano solamente i *120* periodi centrali, per cui il numero di righe complessivo sarà dato da

$$d_{Totali}^{Addestramento} = 800 \times 120 = 96000 \quad (6.4),$$

mentre il numero di colonne sarà pari a

$$\begin{cases} m = 4 \\ r = 5 \end{cases} \quad (6.5),$$

generando così, per l'addestramento, una matrice di input $[96000; 4]$ e una matrice di target $[96000; 5]$.
Se, invece, si costruiscono dei modelli neurali che, utilizzando comunque i dati di tutti gli invasi, producono i controlli per uno solo dei serbatoi, la matrice di input sarà analoga a prima, dimensionalmente $[96000; 5]$, mentre come target si avranno cinque differenti matrici $[96000; 1]$.
Parimenti, per la validazione, si concatenano i dati relativi alle restanti *200* reti idriche sintetiche generate. In questo caso, il numero di righe della matrice è pari a:

$$d_{Totali}^{validazione} = 200 \times 120 = 24000 \quad (6.6),$$

generando, come input, una tabella $[24000; 4]$, e, come target, una tabella $[24000; 5]$ o $[24000; 1]$.

La precedente matrice può essere espansa includendo nei dati di addestramento anche i dati di afflusso, inizialmente relativi all'istante in cui si estrae il valore di invaso, successivamente introducendo un previsore, in modo che riescano ad essere stimati, all'istante *t*, gli afflussi fino all'epoca successiva, *t + 1*. La matrice di addestramento più complessa è quella esemplificata nella seguente Figura 6.3.

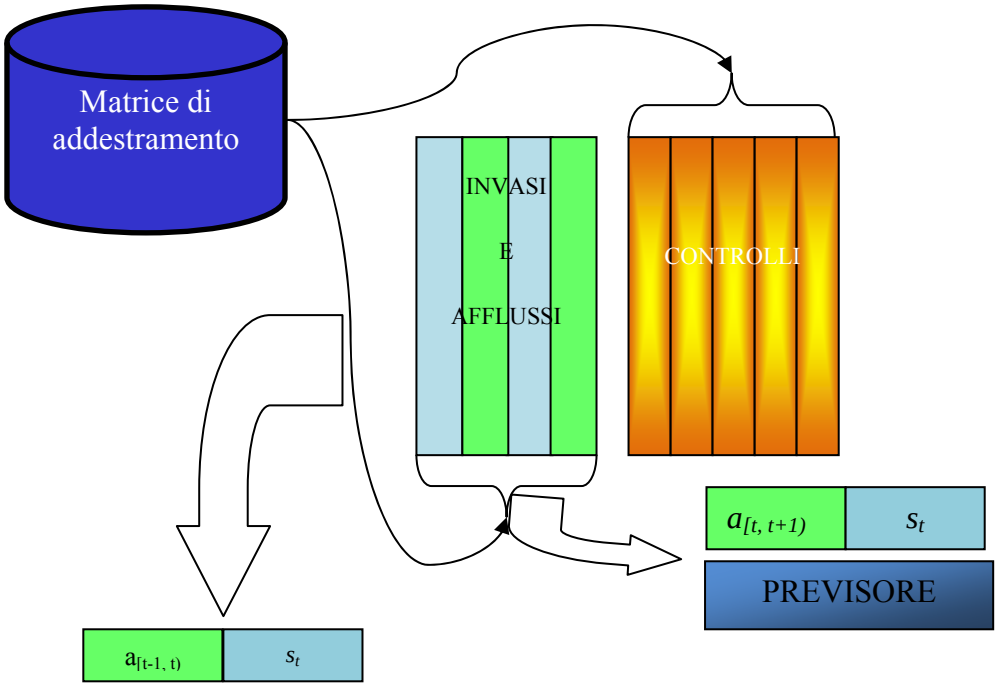

Figura 6.3 – Matrice di addestramento da cui vengono estratti anche i dati di afflusso, ed eventualmente viene inserito un previsore

6.4 ARCHITETTURA DELLA RETE

Prima di procedere con qualsiasi analisi, occorre stabilire quale architettura dare alla rete neurale che si intende addestrare. È verosimile che se si riesce ad individuare una struttura adatta per la gestione completa del bacino (ovverosia tutti e cinque i controlli insieme), allo stesso modo questa si rivelerà adatta anche per le reti costruite per fornire solamente uno dei controlli.
Le forme di reti neurali possibili sono tante (si veda il Capitolo 2), è bene scegliere un'architettura che garantisca la massima generalità possibile e che quasi sempre giunge a convergenza dopo un numero finito, più o meno lungo, di iterazioni.
La struttura adottata per il presente lavoro è quella che risponde meglio al requisito di generalità, ancorché una delle più semplici.
Infatti, si utilizzerà un solo strato nascosto, il cui numero di neuroni deve però essere determinato. Ognuno di essi ha una funzione di attivazione di tipo sigmoidale logaritmico, che garantisce, quasi sempre, il raggiungimento della convergenza. [Cybenko, 1989] Dal punto di vista analitico, riprendendo la definizione generale della funzione di attivazione, introdotta con l'equazione (2.9), la funzione di attivazione adottata nel presente lavoro è:

$$y(I_j) = \text{logsig}(I_j) = \frac{1}{1+\exp(-I_j)} \qquad (6.7),$$

dove I_j è il generico elemento di input.
Tale funzione è definita su tutti gli *n* reali, ma ha codominio nell'intervallo *(0, 1)*, valori cui tende asintoticamente quando l'ascissa diventa infinitamente piccola o grande. Una rappresentazione grafica della funzione log – sigmoidale è fornita nella seguente Figura 6.4.

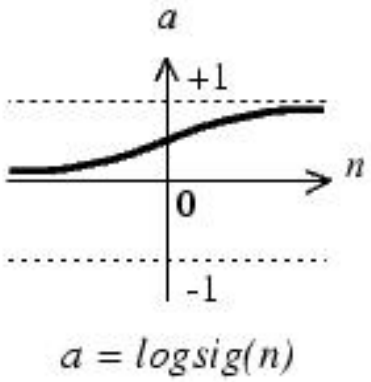

Figura 6.4 – Rappresentazione schematica della funzione di attivazione log – sigmoidale, in cui si mette bene in evidenza l'andamento asintotico agli estremi (da Demuth et al.*, 2000)*

La continuità di questa funzione è il suo principale punto di forza, dal momento che ne garantisce la convergenza al crescere delle iterazioni di addestramento in quasi tutte le circostanze (sono rari i casi, negli esperimenti effettuati, in cui l'addestramento non si è concluso positivamente).
Parimenti, la funzione di attivazione per lo strato di uscita è lineare, in modo che i set di dati vi passino attraverso conoscendo come modifica solamente una variazione di scala. Questa scelta semplifica molto la modellazione, e riduce il carico di lavoro in termini di impiego di risorse software e di tempo.
La struttura della generica Rete Neurale adottata, pertanto, è del tipo di quella riportata nella seguente Figura 6.5.

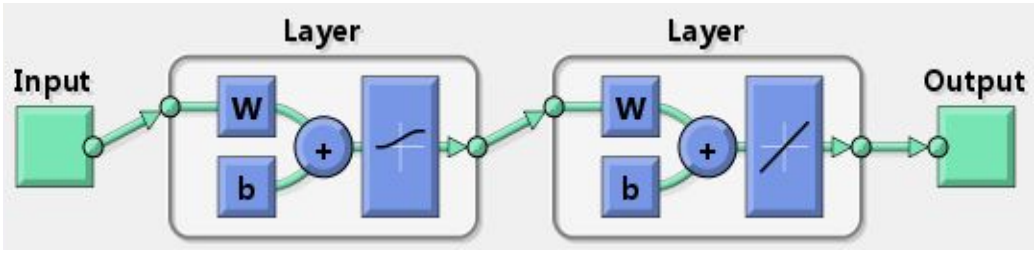

Figura 6.5 – Struttura della generica Rete Neurale, dove "Input" rappresenta la generica matrice di ingresso, "Layer" rappresenta, da sinistra a destra, lo strato nascosto e quello di uscita, "W" la matrice dei pesi, "b" il vettore dei bias, "Output" la matrice o vettore di uscita (da Neural Netwrok Toolbox di Matlab,)

6.5 NOTAZIONI PRELIMINARI ALLE SIMULAZIONI NEURALI

Dopo aver addestrato la rete neurale, essa deve essere utilizzata per simulare la dinamica del sistema dello Zambesi. Tale dinamica è descritta dalla semplice equazione

$$s_{t+1} = s_t + a_{t+1} - e_{t+1} - r_{t+1} \qquad (6.8),$$

che consente, per ogni serbatoio, di calcolare l'invaso all'istante successivo, noto il valore dell'invaso all'istante attuale (s). Ciò che sicuramente è noto, lungo tutto l'orizzonte temporale, è l'evaporazione (e), che si suppone sempre ciclica, secondo lo schema già descritto nel Capitolo 4. Lo scenario degli afflussi (a), analogamente a come opera l'ottimizzatore lineare Aquafun, è noto anch'esso prima di avviare le simulazioni, mentre non è così nei sistemi reali. Si suppone, in effetti, che il simulatore neurale non conosca a priori tutto lo scenario di afflussi, imitando il funzionamento dei sistemi reali: ogni valore, infatti, viene inserito nel modello solamente al passo di simulazione corrente. Infine, il rilascio (r) viene calcolato mediante la rete neurale, già addestrata in precedenza.
La rete neurale viene alimentata, ad ogni istante temporale, con gli invasi relativi solamente al passo stesso, e l'equazione introdotta con la formula (6.8) viene utilizzata per calcolare il volume di invaso all'istante successivo. Qualora esso non sia fisicamente possibile, ad esempio perché esce dalle condizioni vincolari imposte, il rilascio che si considera non è più quello della rete neurale, ma quello tale da rendere accettabile il volume di invaso. Sostanzialmente, in questo caso, viene imposto all'invaso il volume di vincolo, e il rilascio viene calcolato di conseguenza, ignorando il responso della rete neurale. Qualora, inoltre, tale risultato sia negativo, quindi insensato, esso viene posto uguale a zero. Lo schema procedurale è quello descritto nella seguente Figura 6.6.

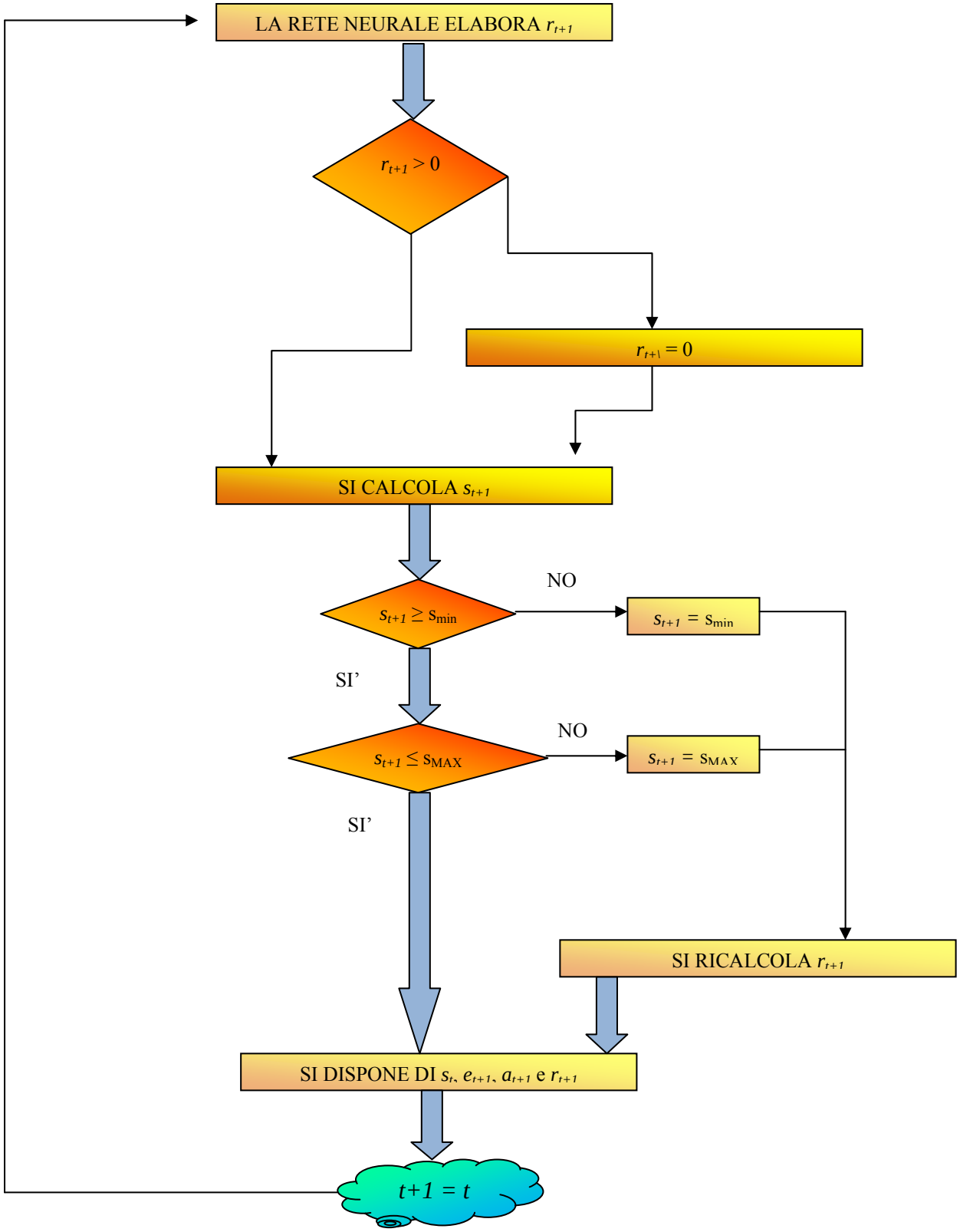

Figura 6.6 – Simulazione con le Reti Neurali

Ottenuti i rilasci dai diversi serbatoi da questo nuovo modello, è possibile calcolare il beneficio che ne consegue, utilizzando le stesse funzioni di utilità implementate in Aquafun (descritte nel Capitolo 4), facilmente replicabili in un foglio di calcolo Excel.
In Aquafun le condizioni di chiusura dei serbatoi venivano imposte (solitamente uguali a quelle di partenza), ma ciò non è possibile per tutti i modelli neurali, soprattutto per quelli che in addestramento non ricevono i dati di afflusso o non considerano un contatore temporale. Pertanto non è detto che queste simulazioni terminino con valori di invaso pari a quelli iniziali

(assegnati uguali alle condizioni iniziali per le simulazioni con Aquafun), quindi, per poter effettuare la comparazione tra i benefici prodotti da Aquafun e quelli prodotti dal modello neurale, si è reso necessario eseguire di nuovo il simulatore lineare, imponendo come condizione di chiusura agli invasi le medesime che si trovano con la rete neurale.

La procedura complessiva da seguire per poter confrontare i benefici prodotti dai modelli, è riassunta nella seguente Figura 6.7.

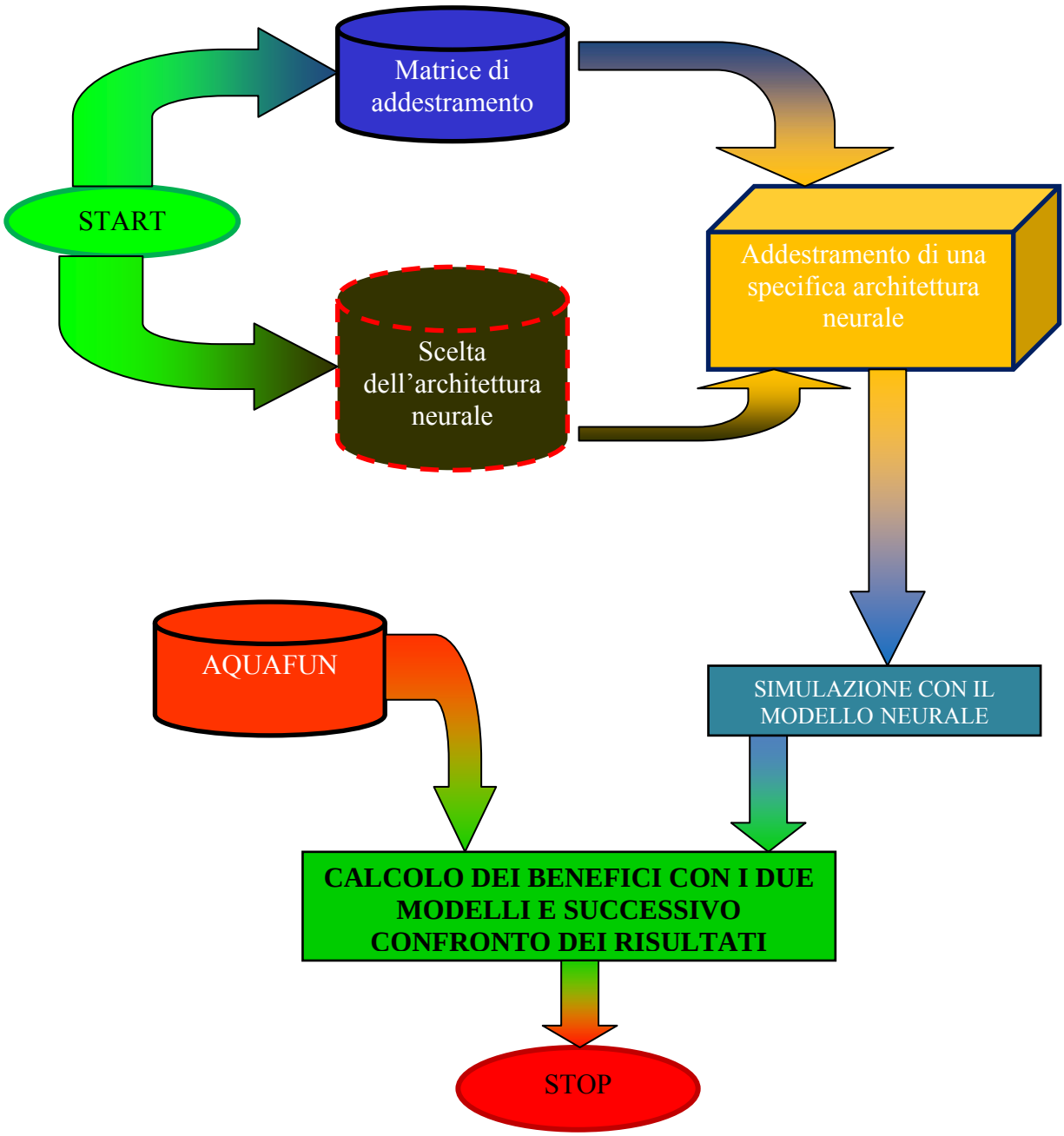

Figura 6.7 – Schema della procedura di utilizzo delle reti neurali per la simulazione della dinamica del sistema idrico

Diversi esperimenti eseguiti con il simulatore implementato in Matlab, hanno permesso di effettuare alcune considerazioni preliminari, che valgono per tutte le tipologie di reti neurali testate:

- I dati in ingresso alla rete neurale, sia in addestramento sia in simulazione, devono essere normalizzati, al fine di utilizzare dati con valori assoluti effettivamente gestibili

sia dalla memoria del sistema operativo, sia da Matlab stesso. La normalizzazione è stata effettuata utilizzando la formula classica:

$$x_{norm} = \frac{x-\mu_x}{\sigma_x} \quad (6.9),$$

in cui μ_x è la media dei valori e σ_x è la deviazione standard;

- I valori normalizzati possono essere utilizzati anche in simulazione, ottenendo dei risultati, anch'essi normalizzati, che devono essere ricondotti alla forma pura invertendo le stesse formule di normalizzazione, utilizzando però media e varianza dei dati di addestramento (mediante i quali si è ottenuta la forma dei pesi e degli errori finale). Definiti con w i pesi della rete e con b i bias (generici, non relativi ad uno specifico neurone), le forme normalizzate sono:

$$\begin{cases} w_{norm} = w \times \sigma_x \\ b_{norm} = w \times \mu_x + b \end{cases} \quad (6.10);$$

- Al fine di rendere la procedura di addestramento uniforme per tutte le reti testate, quando essa non fosse giunta a termine entro le *100* iterazioni, è stata arrestata manualmente. Questo induce senz'altro un'approssimazione, ma si è visto che le oscillazioni dei parametri entro questa epoca si stabilizzano, senza subire più eccessive variazioni, pertanto si è ritenuto di adottare un buon compromesso tra precisione raggiunta e durata dell'addestramento. La procedura di addestramento è quella di *early stopping*, descritta nel Capitolo 2;
- Essendo almeno quattro i dati di input (i soli invasi), si è ritenuto inutile addestrare reti con un numero di neuroni inferiore a quattro; per quanto riguarda invece il valore massimo dei neuroni si è deciso di procedere fino a quando i risultati non fossero equivalenti o non cominciassero a peggiorare.

6.6 RETI NEURALI ADDESTRATE DAI SOLI INVASI

6.6.1 RETE GLOBALE: TUTTI I CONTROLLI

La prima rete neurale che si è progettata è una rete neurale globale, intendendo con questa asserzione una rete che, avendo come input i soli dati degli invasi, presenta in output tutti i controlli per i serbatoi. La matrice di addestramento, nel caso in esame, è costituita da tutti e soli i valori di invaso. Si tratta pertanto di una matrice di input $[96000,4]$, che alimenta una rete con lo scopo di produrre, in uscita, i controlli per tutti e cinque i canali.
Per questa tipologia di rete neurale, il numero massimo di neuroni che possono essere implementati è pari a *10*: oltre questo valore, non è più possibile procedere per mancanza di sufficiente memoria per eseguire i calcoli (problema di *overflow di memoria*: il calcolatore non riesce ad elaborare contemporaneamente tutti i dati di addestramento, i pesi ed i bias di reti complesse).

Di seguito (Tabella 6.1.a) si riportano i risultati per la rete neurale che ha fornito i risultati migliori, ossia quella ad *8* neuroni.

	AQUAFUN	RETE NEURALE	VARIAZIONE [%]
RILASCI TOTALI [m^3/mese]	1521918	1502468	-1,28%
UTILITA' TOTALE	596996408	417280118	-30,10%

Tabella 6.1.a – Confronto tra rilasci totali e utilità totali calcolati da Aquafun e dalla Rete Neurale ad 8 neuroni

Di maggiore interesse è capire come si quantificano le perdite di utilità tra il modello neurale e quello lineare in ognuno degli anni di simulazione (escludendo il primo e l'ultimo per evitare effetti di bordo), come riportato nella seguente Tabella 6.1.b (valori medi).

ANNO	AQUAFUN	RETE NEURALE	DIFFERENZE [%]
II	3242857,50	4189650,08	-22,60%
III	2895440,67	4188979,42	-30,88%
IV	3060241,42	4190051,25	-26,96%
V	3135947,33	4189093,75	-25,14%
VI	2957303,83	4189449,08	-29,41%
VII	3095067,08	4189327,75	-26,12%
VIII	2990601,33	4189210,75	-28,61%
IX	2948311,42	4188644,50	-29,61%
X	3352898,08	4187920,67	-19,94%
XI	3028900,00	4112197,17	-26,34%
TOTALE	*3070756,87*	*4181452,44*	*-26,56%*

Tabella 6.1.b – Differenze di utilità medie annuali prodotte da Aquafun e della rete neurale

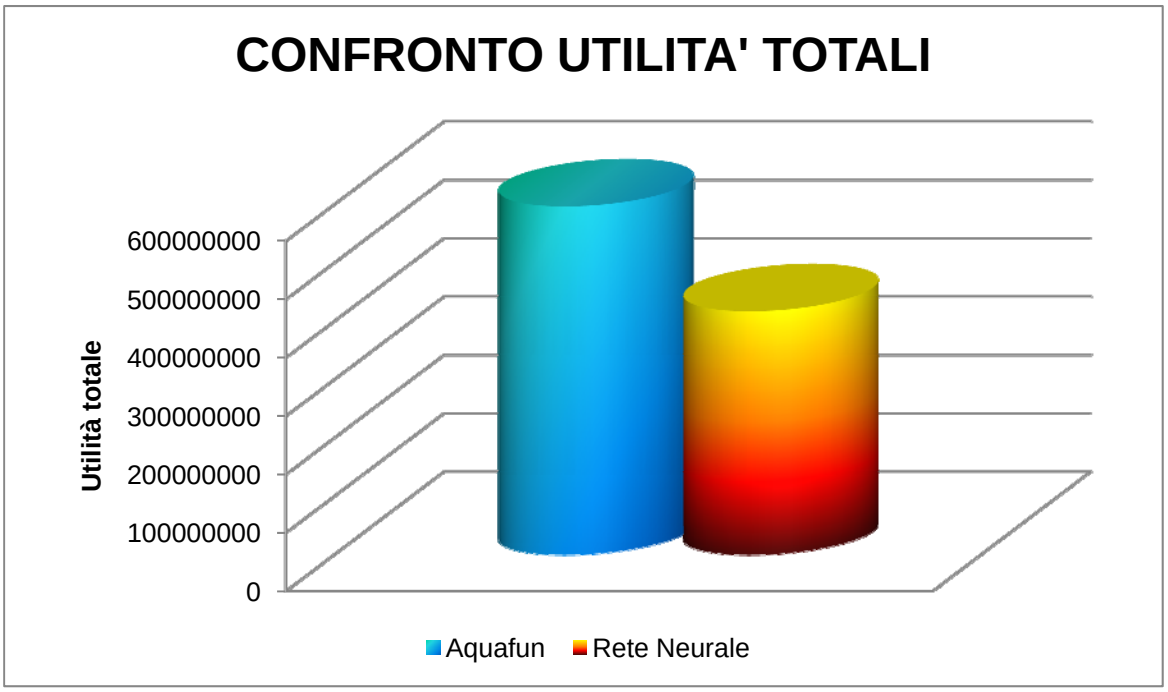

Figura 6.8 – Confronto grafico tra le utilità complessive

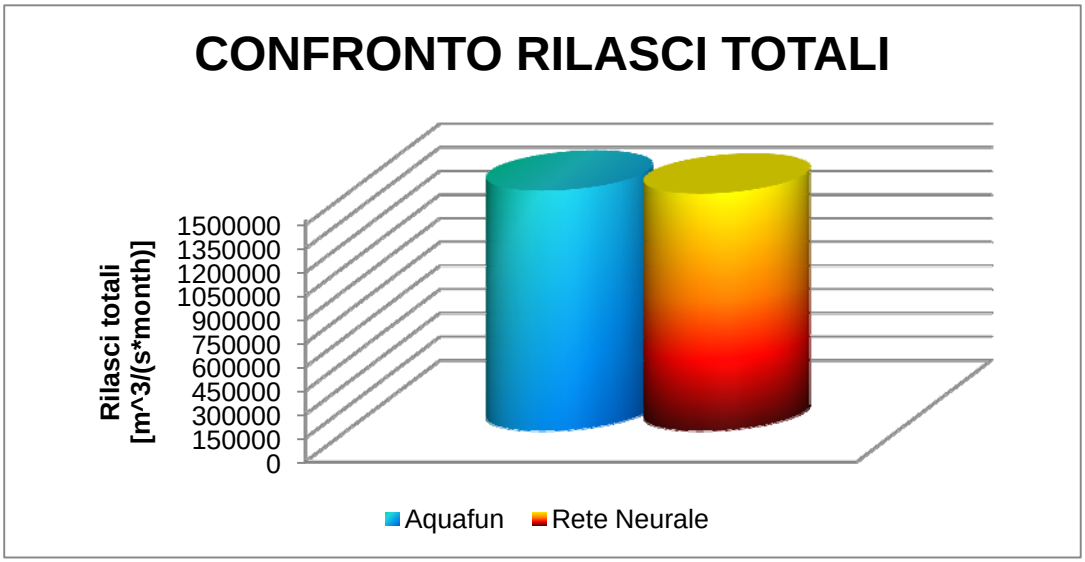

Figura 6.9 – Confronto grafico tra i rilasci complessivi

Come si evince dalla precedente Tabella 6.1.a, sebbene i rilasci complessivi elaborati dalla rete neurale siano molto prossimi a quelli di Aquafun, lo stesso non si può dire per l'utilità totale che ne consegue: ciò indica una cattiva ripartizione tra i rilasci nei *144* mesi della simulazione. Un'analisi più dettagliata aiuta a capire meglio le differenze tra i due modelli adottati.
Essendo il beneficio l'indicatore di interesse, i confronti ritenuti più significativi sono proprio quelli effettuati su questo parametro. Non bisogna, tuttavia dimenticare che anche i rilasci devono essere "sensati". Valori troppo piccoli, prolungati nel tempo, significano che l'acqua rimane nel bacino per un lungo periodo, causando, quasi sicuramente, siccità intorno e mancanza di acqua per gli scopi più disparati. Per converso, valori troppo grandi, seppur realizzabili per come è costruito Aquafun, denotano sfioro, o, peggio ancora, superamento di tale capacità, con conseguenze catastrofiche a valle facilmente immaginabili (tracimazioni ed esondazioni). È bene quindi effettuare un confronto anche tra i rilasci.

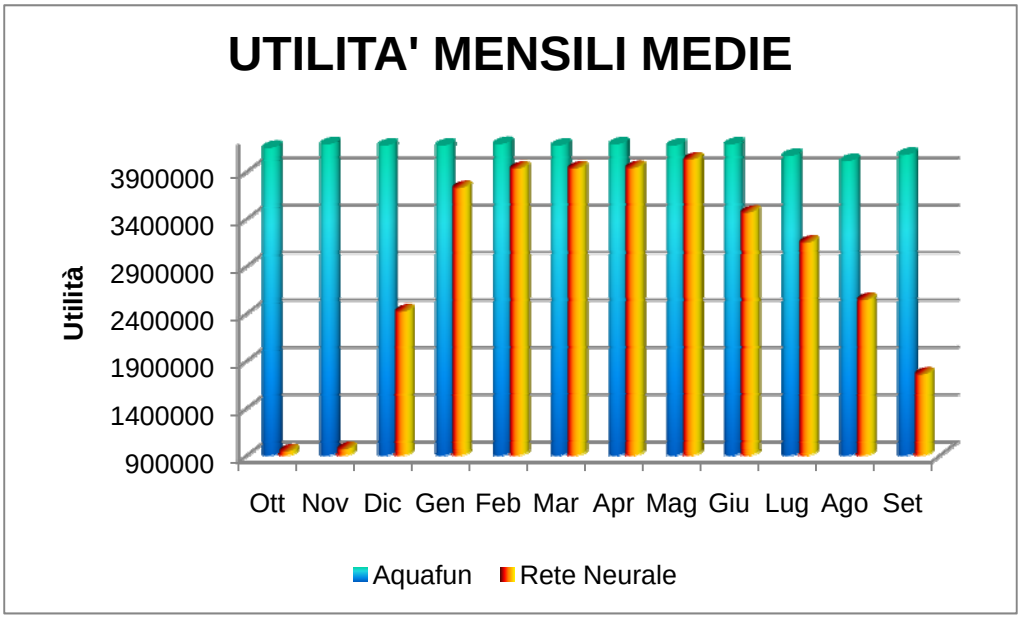

Figura 6.10 – Confronto tra le utilità medie mensili

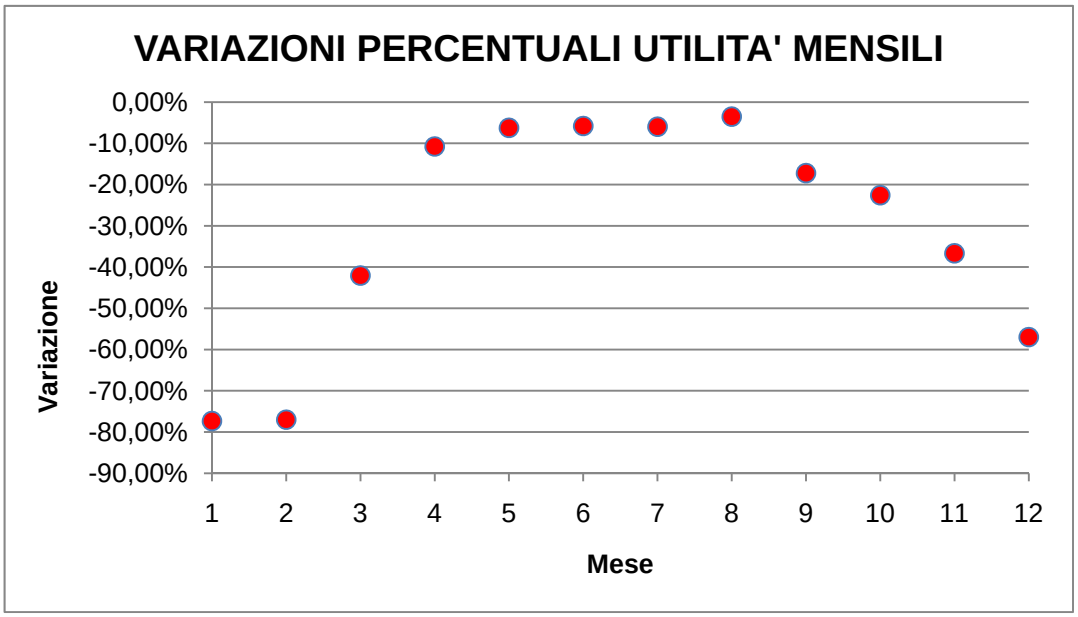

Grafico 6.1 – Variazioni percentuali tra la Rete Neurale ed Aquafun dell'utilità media mensile (il mese 1 corrisponde a Ottobre, il mese 12 a Settembre)

Dalle precedenti Figure 6.10 e Grafico 6.1, si nota che in alcuni mesi la perdita in termini di utilità è considerevole, mentre per altri essa è più ridotta. In particolar modo, discutendo degli afflussi al sistema nel Capitolo 5, si era detto che i mesi primaverili (tra Gennaio e Maggio) sono quelli più piovosi, mentre quelli autunnali (Ottobre – Dicembre), i più siccitosi; ciò trova conferma nei grafici discussi nel Capitolo. Si nota, tornando ai grafici soprastanti, che la perdita in termini di utilità nei mesi più umidi è ridotta, quasi nulla, rispetto ai mesi più secchi.
In particolare, è possibile notare una sostanziale differenza tra i due modelli: Aquafun è un modello parsimonioso, che tende ad ottimizzare i valori su tutto il periodo, per cui non ha interesse ad individuare un rilascio più consistente quando dispone di maggiore acqua ed a conservarla quando ne ha meno, ma tende ad uniformarli lungo tutto l'orizzonte in modo da conseguire il massimo possibile; la rete neurale, invece, è un modello più "ingordo": non distinguendo tra i diversi periodi, infatti, ad ogni passo calcola un rilascio in base ai pesi ottenuti in addestramento, senza preoccuparsi di cosa accadrà al sistema in futuro.
Analizzando invece le utilità medie complessive derivate da ognuno dei serbatoi produttivi, si ottengono dei risultati variabili a seconda del bacino che si considera. Le seguenti Figure 6.11 e Grafico 6.2 forniscono un aiuto nella comprensione.

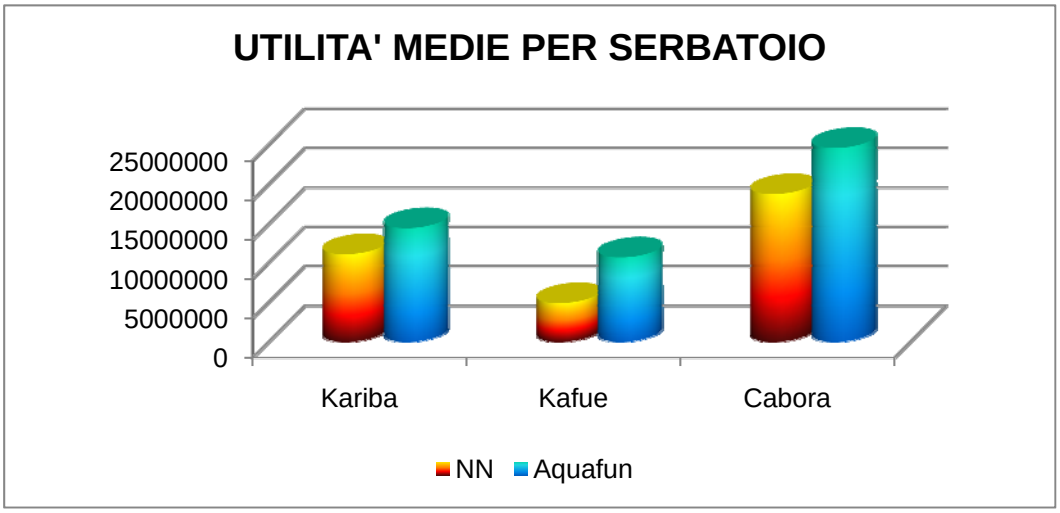

Figura 6.11 – Utilità medie complessive ottenute dai due modelli per i tre serbatoi produttivi

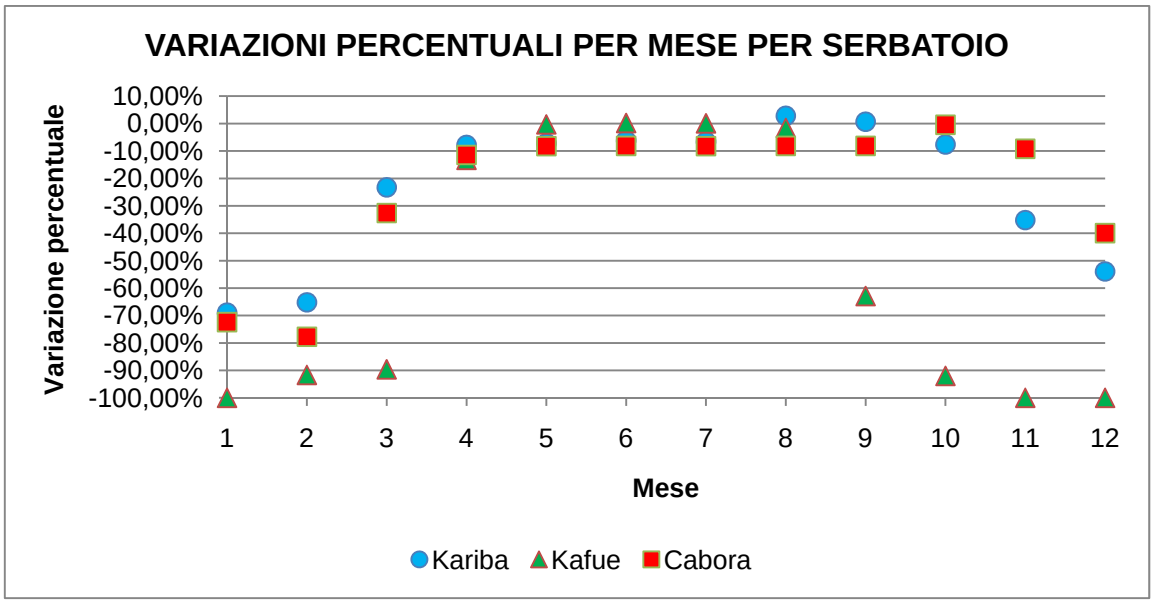

Grafico 6.2 – Variazioni percentuali tra l'utilità calcolata da Aquafun e quella calcolata con la Rete Neurale in ogni mese per ognuno dei serbatoi produttivi

Sebbene il modello neurale, complessivamente, generi utilità minori per tutti e tre i serbatoi (Figura 6.11), l'analisi delle differenze percentuali mensili (Grafico 6.2) mostra come ogni serbatoio si comporti in modo differente, pur con andamento analogo: in particolare, Kariba è il bacino con perdite minori, mentre Kafue è quello che ha perdite complessive maggiori. Nei mesi più piovosi, tuttavia, la rete neurale approssima al meglio il comportamento di Aquafun per il serbatoio di Kafue. Nella seguente Tabella 6.2 si riportano le variazioni percentuali medie dei deflussi da ogni serbatoio per mese ottenuti applicando i due differenti modelli.

MESE	Var%_Ka	Var%_Kf	Var%_Cb
Ott	-68,96%	-100,00%	-72,39%
Nov	-65,19%	-91,66%	-77,75%
Dic	-23,28%	-89,61%	-32,64%
Gen	-7,81%	-13,25%	-11,45%
Feb	-7,19%	-0,33%	-8,24%
Mar	-6,17%	0,18%	-8,17%
Apr	-6,57%	0,05%	-8,25%
Mag	2,83%	-1,61%	-8,15%
Giu	0,64%	-62,87%	-8,15%
Lug	-7,64%	-92,00%	-0,44%
Ago	-35,19%	-100,00%	-9,19%
Set	-53,93%	-100,00%	-39,95%
TOTALI	*-22,94%*	*-54,25%*	*-23,79%*

Tabella 6.2 – Variazioni percentuali medie mensili dei deflussi da ogni serbatoio

La Tabella 6.2 avvalora quanto già discusso analizzando il grafico precedente: il serbatoio di Kafue è il peggiore, avendo, in corrispondenza di Ottobre, Agosto e Settembre perdite pari al *100%* di deflussi, corrispondenti ad un rilascio pari al valore nullo. Questa situazione, ovviamente, non è fisicamente sensata e realizzabile, giacché non è possibile, per diversi mesi all'anno, non rilasciare alcun volume di acqua, e in altri rilasciarne quantità estremamente elevate.
In conclusione, pertanto, una rete neurale di questo tipo non può essere utilizzata per gestire il sistema idrico del fiume Zambesi.

Se, altresì, si analizza l'andamento degli invasi risultanti dalle differenti simulazioni (Aquafun con le condizioni iniziali e finali identiche, la Rete Neurale, Aquafun con le condizioni di chiusura uguali a quelle della rete), si vede come il modello neurale non sia indicato per gestire un simile sistema (Grafici 6.3 – 6.6).

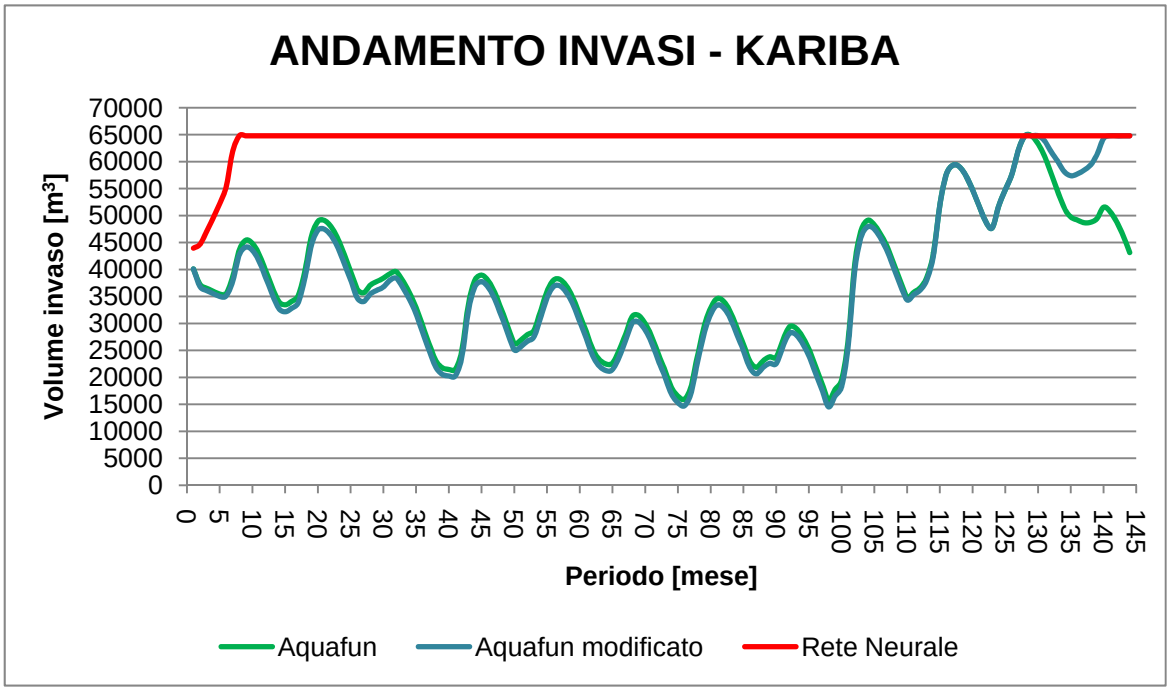

Grafico 6.3 – Andamento del volume di invaso per il serbatoio di Kariba nelle tre diverse simulazioni

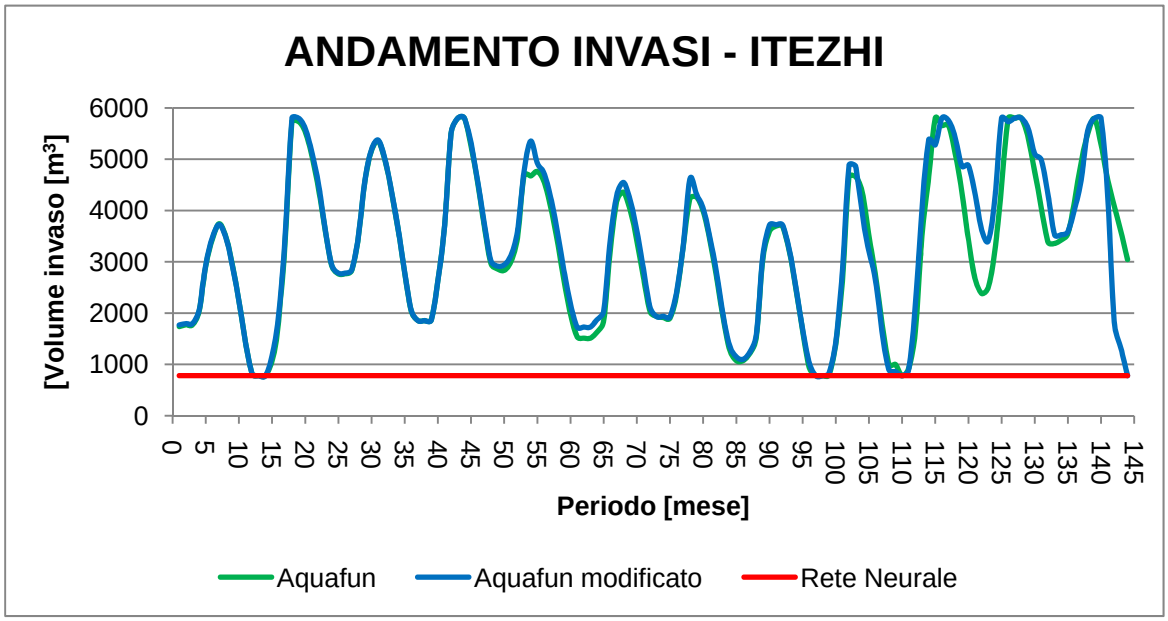

Grafico 6.4 – Andamento degli invasi per il serbatoio di Itezhitezhi

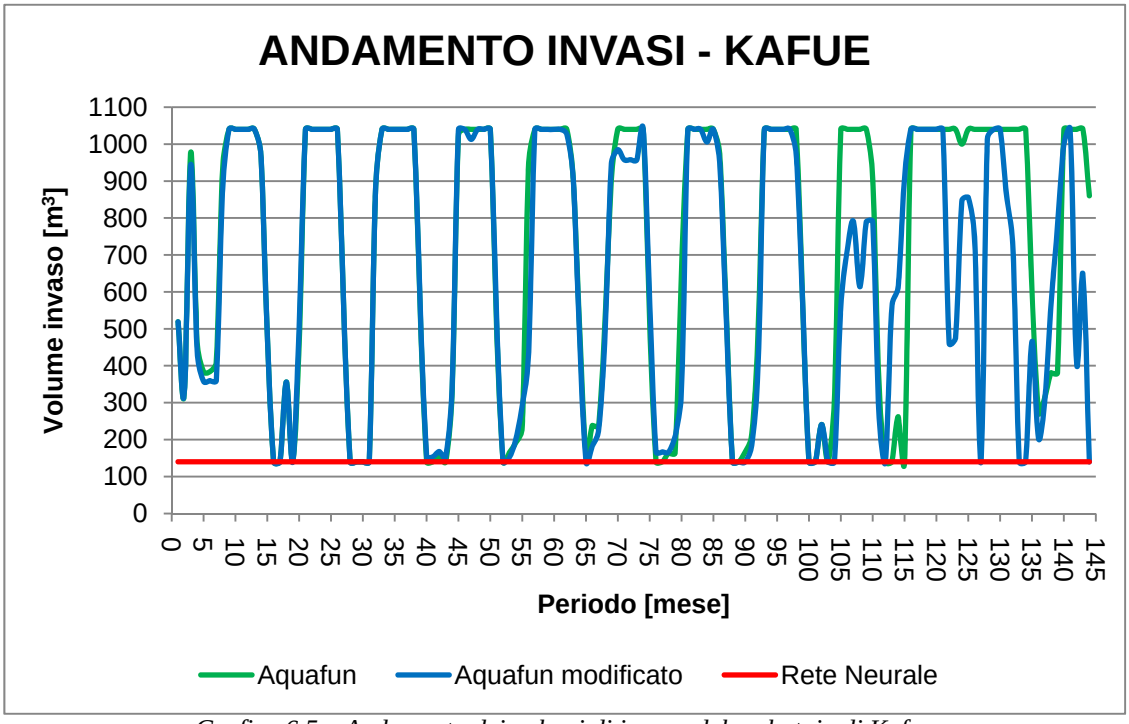

Grafico 6.5 – Andamento dei volumi di invaso del serbatoio di Kafue

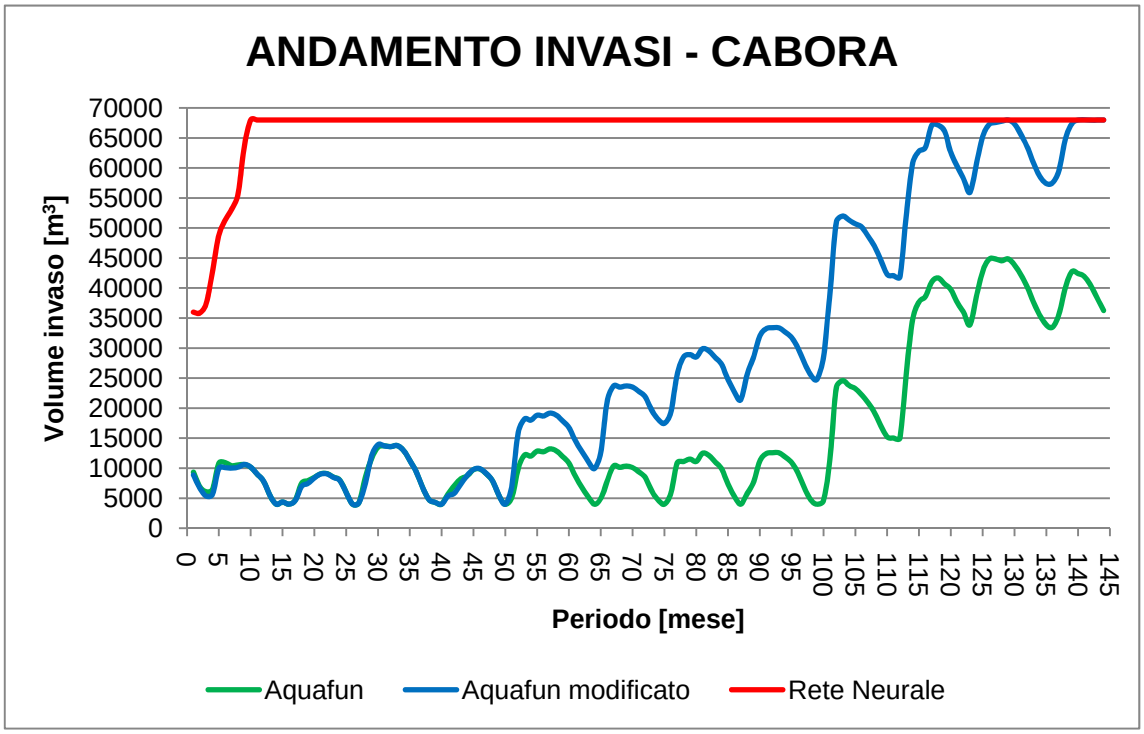

Grafico 6.6 – Andamento degli invasi per il serbatoio di Cabora

Nei grafici precedenti, i valori dei volumi di invaso determinati dal modello neurale sono rappresentati in rosso. Come si vede, essi si discostano molto dalla simulazione eseguita con Aquafun, in entrambi i casi. In nessun caso, infatti, la Rete Neurale tende a conservare l'acqua quando ne dispone di scarse quantità. Ciò conduce a risultati assurdi: i serbatoi meno capienti (Itezhi e Kafue) vengono svuotati rapidamente, e mantenuti al minimo volume di invaso, mentre i serbatoi maggiori vengono portati al massimo invaso, ed ivi mantenuti. L'acqua sarà prevalentemente prelevata da questi bacini, lasciando gli altri con un rilascio nullo per buona parte del tempo. Le considerazioni qui esposte consentono di analizzare meglio anche le

utilità prodotte da ogni serbatoio nelle diverse simulazioni: a parte il serbatoio di Itezhi, che non produce utilità, dai grafici precedenti (Figura 6.12 e Grafico 6.13), si osserva come l'utilità ottenibile con una gestione neurale è migliore per Kariba e Cabora, mentre per Kafue è pessima lungo tutto l'orizzonte: nei bacini maggiori si dispone di più acqua, quindi è possibile ripartirla meglio per produrre una maggiore quantità di energia elettrica e, di conseguenza, un beneficio più elevato. Complessivamente, dunque, nei mesi più piovosi la rete neurale si comporta meglio, proprio perché, essendo un modello che agisce passo – passo, rilascia subito la maggior quantità di risorsa possibile, disinteressandosi del futuro.

6.6.2 ADDESTRAMENTI SUCCESSIVI PER RETE NEURALE UNICA

Il primo strumento sperimentato per ridurre il problema dell'*overflow* di memoria è consistito nell'addestrare una rete neurale in modo progressivo.
In dettaglio, la matrice complessiva di addestramento [96000,4] è stata suddivisa in dieci sottomatrici, prendendo gli elementi in modo progressivo, in modo che risultassero nove tabelle [10000,4], ed una, più piccola [6000,4]. Ciascuna di esse è stata fornita in input alla rete, in modo progressivo, in maniera tale da effettuare l'addestramento per step successivi, aggiornando, con ogni nuova matrice, il valore dei pesi e degli errori per tutti i neuroni. Quello che si è fatto, praticamente, è consistito nell'addestrare una sola rete per dieci volte, prendendo ogni volta una matrice di input diversa e come valore iniziale dei parametri quello ottenuto dallo step precedente.
Con questo *modus operandi* il problema dell'overflow è stato completamente risolto, rendendo definibili architetture con un numero molto grande di neuroni (sebbene, sensatamente, non si sono addestrate reti che contenessero oltre *30* neuroni, al fine di evitare un eccessivo *overfitting*). I risultati, tuttavia, si sono dimostrati differenti a seconda del numero di neuroni implementati nella Rete Neurale.
Sebbene non generalizzabili in termini assoluti, i risultati mostrano un andamento analogo al caso precedente, con una stima relativamente migliore per i mesi centrali del periodo, più umidi, e peggiore per quelli più secchi, estremi al periodo.
Nel complesso, comunque, le reti neurali addestrate con questa procedura si rivelano pessime, in termini di utilità generata rispetto ad Aquafun, sebbene non dissimili in termini di rilasci.

		RILASCI TOTALI [m^3/(s*mese)]	UTILITA' TOTALI	VARIAZIONI [%]	
NN5	AQUAFUN	0	0	*non valutabile*	*non valutabile*
	RETE NEURALE	1472534	402073181		
NN10	AQUAFUN	1680068	604144522	-0,87%	-76,95%
	RETE NEURALE	1665472	139276775		
NN15	AQUAFUN	0	0	*non valutabile*	*non valutabile*
	RETE NEURALE	1613878	436600558		
NN20	AQUAFUN	0	0	*non valutabile*	*non valutabile*
	RETE NEURALE	1548523	426605059		
NN25	AQUAFUN	1638504	604374695	-1,07%	-59,68%
	RETE NEURALE	1620999	243706701		
NN30	AQUAFUN	1679929	604281062	-1,11%	-70,07%
	RETE NEURALE	1661363	180886769		

RILASCI TOTALI	UTILITA'

Tabella 6.3 – Confronto tra rilasci totali ed utilità complessive, ottenute da reti neurale addestrate in modo progressivo

Non avendo problemi di memoria, si è scelto di addestrare reti neurali con numero di neuroni crescente, da *5* a *30* con passo *5*, con lo scopo primario di osservare se, modificando l'architettura, si avessero dei miglioramenti significativi in termini di risultati, osservazione che non è stato possibile effettuare con un unico addestramento, proprio per il problema di mancanza di memoria.
La precedente Tabella 6.3 riporta i rilasci e le utilità totali ottenute con Aquafun e la rete neurale, in ognuno dei casi testati, e le variazioni percentuali delle prestazioni del modello neurale nei confronti di quello lineare: sebbene i rilasci, globalmente, si equivalgano, lo stesso non può dirsi per le utilità. Qualunque modello neurale si adotti, il beneficio che se ne ottiene è molto scarso. È bene osservare come esistano alcuni casi in cui si ha un risultato per la rete neurale e un valore nullo per Aquafun: in questi casi, la rete neurale porta a delle condizioni di chiusura che Aquafun non riesce a gestire. Il risultato è che, per quanto cattiva, la politica ottenuta con il modello neurale è migliore di quella ottenibile con il simulatore lineare.
Analizzando meglio i risultati riportati nella precedente Tabella 6.3, la migliore rete neurale risulta quella con *25* neuroni nello strato nascosto, che produce un beneficio minore del *60%* rispetto ad una simulazione lineare con le stesse condizioni di chiusura.

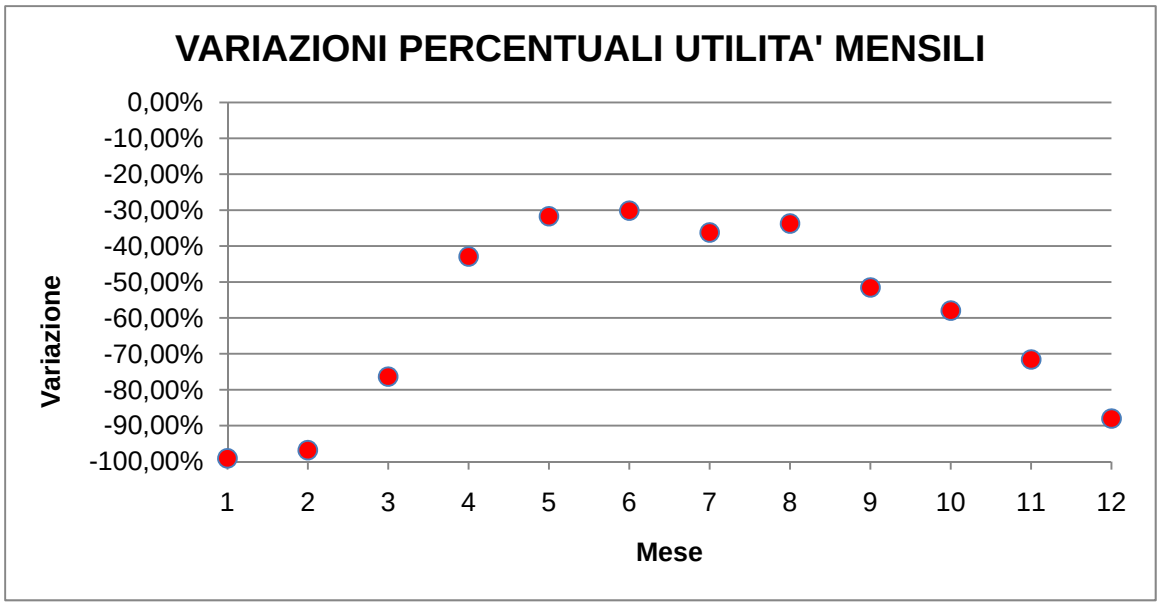

Grafico 6.7 – Perdite percentuali in termini di utilità rispetto al modello lineare per la rete a 25 neuroni

Essendo i benefici linearmente dipendenti (a tratti) dai rilasci, il precedente Grafico 6.7 consente di capire perché il modello neurale non è efficace: in alcuni mesi, prevalentemente quelli secchi, le perdite sono prossime all'unità, denotando assenza quasi totale di rilascio di acqua nei canali a valle dei bacini. Ancora una volta, si nota il trend caratteristico di miglior approssimazione per i mesi più umidi, e di peggior simulazione per i serbatoi più piccoli, come mostrato nella seguente Figura 6.12, in cui colpisce subito che Kariba genera un beneficio molto basso, e quindi il modello prevede rilasci complessivamente molto scarsi lungo tutto l'orizzonte temporale (in alcuni mesi esso è addirittura nullo, in conseguenza del quale il volume di invaso aumenta; saltuariamente ci sono dei rilasci enormi, che svuotano il serbatoio). Essendo uno dei serbatoi più capienti, si conclude rapidamente che il modello simuli in un modo fisicamente non accettabile. Una motivazione potrebbe essere che la rete neurale, anche in questo caso, è addestrata male, per cui durante la simulazione si ottengono risultati cattivi, che comportano rilasci nulli per molti mesi. Ovviamente, ciò è insensato.

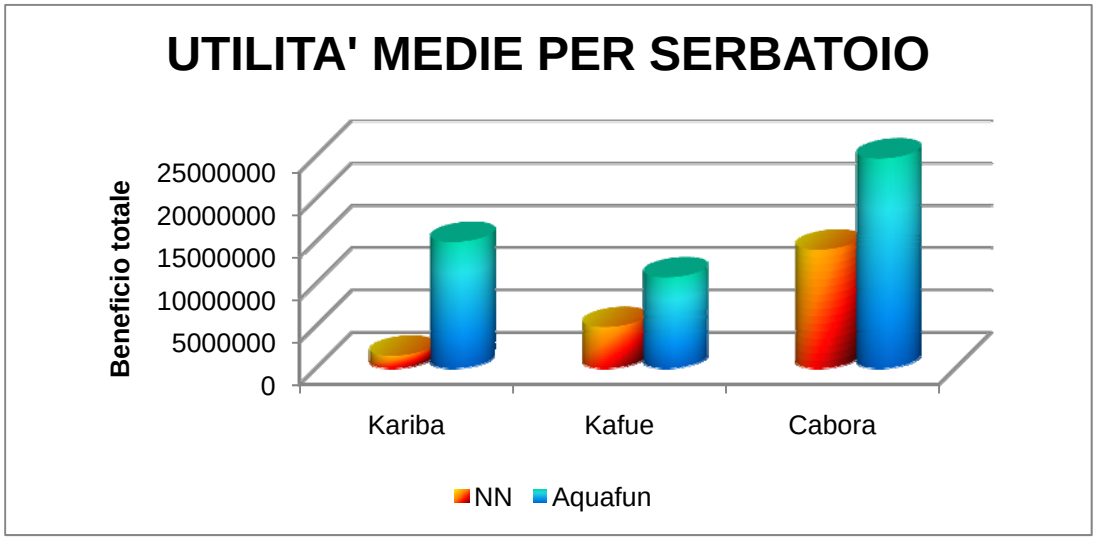

Figura 6.12 – Utilità medie per serbatoio: confronto tra Aquafun e Rete Neurale a 25 neuroni

Disporre di una rete i cui neuroni calcolano tutte le uscite, significa avere molti pesi e molti bias probabilmente prossimi a zero, perché ciascun neurone, o gruppo di essi, si specializza per una sola uscita, quella che riesce a descrivere meglio.

6.6.3 RETI NEURALI PER UN SOLO RILASCIO

Un modello che dispone di una rete neurale per ognuna delle uscite del sistema dovrebbe essere meno complesso, giacché ciascuna di esse, pur ricevendo in ingresso gli stessi dati, ossia gli invasi, dovrà calcolare una sola uscita, riducendo molto la complessità dell'architettura, e rendendo più agile il calcolo. In pratica, spariscono dall'architettura tutti i pesi ed i bias riferiti agli altri controlli: ogni rete è più specializzata, ed anche la generalità (intesa come bontà di risposta al variare delle situazioni di ingresso) dovrebbe essere più alta.
Dal punto di vista teorico, ognuna di queste reti è indipendente dalle altre, per cui le procedure di addestramento dovrebbero essere incorrelate tra loro. In effetti, ciascun rilascio dovrebbe essere descritto da una specifica architettura neurale, con un numero di neuroni indipendente da quello delle reti che descrivono le altre uscite. Tuttavia, come riportato nella seguente Tabella 6.4, per tutti i rilasci, al crescere dei neuroni anche le prestazioni di addestramento migliorano:

		RILASCIO				
# NEURONI		KARIBA	ITEZHI	KAFUE 1	KAFUE 2	CABORA
10	MSE	0,73624	0,90730	0,98797	0,88583	0,69250
	R	0,47778	0,33299	0,17313	0,30770	0,50940
15	MSE	0,80961	0,87292	0,97697	0,94911	0,78487
	R	0,47931	0,34990	0,18606	0,32929	0,50743
20	MSE	0,80047	0,89862	0,98138	0,78516	0,79060
	R	0,48123	0,35043	0,18182	0,31365	0,45329
25	MSE	0,89341	0,88584	0,95483	0,80240	0,73778
	R	0,48181	0,34892	0,18809	0,34526	0,51278
30	MSE	0,69580	0,85583	0,94363	0,81387	0,75479
	R	0,48259	0,35372	0,18665	0,34524	0,50332

Tabella 6.4 – Risultati dell'addestramento per le reti neurali ad un singolo rilascio; con Kafue 1 *si intende il rilascio verso il pozzo, con* Kafue 2 *si intende quello verso il serbatoio di valle, Cabora;* MSE *è il parametro che calcola il valor medio del quadrato degli scarti, laddove* R *è il coefficiente di correlazione; in verde sono evidenziate le migliori prestazioni*

Si osserva che, generalmente, gli addestramenti non sono soddisfacenti, dal momento che il coefficiente di correlazione tra le uscite estratte da Aquafun, e quelle calcolate dal modello neurale, non si porta mai su valori molto alti, mentre il parametro di errore (*MSE*) degli scarti non scende mai sotto valori elevati, sebbene sia il parametro da minimizzare. Questo trend, tuttavia, non è un problema di questo specifico esperimento, ma accompagna tutti gli addestramenti e le simulazioni compiute. La causa, molto probabilmente, è da ricercarsi in due fattori: anzitutto il modello implementato in Aquafun è, come probabilmente il vero sistema idrico, molto sovradimensionato alla realtà, per cui anche quando si verificano afflussi molto intensi, o molto scarsi, sia per durata, sia per intensità, il sistema risponde senza discostarsi significativamente dal comportamento che mantiene in casi normali; in secondo luogo, probabilmente, esiste un errore insito nelle serie sintetiche di afflussi, tale da alterarne la periodicità e la correlazione tra i diversi valori.
In questo caso la simulazione con il modello neurale deve essere condotta in modo differente, nel senso che ogni serbatoio deve essere simulato con la rete meglio addestrata. I risultati riportati nella precedente Tabella 6.4 mostrano che i rilasci da Kariba, Itezhi e da Kafue verso il pozzo sono meglio descritti da un'architettura neurale a *30* neuroni, Kafue verso Cabora da una a *20*, mentre il rilascio da Cabora è meglio descritto da una rete a *10* neuroni. Ciò è

dovuto, quasi sicuramente, alle differenti complessità degli andamenti dei rilasci dai singoli bacini, che richiedono proprio diverse architetture per essere simulati al meglio.
I risultati rispecchiano, ancora una volta, gli andamenti già trovati nelle sperimentazioni precedenti.

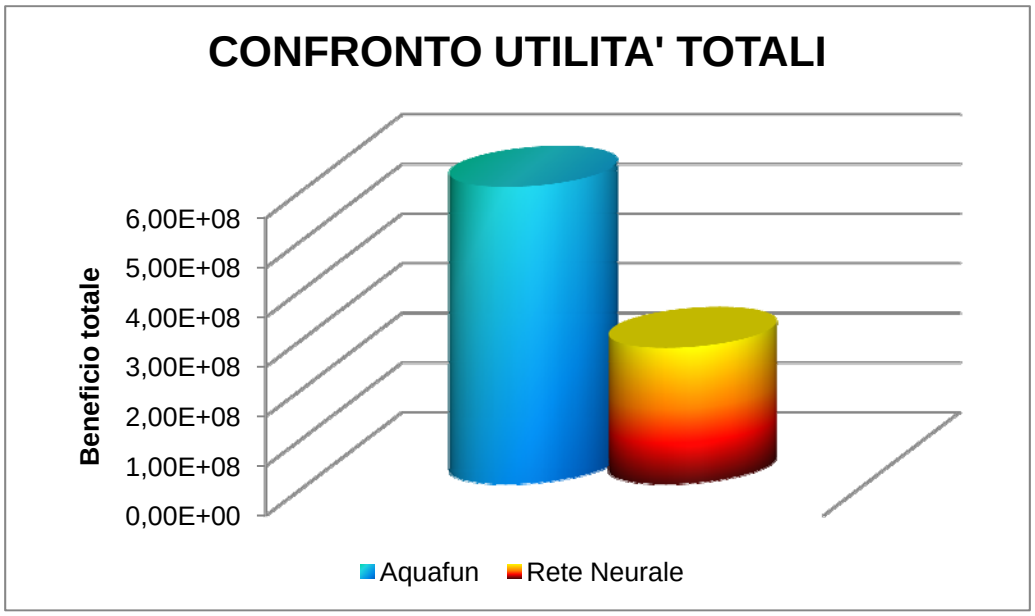

Figura 6.13 – Confronto tra le utilità totali prodotte da Aquafun e dal modello neurale

	UTILITÀ TOTALE	RILASCIO TOTALE [m^3/(sec*mese)]
RETE NEURALE	274326029	1566913
AQUAFUN	598732515	1582853
VARIAZIONE [%]	*-54,18%*	*-1,01%*

Tabella 6.5.a – Confronto tra rilasci ed utilità totali ottenuti dal modello lineare e da quello neurale, nel caso di reti ad un solo rilascio

ANNO	AQUAFUN	RETE NEURALE	DIFFERENZE [%]
II	2114160,75	4189507,42	-49,54%
III	1883159,75	4190219,00	-55,06%
IV	1909322,33	4190193,92	-54,43%
V	2132760,08	4189800,08	-49,10%
VI	1794177,25	4189093,75	-57,17%
VII	2087455,58	4189093,75	-50,17%
VIII	1744234,33	4189093,75	-58,36%
IX	1838967,67	4187431,42	-56,08%
X	2322095,17	4187751,50	-44,55%
XI	1903604,42	4105814,25	-53,63%
TOTALE	*1972993,73*	*4180799,88*	*-52,81%*

Tabella 6.5.b – Differenze di utilità medie annue tra i due modelli

Anche in questo caso i risultati del modello neurale non sono completamente soddisfacenti, soprattutto in termini di utilità, che è circa la metà di quella ottenibile simulando il sistema con Aquafun. Ancora una volta, la perdita maggiore si ha nei mesi più secchi (Figura 6.14, di

seguito), e per il serbatoio di Kafue (Figura 6.15, di seguito), che si rivela quello gestibile con maggiori difficoltà, forse perché dispone di due canali di uscita, uno dei quali (quello verso il nodo intermedio), peraltro, non contribuisce in alcun modo al beneficio, anzi presenta rilasci costanti funzione solamente del numero di giorni del mese in cui ci si trova.

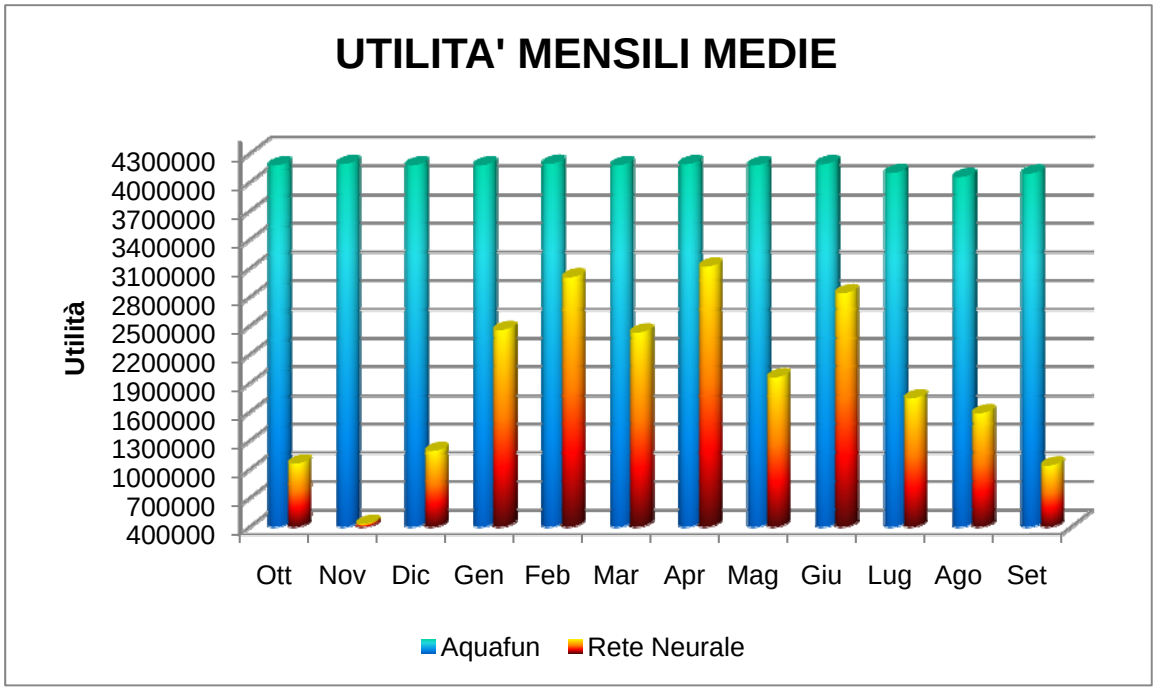

Figura 6.14 – Differenze tra le utilità medie mensili

Rispetto alle analisi effettuate in precedenza (Figura 6.10 e Grafico 6.1), nei mesi più secchi le perdite sono leggermente più contenute, ma, nel contempo sono molto più consistenti nei mesi più umidi, dove è possibile dire che il modello neurale opera sensibilmente peggio rispetto ad Aquafun.

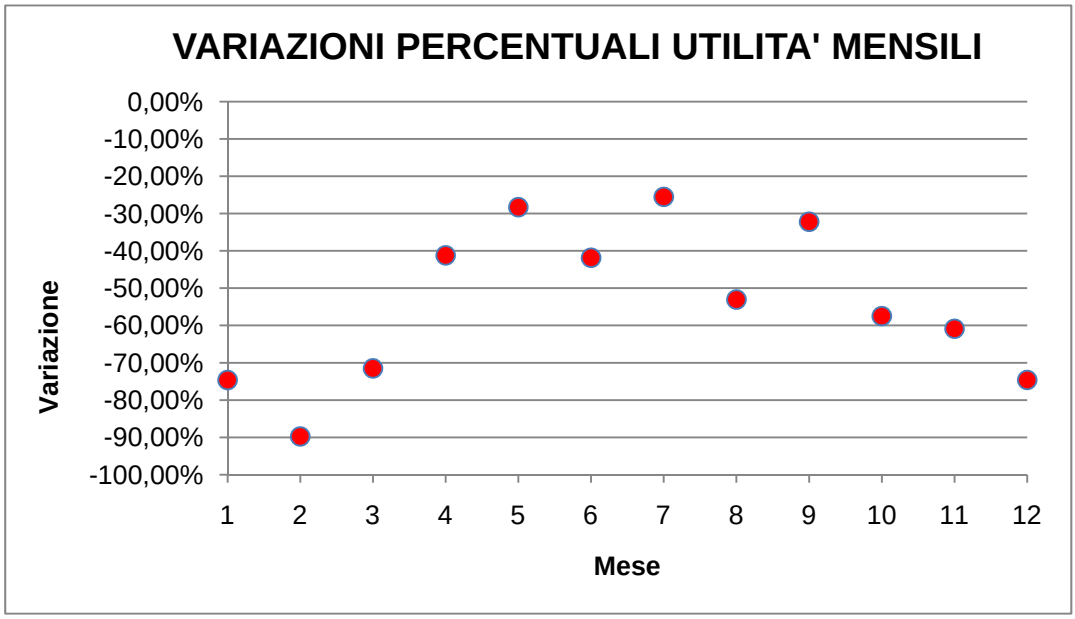

Grafico 6.8 – Perdite percentuali di utilità del modello neurale in ogni mese

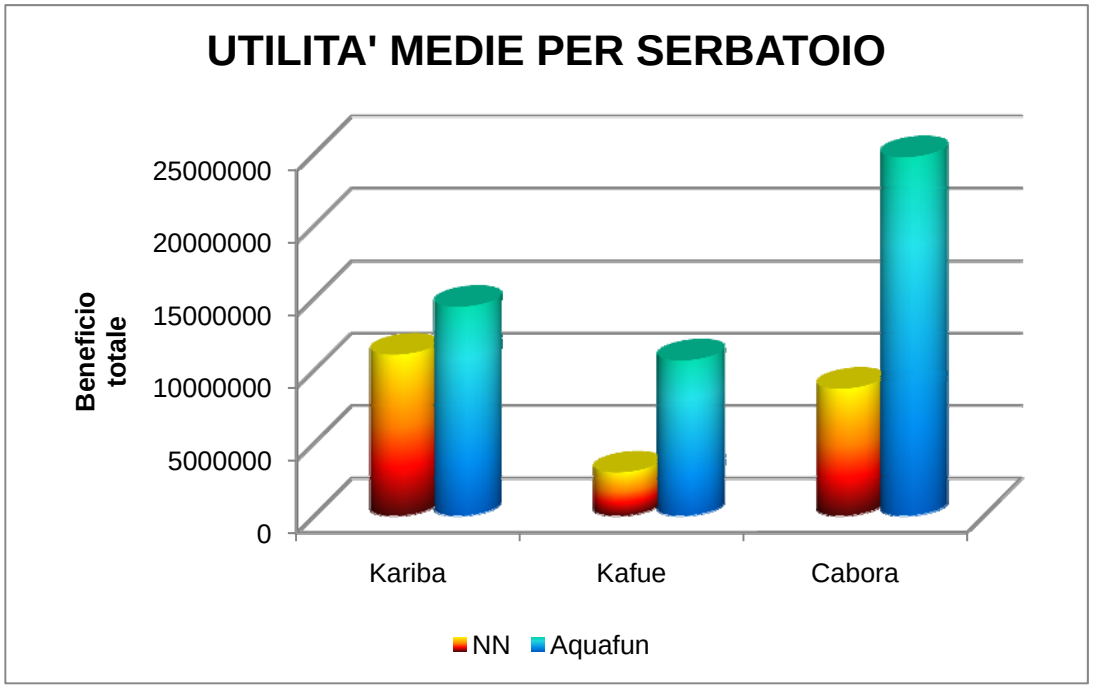

Figura 6.15 – Differenze di utilità per ognuno dei tre serbatoi produttivi

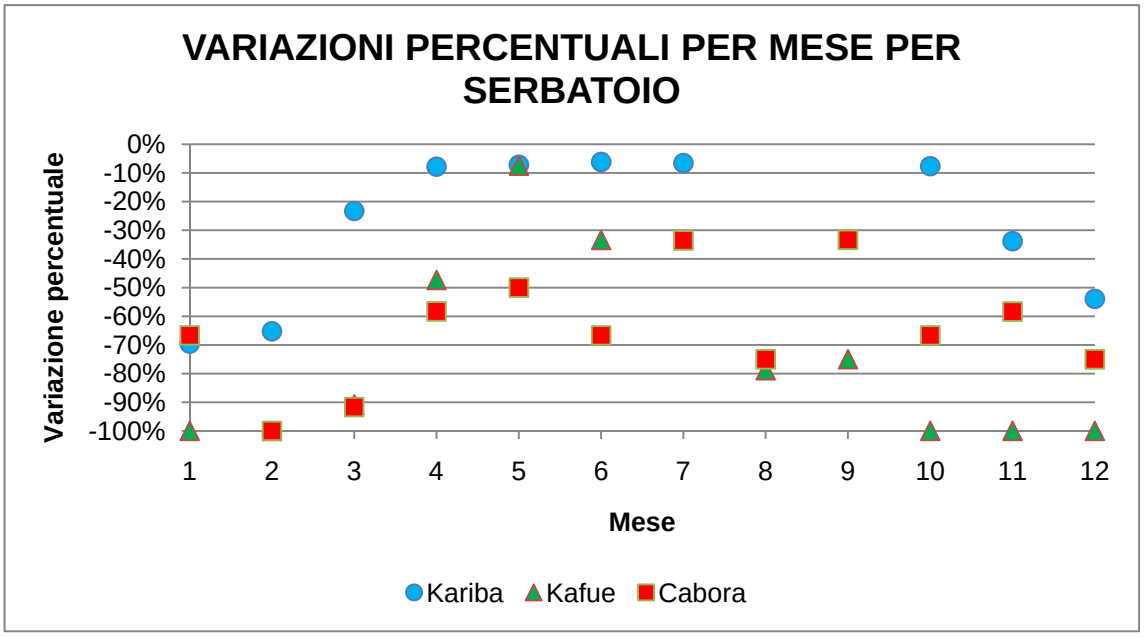

Grafico 6.9 – Perdita di utilità percentuale in ogni serbatoio

Un rapido confronto tra il precedente Grafico 6.9 e l'analogo Grafico 6.2 permette di concludere che Kariba è sempre il serbatoio il cui comportamento riesce ad essere riprodotto meglio, mentre Kafue è il peggiore. Con questo modello nemmeno il comportamento di Cabora riesce ad essere riprodotto in modo ottimale.

Fino a questo punto sono stati provati differenti modelli neurali, nessuno dei quali ha portato al risultato sperato, ossia riuscire a replicare le traiettorie del modello lineare. Le cause di questo fallimento possono derivare da diversi problemi:

- Il sistema idrico del Fiume Zambesi, come già detto, è molto sovradimensionato, per cui nessuna serie di afflussi, per quanto molto elevata, è realmente critica;
- Esiste un qualche problema nelle serie sintetiche costruite, che altera eccessivamente il comportamento fisico del sistema;
- Le reti neurali non sono un buon modello per descrivere la successione delle leggi di controllo di un sistema come quello in esame.

6.7 INCLUSIONE DI UN CONTATORE TEMPORALE

Si è visto che tutte le tipologie di rete neurale viste sino ad ora non sono in grado di riprodurre sufficientemente bene la sequenza dei controlli prodotta da Aquafun, e quindi il beneficio che se ne ricava è considerevolmente inferiore.
Si deve tenere presente, tuttavia, che tutti i modelli neurali considerati fino a questo punto non riescono a considerare né il tempo, né, tantomeno, la periodicità del sistema dello Zambesi. Ad esempio, non si è potuto considerare il fatto che la condizione di chiusura, imposta in fase di input ad Aquafun, è uguale alla condizione iniziale del sistema.
L'unico metodo per insegnare alla rete neurale che esiste una periodicità, e che i valori di input sono correlati a quelli immediatamente precedenti e successivi molto più fortemente che non a valori più distanti, è addestrarla includendo tra gli stessi input anche i dati relativi al tempo.
Se si immagina il tempo come un punto che ruota intorno ad una circonferenza con le leggi del moto circolare uniforme, esso è ben descritto da semplici formule che lo identificano univocamente nel piano:

$$\begin{cases} x_{punto} = \cos\frac{2\times\pi\times t}{T} \\ y_{punto} = \sin\frac{2\times\pi\times t}{T} \end{cases} \quad (6.11),$$

Dove *t* è il passo temporale, e *T* rappresenta invece l'ampiezza del periodo considerato; in questo caso

$$\begin{cases} t = 1 \div 144 \\ T = 12 \end{cases} \quad (6.12),$$

perché la durata della simulazione è pari a *144* periodi (sebbene in addestramento debbano essere esclusi il primo e l'ultimo anno, quindi $t_{train} = 13 \div 132$), mentre ogni *12* passi si ritrova lo stesso mese. La seguente Figura 6.16 mostra come ogni *12* passi ci si ritrovi sempre al medesimo punto. I punti consecutivi rappresentano mesi successivi, tra cui il legame è ovviamente più forte.

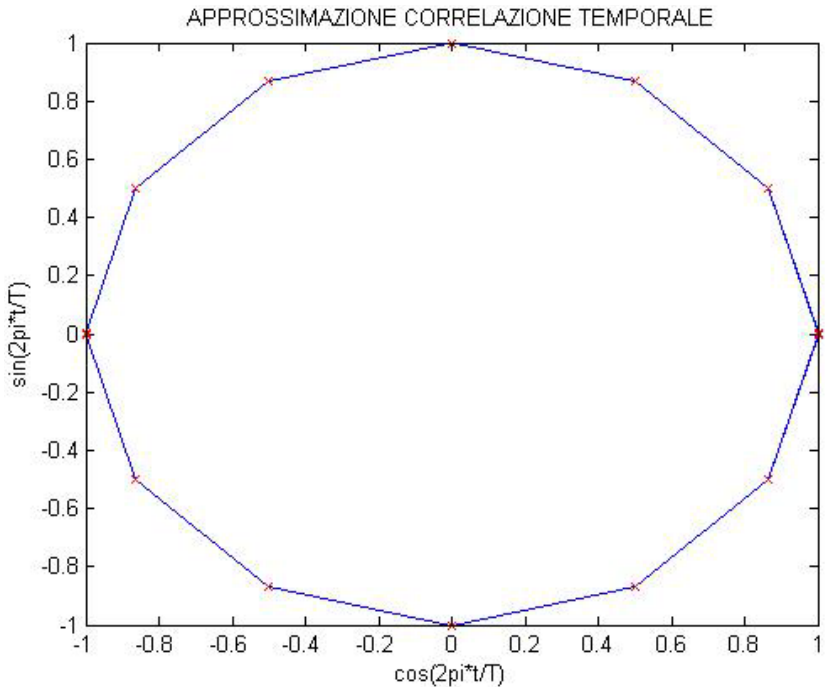

Figura 6.16 – Rappresentazione del legame temporale tra mesi successivi, tramite un'approssimazione del moto circolare uniforme

Considerare il tempo con l'approccio qui descritto non fa riferimento ad alcuna metodologia standard adottata in altre ricerche, ma piuttosto fa riferimento ad un nuovo filone di ricerca sulle reti neurali, sviluppato anche all'interno dei pacchetti di software libero Mastrave. [mastrave.org]
La matrice di addestramento per questa tipologia di rete neurale contiene, ovviamente, non solo i dati di invaso, ma anche quelli di afflusso (sono le grandezze realmente correlate nel tempo) e un indice costruito basandosi sulle precedenti equazioni (6.11) e (6.12), con cui è possibile collegare istanti più vicini e più lontani nel tempo.
Di tutte le serie sintetiche costruite (come spiegato nel Capitolo 5), quelle che seguono un andamento simile alla serie storica sono le sequenze create utilizzando il modello di forma simile ad un *AR(1)*, mentre il modello *AR(0)* genera dati senza nessuna correlazione temporale: queste ultime non possono essere utilizzate per addestrare questo tipo di matrici, mentre è possibile usare le prime serie, le quali, appunto, contengono al loro interno una struttura tempo – variante.
La matrice di addestramento, riprendendo le simbologie già adottate all'inizio del presente Capitolo e nel Capitolo 3, avrà dimensioni

$$\begin{cases} d = 120 \times 200 = 24000 \\ m = 4 + 4 + 2 = 10 \end{cases} \quad (6.13),$$

in quanto si usano soltanto le *200* serie affini ad un modello auto regressivo di primo ordine, ciascuna di *120* periodi, e i dati di quattro invasi, quattro afflussi e due indici temporali (seno e coseno del punto rotante su una circonferenza).

6.7.1 RETI PER TUTTI I CONTROLLI

Le prime sperimentazioni compiute riguardano reti neurali addestrate per fornire in uscita tutti e cinque i controlli contemporaneamente.
Come già in precedenza, sono state addestrate architetture a *5*, *10*, *15*, *20*, *25*, e *30* neuroni, utilizzando dati normalizzati. Sebbene non si siano verificati problemi di alcun tipo in fase di addestramento, solamente le reti a cinque neuroni si sono potute utilizzare in simulazione, dal momento che l'addestramento delle altre architetture ha portato a valori dei pesi e dei bias che non possono essere utilizzati in simulazione, perché troppo elevati (probabilmente per un problema di overfitting che si presenta al crescere del numero di neuroni).
A prescindere dall'usabilità effettiva, la qualità dell'addestramento migliora di molto rispetto ai casi precedenti, sia in termini di coefficiente di correlazione (è molto elevato), sia di errore medio quadratico, come mostrato nella seguente Tabella 6.6.

# NEURONI		TUTTI I CONTROLLI
5	MSE	0,49615
	R	0,72023
10	MSE	0,42976
	R	0,75060
15	MSE	0,38022
	R	0,79007
20	MSE	0,34587
	R	0,80301
25	MSE	0,30093
	R	0,82050
30	MSE	0,33541
	R	0,82395

Tabella 6.6 – Parametri di addestramento reti per tutti i controlli

Se si analizzano i risultati in termini di utilità prodotti dalla simulazione con rete a cinque neuroni, si ottengono dei risultati molto più confortanti rispetto alle simulazioni passate. In particolare, si osserva una cospicua riduzione della perdita di utilità, che si attesta su un valore inferiore a circa il *28%*, come riportato nella seguente Tabella 6.7.a.

AQUAFUN		RETE NEURALE	VARIAZIONE [%]
1585918	**RILASCI TOTALI [m^3/(s*mese)]**	1566613	-1,22%
598732515	**UTILITÀ TOTALE**	431102151	-28,00%

Tabella 6.7.a – Risultati in termini di rilasci ed utilità dopo simulazione con rete neurale a 5 neuroni

Anche l'analisi delle utilità medie prodotte ogni anno (Tabella 6.7.b) mette in risalto la diminuzione delle perdite tra i due differenti modelli.

ANNO	AQUAFUN	RETE NEURALE	DIFFERENZE [%]
II	3242857,50	4188410,50	-22,58%
III	2895440,67	4190219,00	-30,90%
IV	3060241,42	4190051,25	-26,96%
V	3135947,33	4189444,75	-25,15%
VI	2957303,83	4189566,08	-29,41%
VII	3095067,08	4189093,75	-26,12%
VIII	2990601,33	4189093,75	-28,61%
IX	2948311,42	4189767,08	-29,63%
X	3352898,08	4186681,08	-19,92%
XI	3028900,00	4099120,50	-26,11%
TOTALE	*3070756,87*	*4180144,78*	*-26,54%*

Tabella 6.7.b – Confronto tra utilità medie annuali prodotte da Aquafun e dalla rete neurale

Decisi miglioramenti si trovano anche nei valori mensili e nei valori per serbatoio (Grafico 6.10 e Figura 6.17)

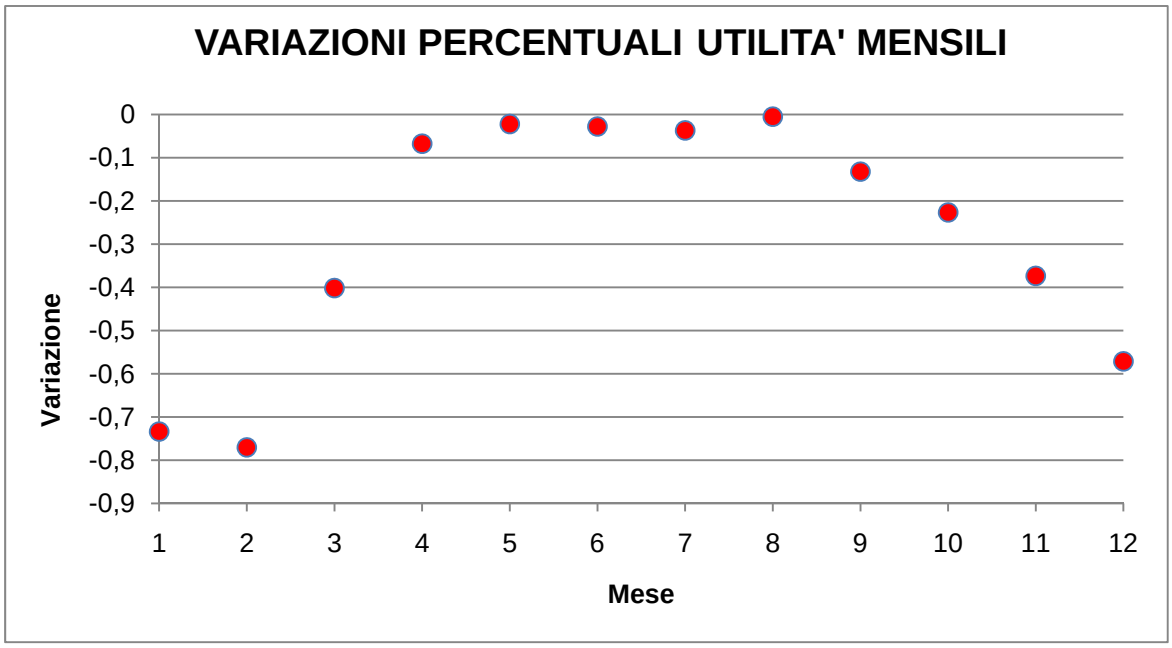

Grafico 6.10 – Perdite percentuali di utilità per ogni mese tra la simulazione neurale e quella lineare

L'andamento delle perdite è sempre il medesimo, ossia più concentrate nei mesi più secchi, ma con questo tipo di rete neurale è molto più lungo il periodo in cui i valori di utilità sono prossimi a quelli ottenuti con modello lineare; per converso, le perdite massime sono più contenute.

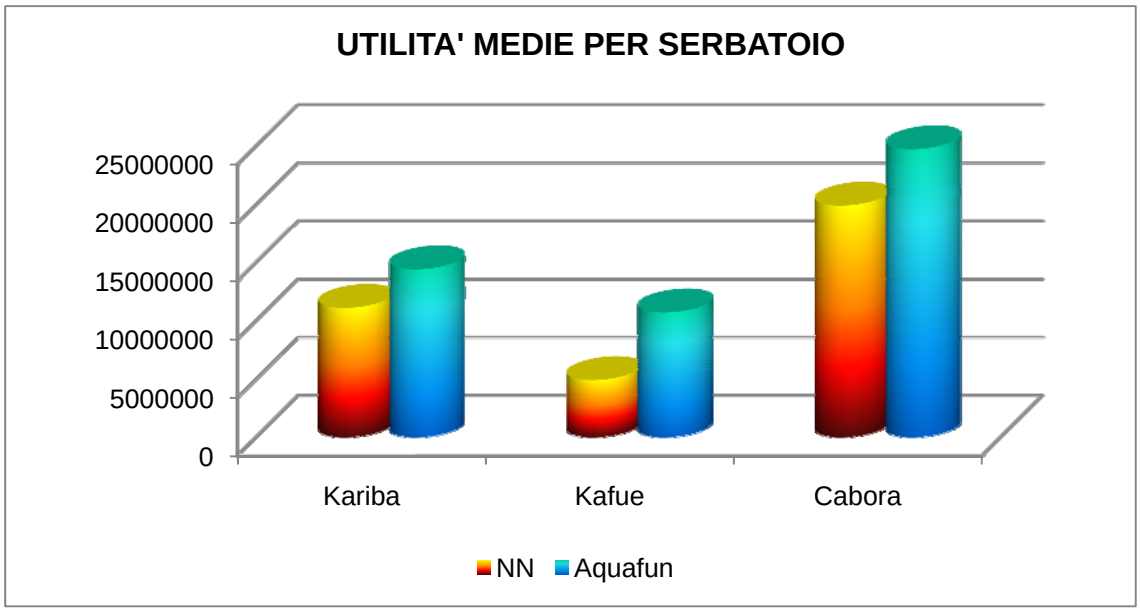

Figura 6.17 – Differenze tra utilità per serbatoio nel modello neurale ed in Aquafun

Sebbene rimanga l'usuale trend per cui i serbatoi di Kariba e di Cabora vengono approssimati meglio dal modello neurale, in questo caso l'approssimazione è molto più buona che nelle simulazioni precedenti; anche il serbatoio di Kafue, pur rimanendo il peggiore, presenta perdite minori rispetto ai modelli discussi nei Paragrafi precedenti. Il seguente Grafico 6.11, in termini percentuali, aiuta a comprendere meglio l'entità delle perdite.

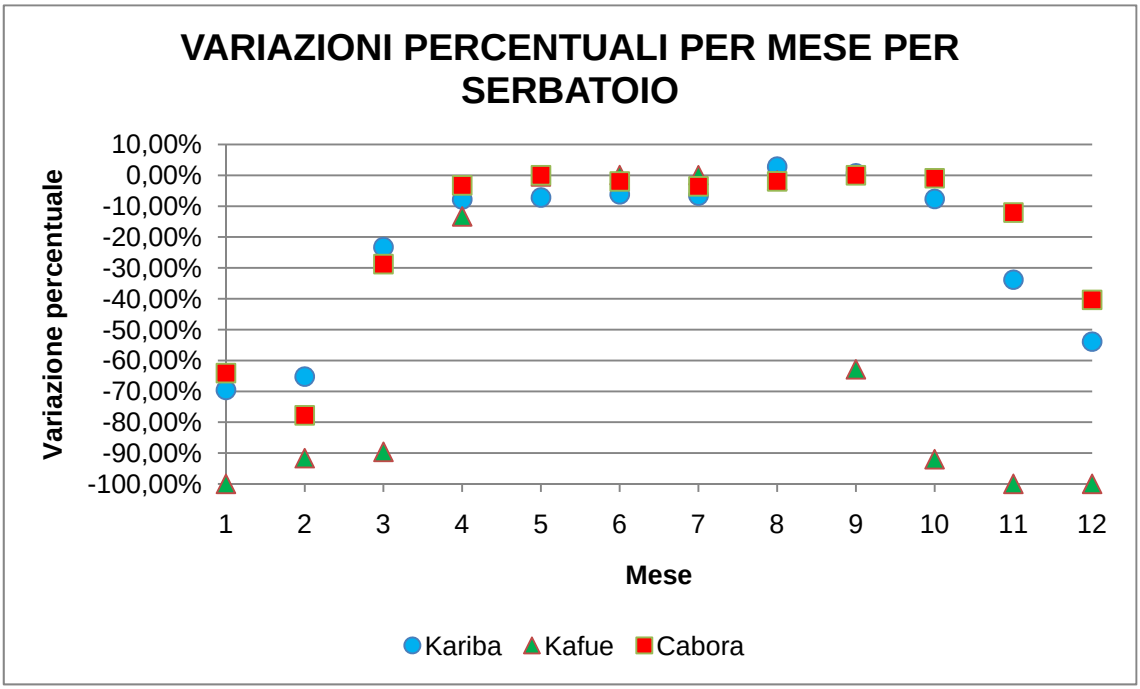

Grafico 6.11 – Riduzioni percentuali di utilità per serbatoio per mese simulando con il modello neurale

Nei mesi centrali, quelli più umidi, anche il serbatoio di Kafue viene stimato con una precisione elevata. Da notare, per i serbatoi di Kariba e Cabora, che le massime perdite non scendono mai al di sotto del *70 – 80%*, mentre con le architetture neurali discusse in precedenza, esse si attestavano anche al *100%*.

6.8 RETI NEURALI ADDESTRATE CON DATI GIORNALIERI

Il problema del cattivo addestramento delle reti neurali per questo sistema può dipendere anzitutto dal fatto che il sistema dello Zambesi è fisicamente molto sovradimensionato, per cui tutte le serie di afflusso generate, anche quelle che contengono valori molto elevati, non portano mai a gestioni critiche. È bene ricordare che gli obiettivi con cui si addestra l'architettura neurale sono rappresentati dai rilasci calcolati con Aquafun, per cui se esso produce sempre il medesimo risultato, qualunque sia l'entità di acqua in ingresso, il modello neurale non riuscirà a capirne le differenze, e produrrà anche esso sempre lo stesso risultato, inficiando la qualità dell'addestramento.
Per ottenere i valori medi giornalieri è sufficiente dividere il valore mensile di afflusso che si ha a disposizione per il numero di giorni del mese, in modo tale da far risaltare le differenze dovute unicamente al diverso numero di giorni in ogni mese.

6.8.1 RETI COMPLETE: TUTTI I RILASCI

Al solito, le prime sperimentazioni riguardano reti neurali addestrate per produrre tutti e cinque i controlli insieme.
Le reti hanno un'architettura con un numero di neuroni variabile da *6* a *12*, con passo *2* (oltre si incorre, ancora una volta, nel problema di overflow).

Se si osservano i risultati dell'addestramento (Tabella 6.8 di seguito), si nota come, ancora una volta, non si ottengano dei buoni risultati.

# NEURONI	MSE	R [%]	R_tr [%]	R_val [%]
6	0,91350	0,31276	0,31345	0,30880
8	0,93508	0,31786	0,31925	0,31165
10	0,89878	0,35209	0,35407	0,35034
12	0,85930	0,35944	0,36100	0,35925

Tabella 6.8 – Risultati dell'addestramento; MSE *indica lo scarto quadratico medio,* R *il coefficiente di correlazione totale,* R_tr *sulla serie di addestramento,* R_val *sulla serie di validazione*

Rimangono, per tutte le reti, dei coefficienti di correlazione molto bassi e degli scarti molto lontani da *0*, tuttavia pare che la rete a *12* neuroni sia quella meglio addestrata. Non è così, dal momento che le migliori prestazioni in simulazione si ottengono con un'architettura più semplice, mentre è proprio la rete neurale più complessa ad avere le prestazioni peggiori, come riportato nella seguente Tabella 6.9.a. Questo fatto può dipendere dal fenomeno di overfitting, in cui, al crescere del numero di neuroni, la rete perde di generalità.

		RILASCI TOTALI [m3/(s*mese)]	UTILITA' TOTALI	VARIAZIONI [%]	
NN6	AQUAFUN	1715418	604525039	-12,41%	-30,97%
	RETE NEURALE	1502468	417280118		
NN8	AQUAFUN	1521918	596996406	-1,28%	-30,10%
	RETE NEURALE	1502468	417280118		
NN10	AQUAFUN	1715418	604525039	-1,27%	-21,58%
	RETE NEURALE	1693621	474049071		
NN12	AQUAFUN	1702504	604389442	-1,73%	-54,68%
	RETE NEURALE	1673129	273886345		
				RILASCI TOTALI	UTILITA'

Tabella 6.9.a – Risultati delle simulazioni per i diversi modelli neurali

La rete neurale a sei neuroni sottostima i rilasci in misura considerevole, mentre l'architettura con *12* neuroni causa delle perdite complessive in termini di utilità troppo elevate. La rete neurale migliore appare essere quella a *10* neuroni, che produce, rispetto ad Aquafun, delle perdite di utilità molto contenute.
Nella seguente Tabella 6.9.b si riportano le utilità medie calcolate dai due modelli per ognuno degli anni di simulazione, eccezion fatta per gli anni estremi, esclusi per evitare gli effetti di bordo.

ANNO	AQUAFUN	RETE NEURALE	DIFFERENZE [%]
II	3727917,17	4189650,08	-11,02%
III	2895440,67	4190219,00	-30,90%
IV	3245235,62	4193942,25	-22,62%
V	3233741,31	4198506,92	-22,98%
VI	2957470,83	4197800,58	-29,55%
VII	3175550,51	4197800,58	-24,35%
VIII	2990601,33	4197800,58	-28,76%
IX	3233940,50	4206181,83	-23,11%
X	3889151,14	4208181,17	-7,58%
XI	3616238,47	4205108,17	-14,00%
TOTALE	*3296528,76*	*4198519,12*	*-21,48%*

Tabella 6.9.b – Differenze nel calcolo delle utilità annuali medie con i due modelli

Il modello neurale ben stima le utilità annuali, soprattutto in quegli anni, come ad esempio il *X* del periodo considerato, più umidi. Comunque, anche negli anni più secchi, il modello neurale riesce a limitare le perdite nel calcolo dell'utilità.

Se si guarda, in effetti, alle differenze di utilità, mese per mese (Grafico 6.12 di seguito), si nota anzitutto l'andamento già riscontrato nelle analisi precedenti, ossia una migliore stima per i mesi umidi ed una peggiore per quelli secchi.

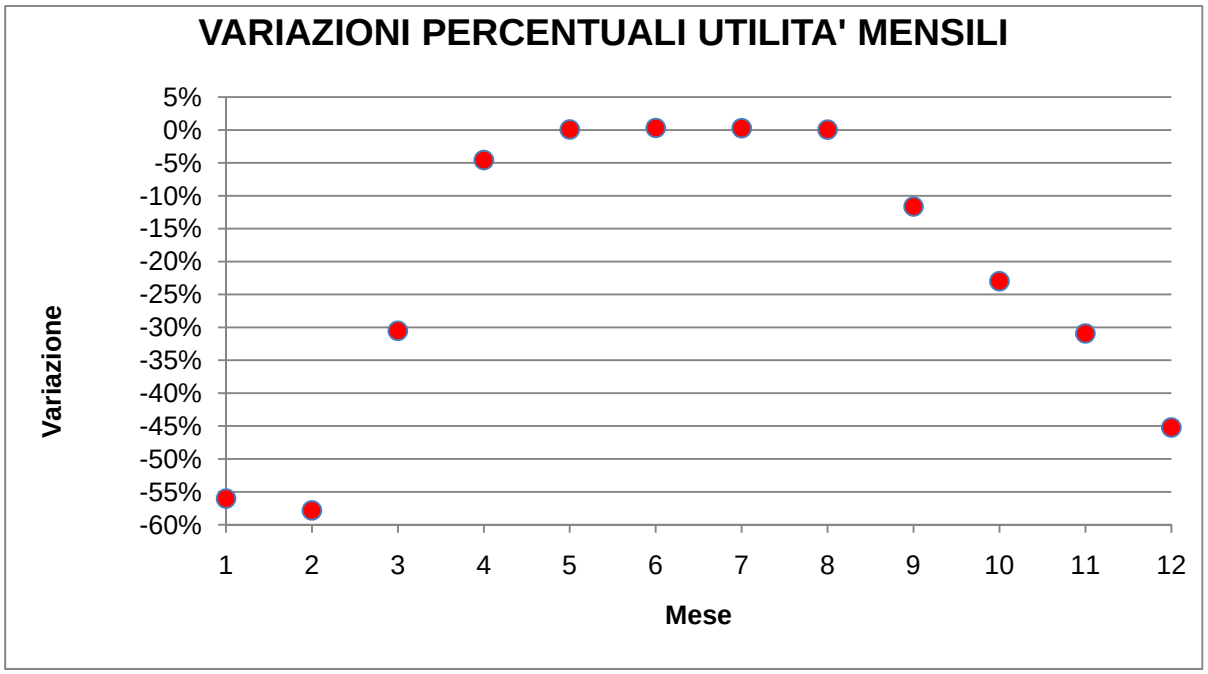

Grafico 6.12 – Perdite di utilità medie per mese, espresse in percentuale, tra la rete neurale ed Aquafun

Nei mesi più secchi, rispetto alle precedenti simulazioni, le perdite sono assai più contenute, mentre l'utilità complessiva calcolata con il simulatore neurale è molto prossima a quella calcolata con Aquafun, sia nei mesi umidi, come già riscontrato, sia nei mesi più asciutti, dove le differenze sono meno marcate, come mostrato nella seguente Figura 6.18.

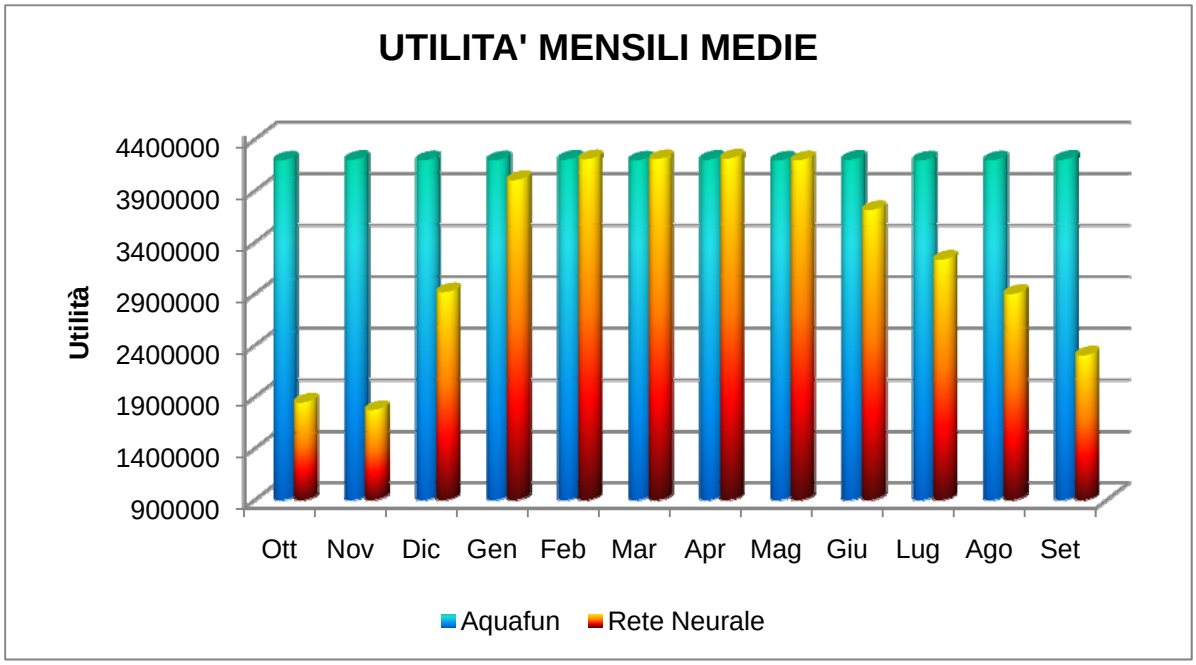

Figura 6.18 – Differenze di utilità media tra il modello lineare e quello neurale

Analizzando i risultati per i singoli serbatoi, si rispecchia il trend già trovato nelle precedenti sperimentazioni: il serbatoio di Kafue è quello i cui rilasci vengono calcolati peggio dalla rete neurale, mentre sia Kariba sia Cabora vengono simulati bene. La seguente Figura 6.19 riporta le utilità totali generate dai tre serbatoi con i due differenti modelli.

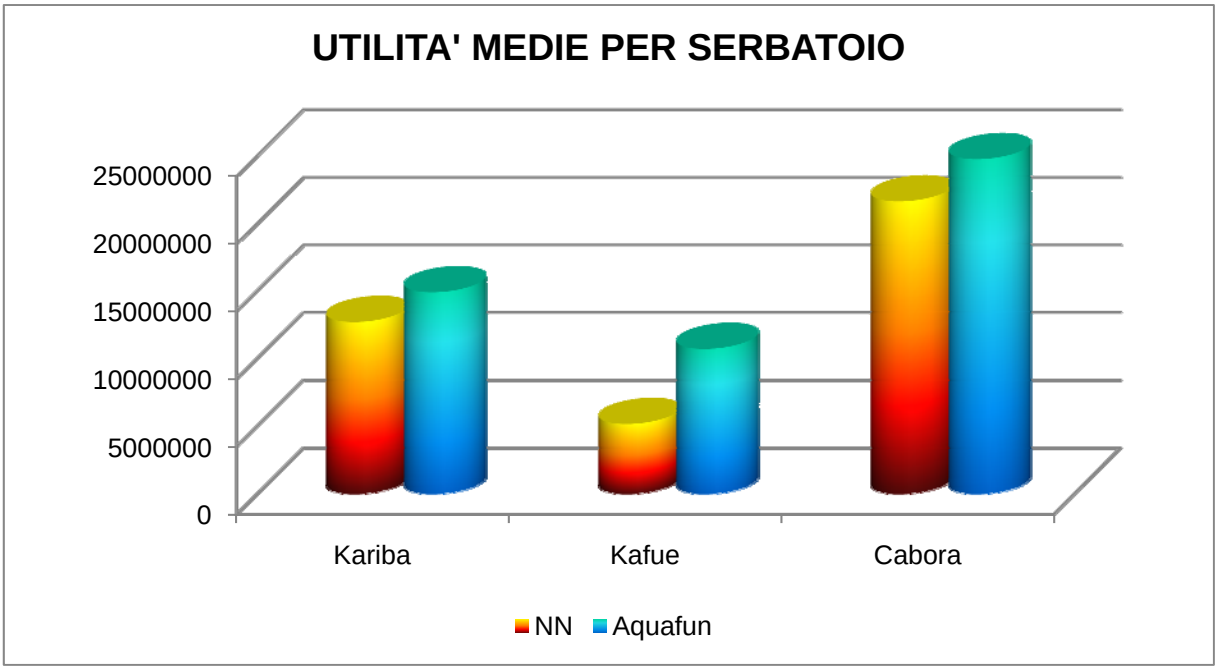

Figura 6.19 – Utilità totali generate dai rilasci nei tre differenti serbatoi, con i due modelli utilizzati

Il seguente Grafico 6.13 riporta le differenze percentuali di utilità tra Aquafun e la rete neurale, suddivise per mese e per serbatoio.

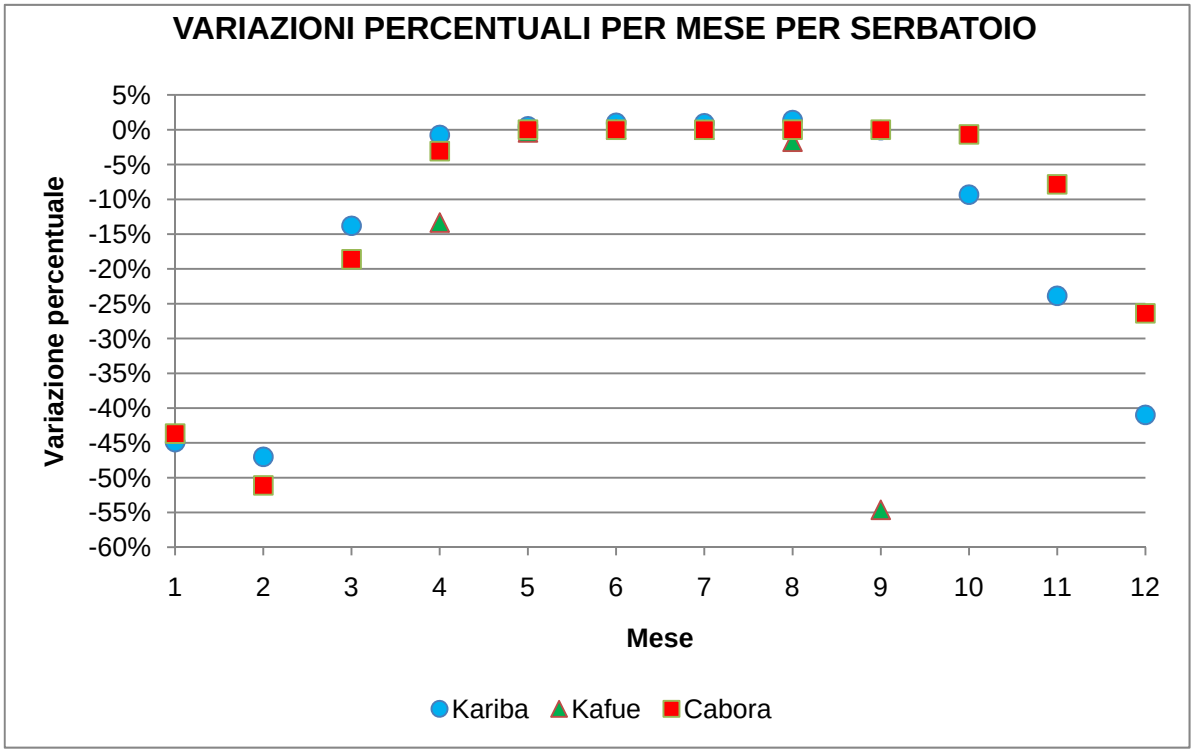

Grafico 6.13 – Variazioni percentuali nel calcolo delle utilità tra il modello lineare e quello neurale

Utilizzando questo modello neurale, quindi, la stima delle utilità appare decisamente migliore che nelle sperimentazioni passate, se ci si limita ai due serbatoi di Kariba e Cabora, dal

momento che nei mesi più secchi le perdite sono più contenute (anche per i singoli serbatoi), mentre sono pressoché nulle nei mesi più umidi.
Il serbatoio di Kafue, invece, ha un comportamento molto lontano da quello fornito dal modello lineare: la causa di questo potrebbe dipendere dal fatto che uno dei due rilasci non è tempo variante, e questo potrebbe inficiare la qualità dell'addestramento della rete neurale, che non riesce a capire come comportarsi con quel determinato serbatoio.
È interessante capire come il modello produce questi risultati, analizzando l'andamento degli invasi e dei rilasci in ognuno dei serbatoi che compongono il sistema (Grafici 6.14 – 6.17 di seguito).

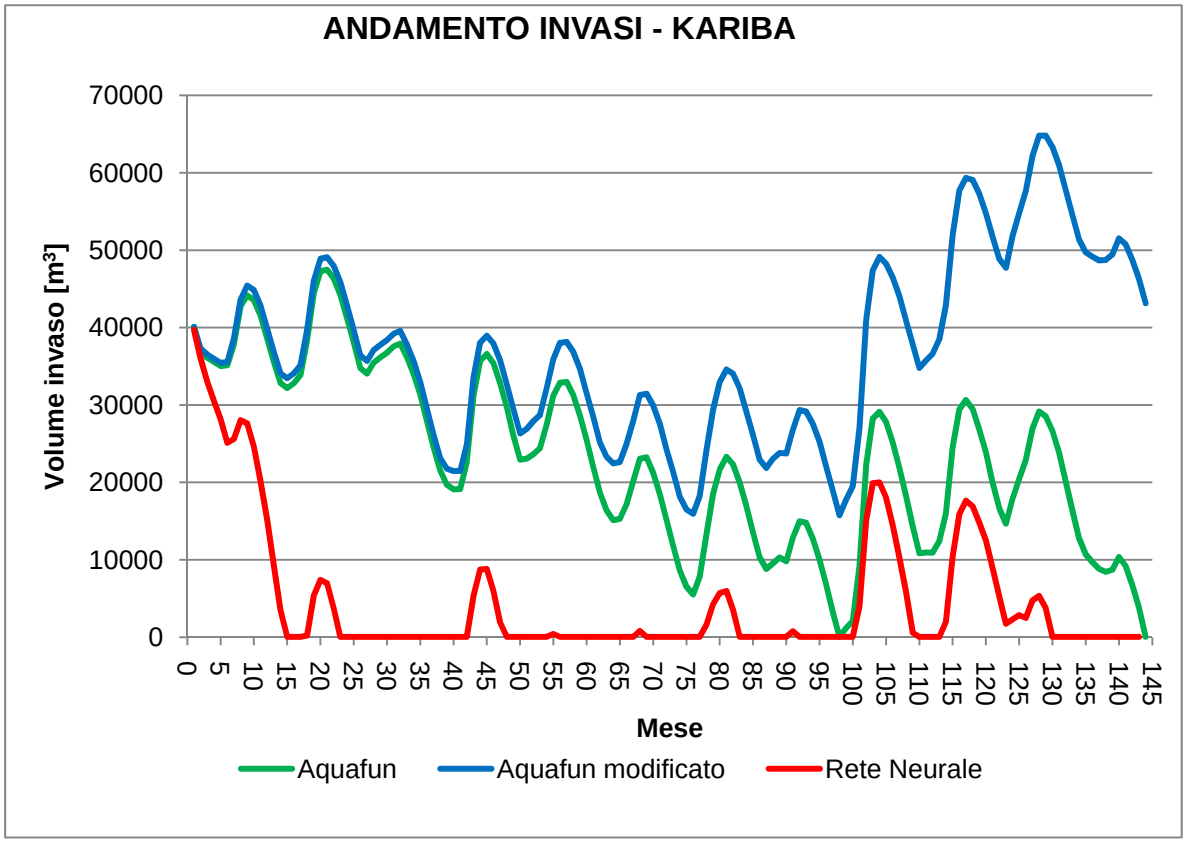

Grafico 6.14 – Andamento del volume dell'invaso di Kariba nelle tre diverse simulazioni

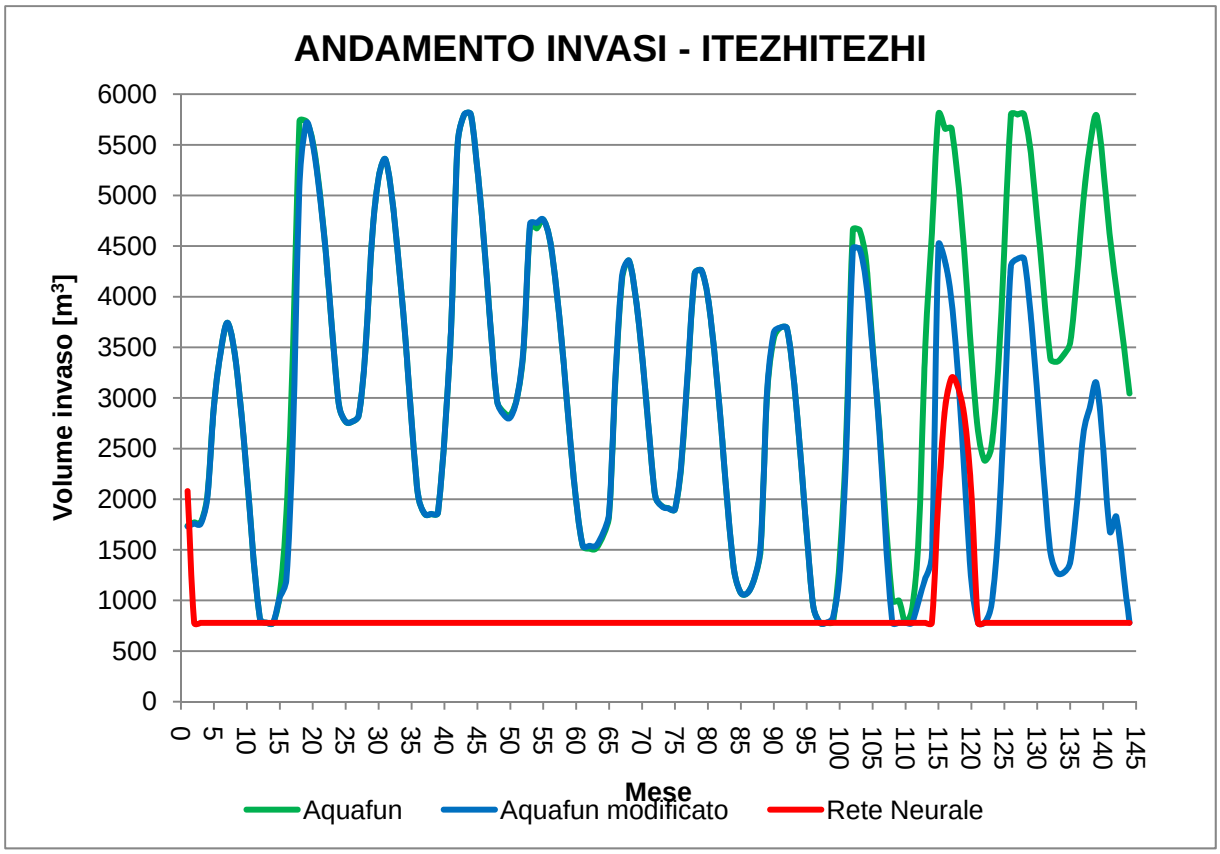

Grafico 6.15 – Andamento del volume di invaso per il serbatoio di Itezhi

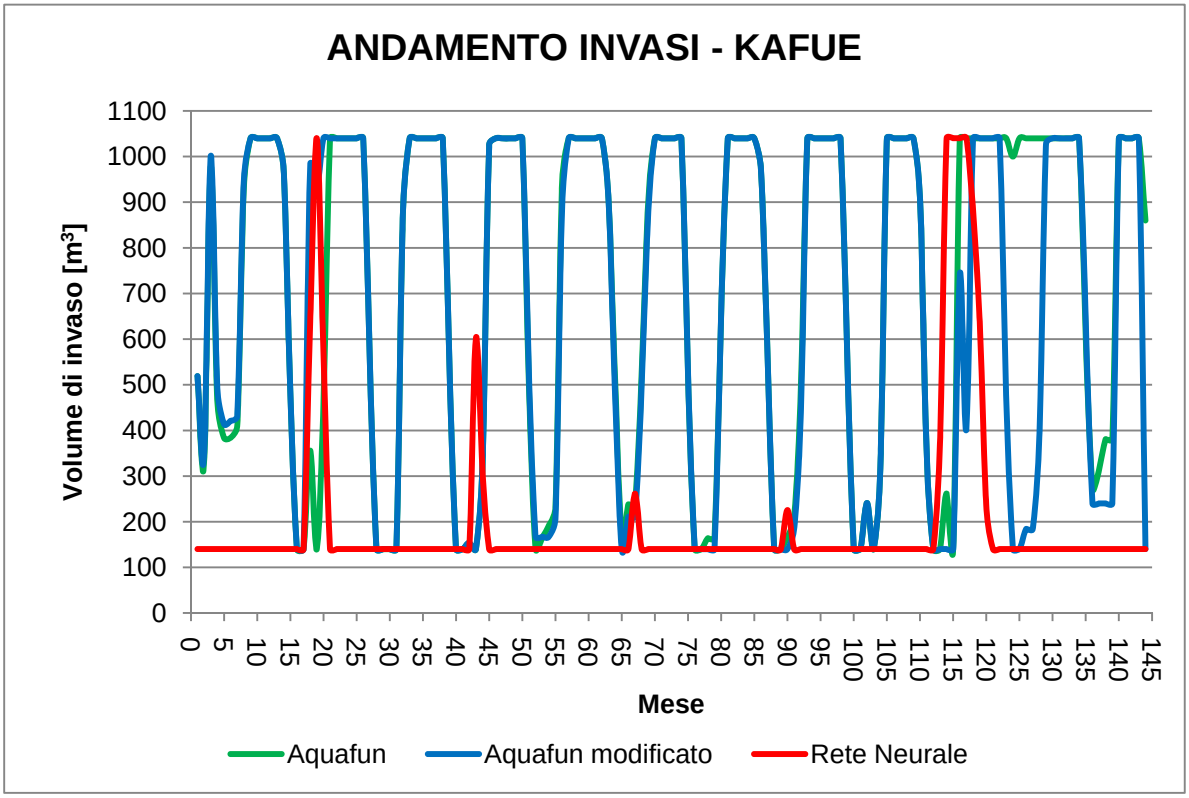

Grafico 6.16 – Andamento del volume di invaso per il serbatoio di Kafue

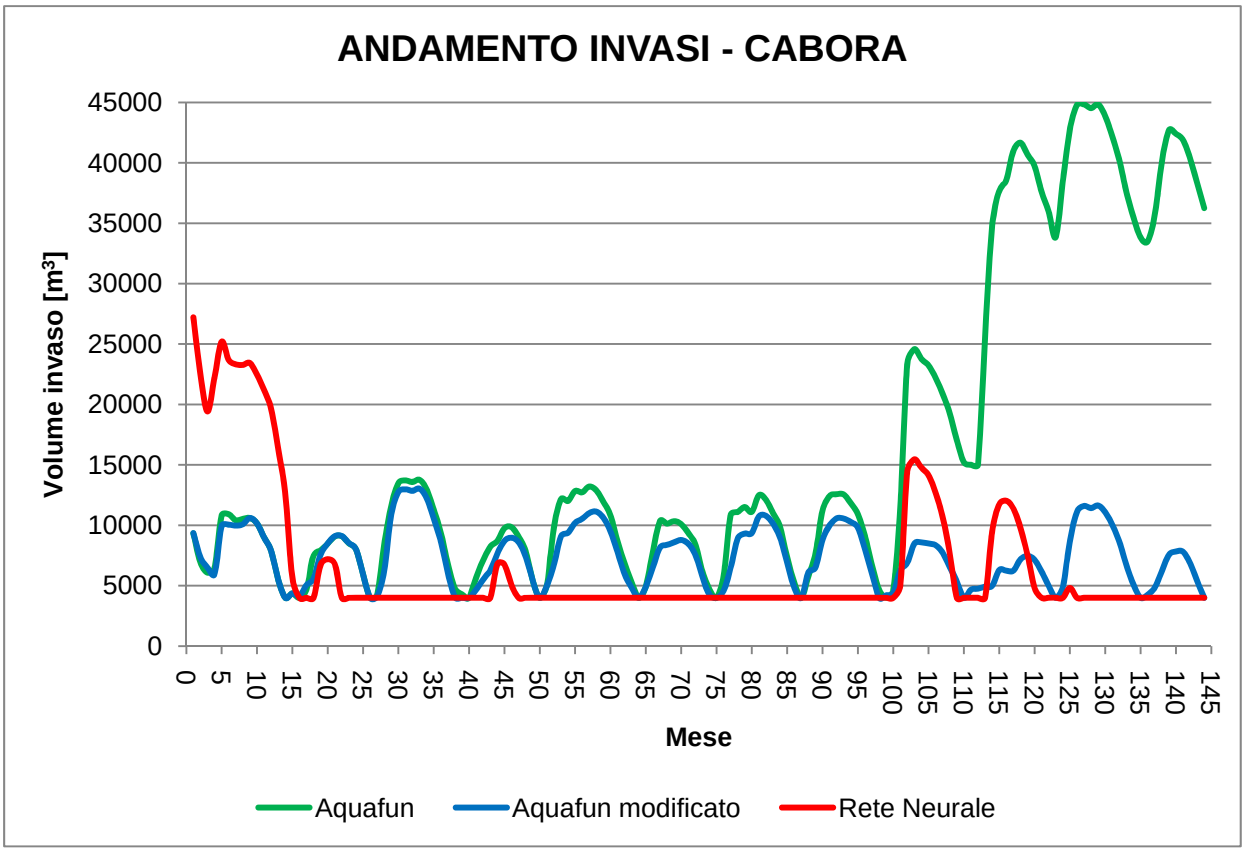

Grafico 6.17 – Andamento del volume di invaso per il serbatoio di Cabora

Sebbene questo modello neurale sia migliore dei precedenti per calcolare i rilasci dai diversi serbatoi ed ottenerne un beneficio, l'analisi degli andamenti dei volumi di invaso rivela che la politica trovata dal modello non è buona, perché i serbatoi non vengono regolati con continuità: esistono dei periodi in cui il volume si attesta ai valori minimi possibili, seguiti da rapidissimi incrementi, che raggiungono il massimo dell'invaso in pochi periodi. Inoltre, alla fine del periodo di simulazione, i volumi di invaso si attestano sui minimi, mettendo in luce come la risorsa idrica non venga mai conservata. Comunque, confrontando i precedenti Grafici (6.14 – 6.17) con gli analoghi (6.3 – 6.6), si osserva come in quest'ultima analisi gli invasi abbiano delle oscillazioni, che si rifanno, almeno grossolanamente, alle oscillazioni del modello lineare. Nel primo caso, invece, gli invasi vengono portati rapidamente alla condizione di massimo o minimo, ed ivi restano per il resto della simulazione.
La rete neurale che utilizza i valori medi mensili in addestramento è in grado, quindi, di capire le oscillazioni cui sono soggetti i serbatoi, e di replicarle meglio.

6.8.2 RETI A SINGOLO RILASCIO

I miglioramenti ottenuti addestrando reti neurali con dati medi giornalieri potrebbero essere ancora maggiori se si costruiscono modelli neurali che forniscono in uscita i controlli per un solo serbatoio. Ad esempio, in questo modo, il modello potrebbe apprendere meglio il funzionamento del serbatoio di Kafue, riuscendo a capire la natura dei suoi rilasci non tempo – varianti.
La procedura seguita è quella usuale: per ognuno dei cinque controlli sono state addestrate reti neurali con un numero di neuroni variabile tra *5* e *30* con passo cinque. Il simulatore della dinamica sarà costituito da reti neurali diverse: per ogni rilascio si è scelta quella in cui l'addestramento ha dato i risultati migliori, come riportato nella seguente Tabella 6.10.

# NEURONI		RILASCIO				
		KARIBA	ITEZHI	KAFUE 1	KAFUE 2	CABORA
5	MSE	0,78724	0,90703	0,84872	0,86071	0,75287
	R	0,45389	0,30145	0,38825	0,27605	0,50730
10	MSE	0,75610	0,89527	0,98321	0,80830	0,74071
	R	0,46072	0,32743	0,11967	0,30680	0,50679
15	MSE	0,80687	0,90797	0,83200	0,93519	0,78531
	R	0,46191	0,33265	0,40060	0,31878	0,51064
20	MSE	0,78574	0,91364	0,83291	0,81897	0,71690
	R	0,46173	0,34241	0,41179	0,31951	0,51147
25	MSE	0,71567	0,88891	0,85205	0,90766	0,75410
	R	0,46682	0,33034	0,39380	0,32779	0,45451
30	MSE	0,92072	0,86668	0,83198	0,85906	0,73270
	R	0,46233	0,34715	0,41828	0,33928	0,50902

Tabella 6.10 – Risultati dell'addestramento delle differenti architetture neurali. In verde sono evidenziati i migliori per ognuno dei controlli

Ancora una volta, l'addestramento non è ottimale per nessuna delle architetture neurali sperimentate, perché i valori di *MSE* sono elevati, mentre i valori di *R* si mantengono piuttosto bassi. Nella precedente Tabella 6.10 sono stati evidenziati, comunque, i migliori risultati ottenuti per ciascun controllo, le cui corrispondenti reti neurali saranno utilizzate in simulazione.

AQUAFUN		RETE NEURALE	VARIAZIONE [%]
1647852	RILASCI TOTALI [m^3/(s*mese)]	1627971	-1,21%
604474343	UTILITÀ TOTALE	285368266	-52,79%

Tabella 6.11.a – Risultati delle simulazioni con il modello neurale e con quello lineare

Nella precedente Tabella 6.11.a si riportano i risultati delle simulazioni lanciate sia con Aquafun sia con il modello neurale. Rispetto ai risultati ottenuti con un'unica architettura neurale, utilizzando i dati giornalieri in addestramento con una rete neurale per ognuno dei controlli, non si ottengono dei risultati soddisfacenti, come mostrano anche la seguente Tabella 6.11.b (risultati medi annuali) e seguenti Figure 6.20 – 6.21 ed il Grafico 6.18.

ANNO	AQUAFUN	RETE NEURALE	DIFFERENZE [%]
II	2285524,58	4189650,08	-45,45%
III	2094792,25	4190219,00	-50,01%
IV	1968244,33	4193942,25	-53,07%
V	2183975,50	4198389,92	-47,98%
VI	1806861,33	4197800,58	-56,96%
VII	2081452,42	4197800,58	-50,42%
VIII	1885681,92	4197800,58	-55,08%
IX	2243234,50	4203225,83	-46,63%
X	1916723,58	4202803,42	-54,39%
XI	1437557,92	4205402,17	-65,82%
TOTALE	*1990404,83*	*4197703,44*	*-52,58%*

Tabella 6.11.b – Differenze nel calcolo delle utilità medie annuali tra il modello neurale e quello lineare

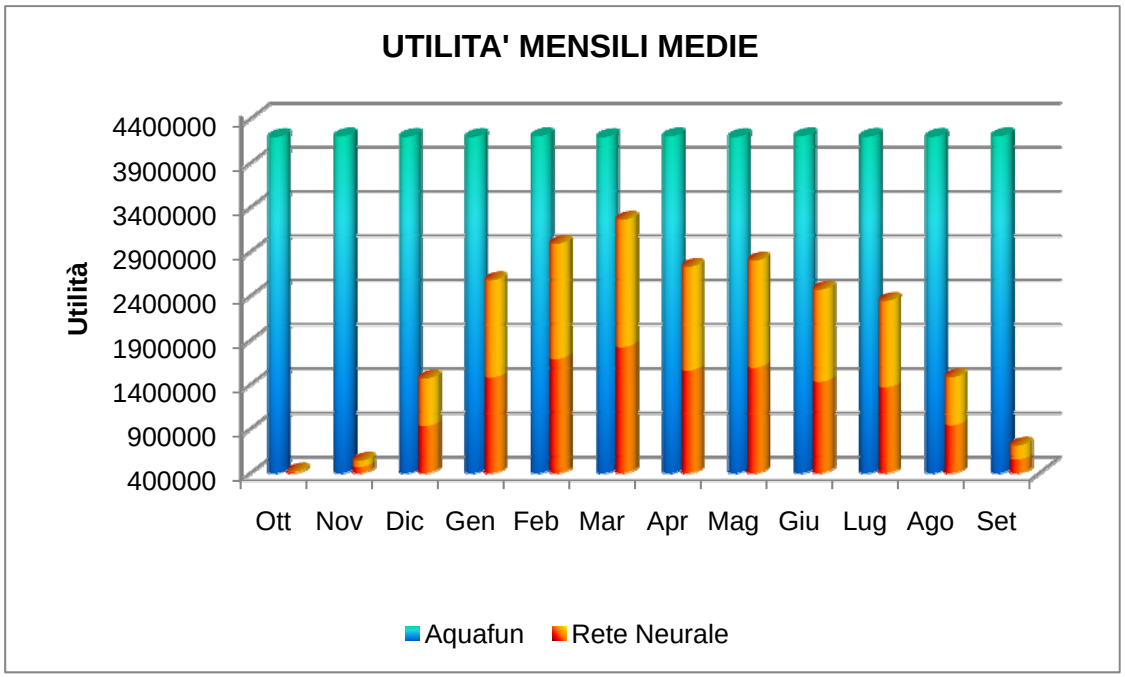

Figura 6.20 – Differenze delle utilità mensili medie calcolate dal modello lineare e da quello neurale

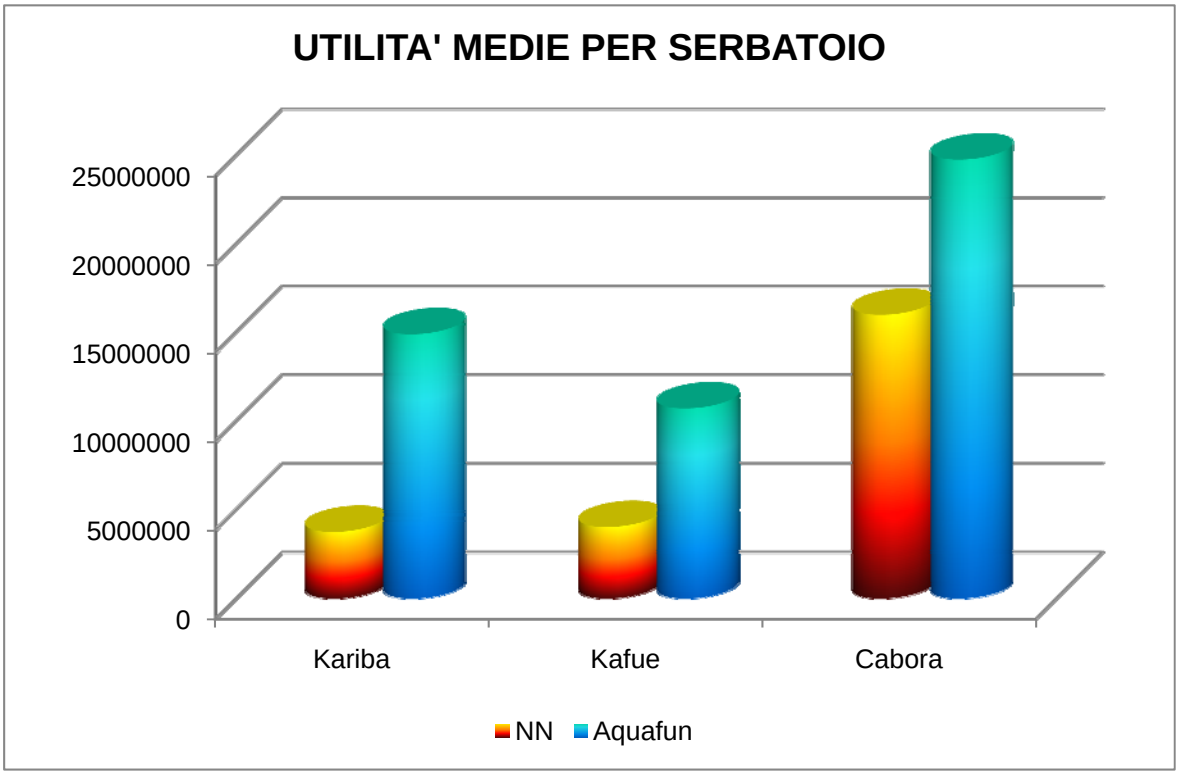

Figura 6.21 – Utilità medie per ogni serbatoio, calcolate con i due modelli

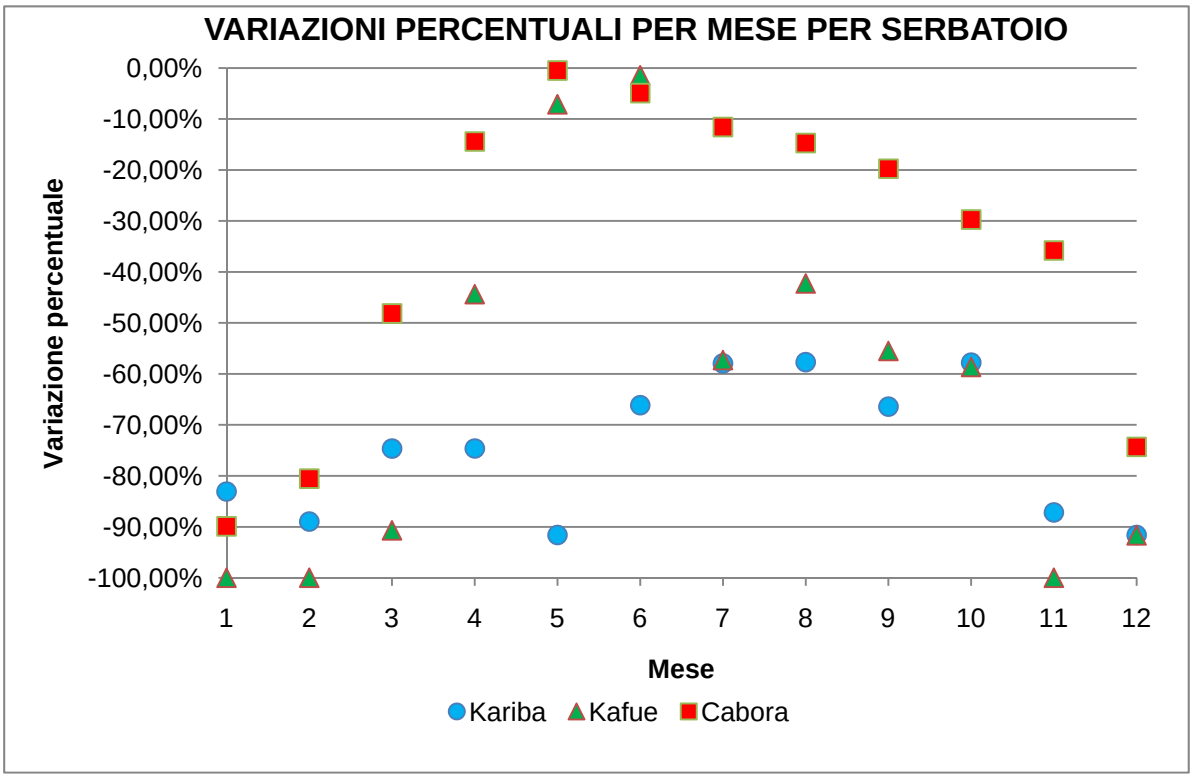

Grafico 6.18 – Differenze percentuali di utilità medie mensili calcolate per ognuno dei serbatoi con i due differenti modelli

Rispetto alle analisi precedenti, in questo caso la rete neurale ha prestazioni peggiori in ogni caso: sia nei mesi secchi, sia in quelli umidi, in cui l'utilità calcolata è sempre molto inferiore di quella elaborata da Aquafun. Anche l'analisi dei comportamenti dei singoli serbatoi, sia complessivamente sia per mese, mostra come questo modello sia meno performante di quello che utilizza un'unica rete neurale per tutti i controlli; osservando il Grafico 6.18, si nota come anche nei mesi più umidi, normalmente simulati meglio dal modello neurale, le differenze di utilità siano piuttosto pronunciate, anche per il serbatoio di Kariba.
A suffragio di questo, nel seguente Grafico 6.19 si riporta l'andamento dell'invaso proprio del serbatoio di Kariba, il cui andamento oscillatorio era abbastanza ben approssimato dal modello neurale (si veda il precedente Grafico 6.14).

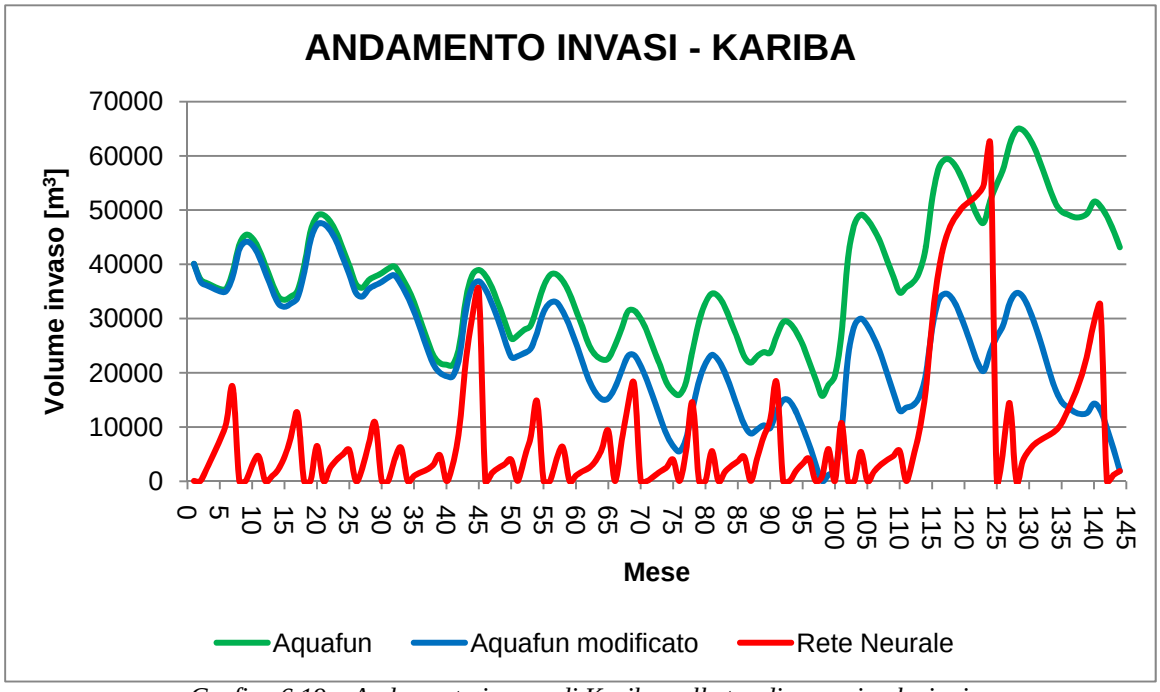

Grafico 6.19 – Andamento invaso di Kariba nelle tre diverse simulazioni

Rispetto ai modelli precedenti, la rete neurale risponde meglio alle oscillazioni del serbatoio, ma, complessivamente, la politica generata è scadente.

6.8.3 PERFEZIONAMENTO DELL'ADDESTRAMENTO

I rilasci dal serbatoio di Kariba sono sempre molto elevati, indipendentemente dalle condizioni di afflusso, mentre quelli dal serbatoio di Kafue verso il pozzo sono indipendenti dal tempo. Se, in fase di addestramento si utilizzano tutti i dati di invaso con valori medi giornalieri, si possono ottenere dei risultati inesatti.
È meglio, allora, escludere questi dati dalla procedura di addestramento ed utilizzare solamente quelli relativi al serbatoio di Itezhitezhi, Kafue verso Chirundu e Cabora.
Analogamente al caso precedente, sono stati testati modelli neurali a *6*, *8*, *10* e *12* neuroni; in simulazione, per quanto riguarda i controlli su cui non sono state addestrate le reti neurali, si sono utilizzati quelli elaborati da Aquafun, lanciato con le condizioni di chiusura di default. I risultati delle simulazioni sono riportati in Tabella 6.13 (come già capitato in precedenza, per alcune architetture neurali, la simulazione porta a condizioni di chiusura non gestibili da Aquafun).
L'addestramento delle reti, anche in questo caso, conduce a risultati cattivi, con, al solito, coefficienti di correlazione molto bassi e errori medi quadratici piuttosto elevati, riportati nella seguente Tabella 6.12.

# NEURONI	MSE	R [%]	R_tr [%]	R_val [%]
6	0,89289	0,32597	0,32757	0,32640
8	0,84297	0,33810	0,33662	0,34541
10	0,87932	0,34109	0,34345	0,33103
12	0,82762	0,38327	0,38031	0,38837

Tabella 6.12 – Risultati dell'addestramento

		RILASCI TOTALI [m3/(s*mese)]	UTILITA' TOTALI	VARIAZIONI [%]	
NN6	AQUAFUN	1551096	602638489	-2,00%	-18,98%
	RETE NEURALE	1520004	488241413		
NN8	AQUAFUN	1586466	602878409	-2,01%	-58,25%
	RETE NEURALE	1554598	251699092		
NN10	AQUAFUN	0	0	*non valutabile*	*non valutabile*
	RETE NEURALE	1502278	500006191		
NN12	AQUAFUN	0	0	*non valutabile*	*non valutabile*
	RETE NEURALE	1502278	500006191		

RILASCI TOTALI	UTILITA'

Tabella 6.13.a – Risultati delle simulazioni, in termini di utilità e rilasci

Già con una rete neurale a sei neuroni si ottengono, in questo caso, dei risultati molto buoni: la seguente Figura 6.22 mostra i risultati in termini di utilità totale, mentre la Tabella 6.13.b evidenzia le utilità medie annuali calcolate con i due differenti modelli, mettendone in luce le differenze.

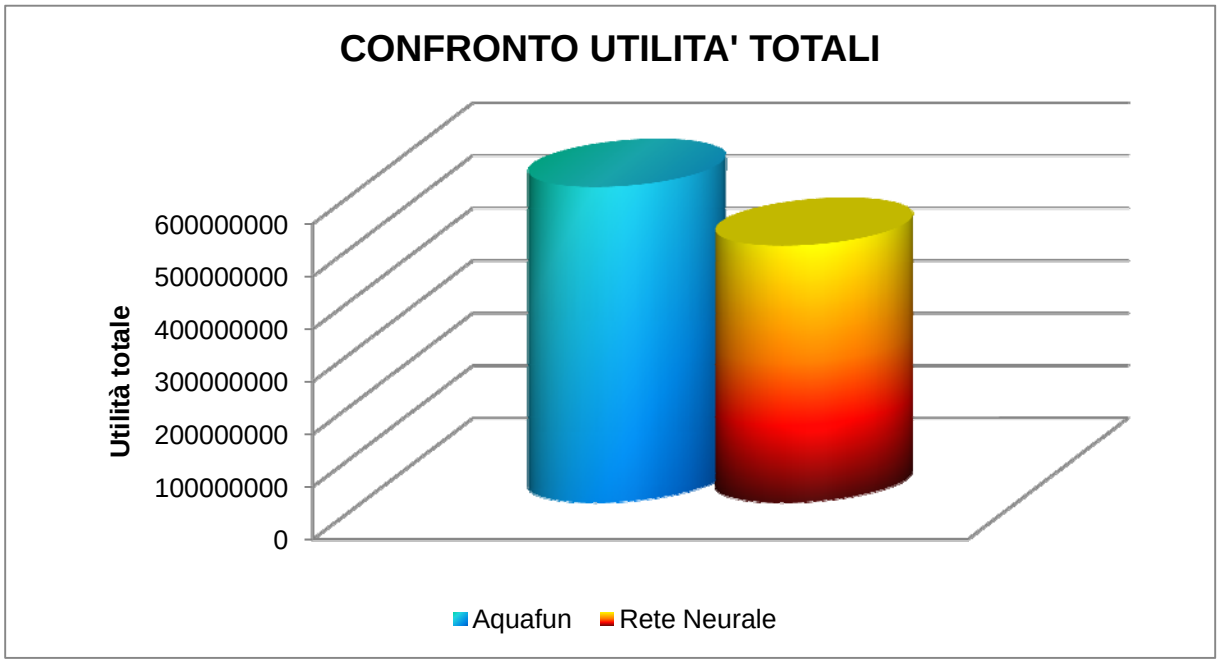

Figura 6.22 – Utilità totale calcolata dai due modelli

ANNO	AQUAFUN	RETE NEURALE	DIFFERENZE [%]
II	3419700,17	4189650,08	-18,38%
III	3540638,92	4190219,00	-15,50%
IV	3481117,33	4190051,25	-16,92%
V	3415355,92	4187146,42	-18,43%
VI	3408618,83	4188122,83	-18,61%
VII	3414581,25	4189093,75	-18,49%
VIII	3401096,25	4189093,75	-18,81%
IX	3482520,17	4188731,17	-16,86%
X	3627958,17	4190219,00	-13,42%
XI	3475245,08	4180675,08	-16,87%
TOTALE	*3466683,21*	*4188300,23*	*-17,23%*

Tabella 6.13.b – Differenze di utilità medie annuali calcolate con il modello neurale e con quello lineare

Anche l'analisi mensile delle utilità (Figura 6.23, di seguito) mostra che le perdite nei mesi più secchi sono molto contenute (mai sopra al *30%*), e le utilità calcolate con il modello neurale per i mesi più umidi si avvicinano molto a quelle del modello lineare.

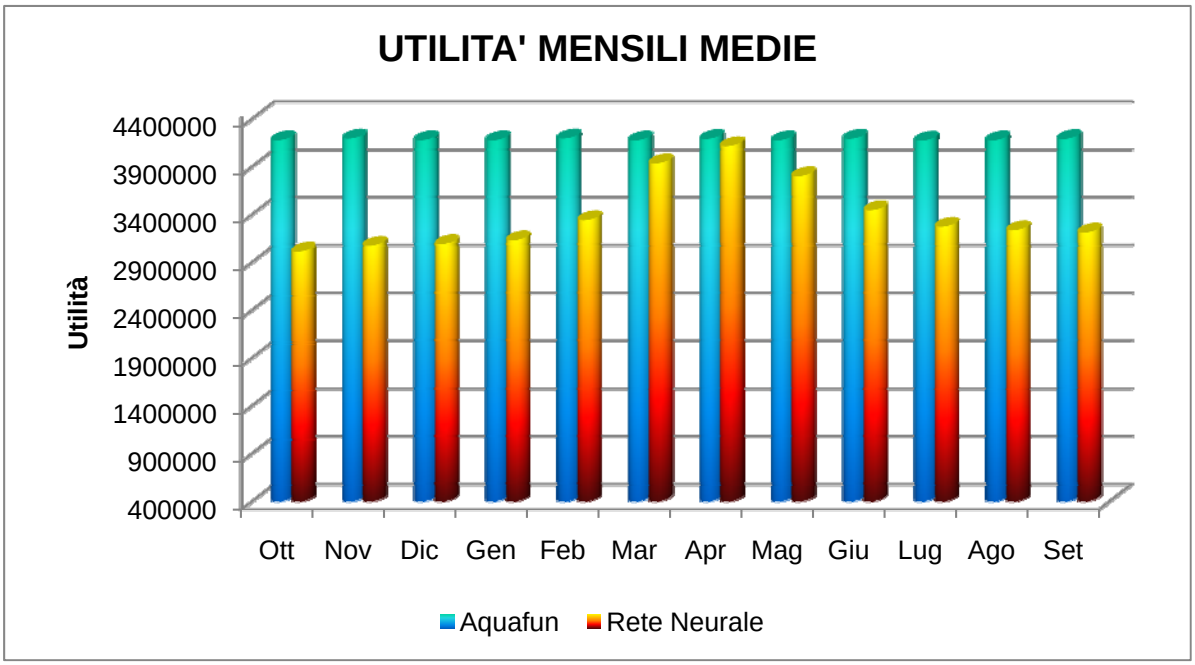

Figura 6.23 – Calcolo delle utilità medie mensili, ripartite per mese

Analizzando, al solito, i risultati per serbatoio, riportati nella seguente Figura 6.24 e nel Grafico 6.20, si evince che le stime per i serbatoi sono molto buone tranne per Kafue, che rimane di gran lunga il peggiore. Le perdite percentuali per i mesi più secchi non scendono mai al di sotto del *15%*.

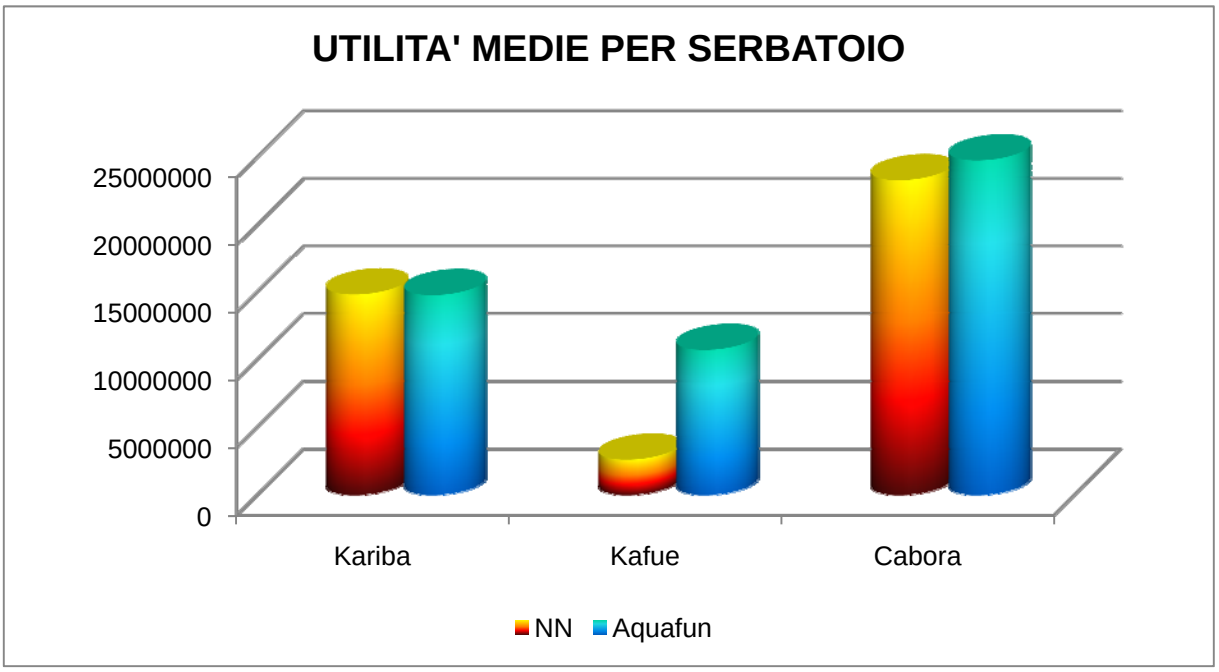

Figura 6.24 – Confronto tra le utilità medie calcolate ripartite per serbatoio

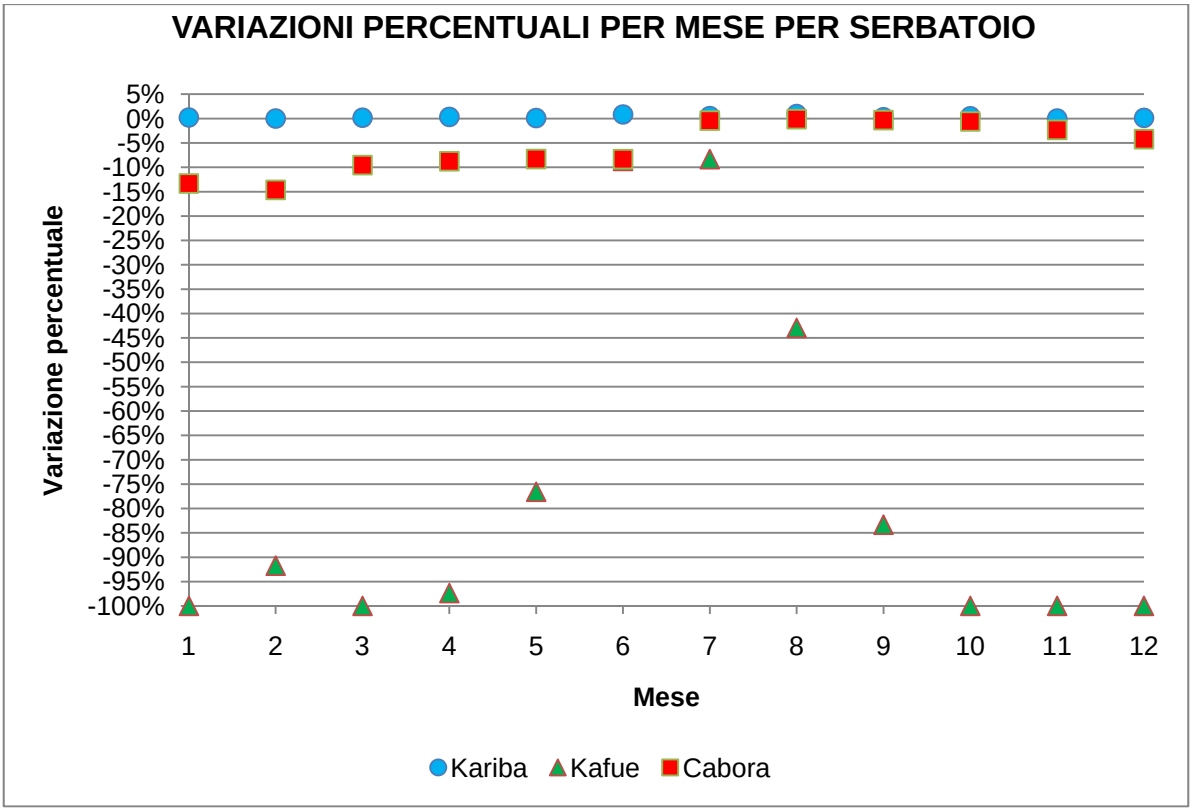

Grafico 6.20 – Perdite percentuali di utilità per serbatoio

Se, con questo modello, il comportamento dei serbatoi di Kariba e Cabora pare ben stimato in tutti i mesi, così non può dirsi per Kafue, il quale, anche nei mesi più umidi, genera un beneficio del 15% inferiore a quello ottenuto con il modello lineare.

Anche depurando, in fase di addestramento, dei valori dei controlli non tempo – varianti, non si riesce a produrre un risultato soddisfacente. Evidentemente, per questo serbatoio, non può essere utilizzato un simulatore neurale.

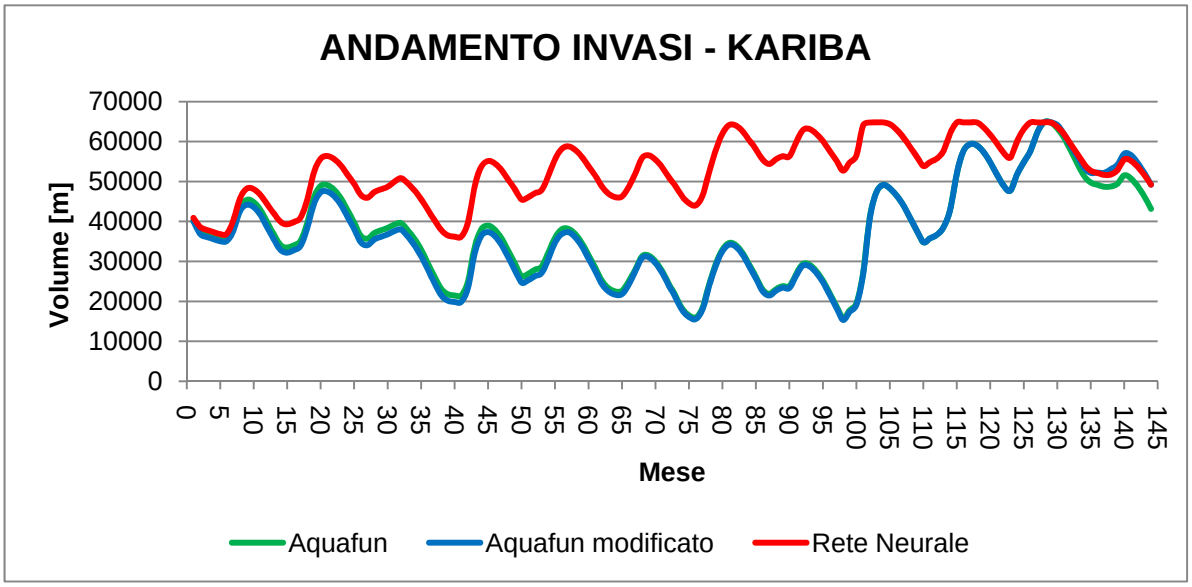

Grafico 6.21 – Andamento dell'invaso del serbatoio di Kariba

L'andamento mostrato nel precedente Grafico 6.21 è quello del volume di invaso del serbatoio di Kariba che, comunque, non è stato simulato con la rete neurale, giacché i rilasci da esso sono quelli di Aquafun.
Nel seguente Grafico 6.22 si riporta, invece, l'andamento dell'invaso di Cabora, simulato con la rete neurale.

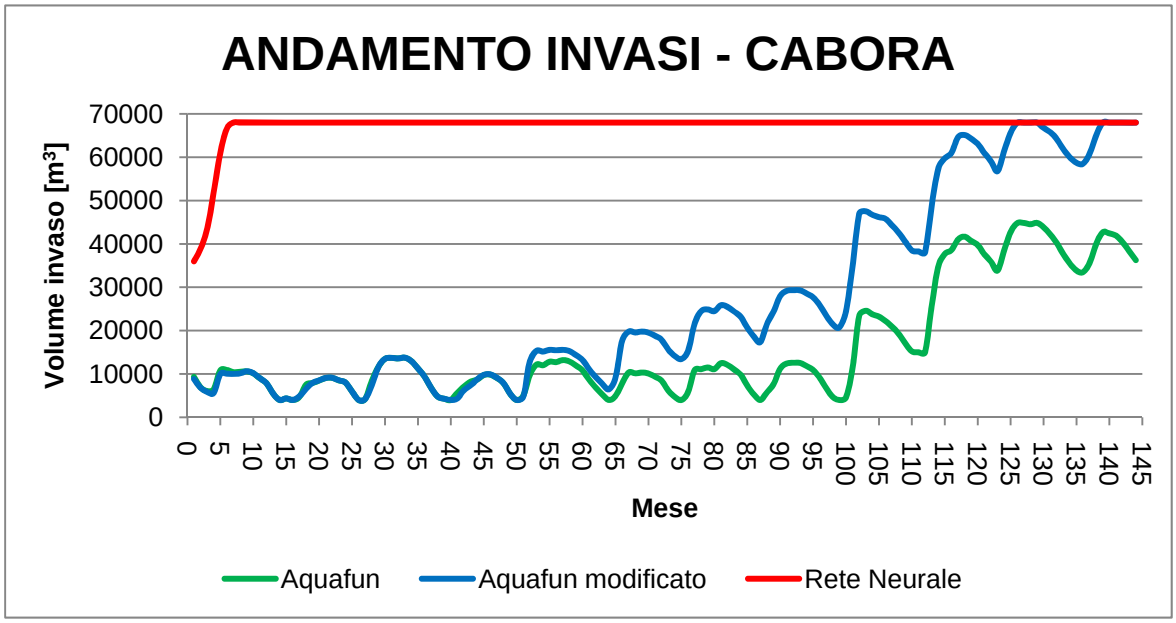

Grafico 6.22 – Andamento invaso di Cabora

Nonostante il modello neurale sia limitato alla simulazione dei serbatoi meno critici, sebbene la risposta in termine di beneficio totale conseguente sia soddisfacente, l'andamento dell'invaso di Cabora non può essere definito buono: ancora una volta, la rete neurale tende a mantenere la risorsa il più possibile nell'invaso, il cui volume viene portato ai livelli massimi in pochi periodi, e viene mantenuto costante.

7

CONCLUSIONI E SVILUPPI FUTURI

In questo capitolo conclusivo si discutono i risultati ottenuti, mettendo in luce aspetti positivi e negativi nell'utilizzo dei modelli neurali per risolvere un problema di allocazione ottima delle risorse; si tracciano le conclusioni e si forniscono le linee per i possibili sviluppi futuri del lavoro

I risultati ottenuti nel Capitolo 6 non paiono essere molto soddisfacenti, per cui i modelli neurali sembrano non rispondere bene all'esigenza di calcolare i rilasci di un sistema idrico che debba essere gestito per massimizzarne l'utilità prodotta dalla produzione elettrica.
Tuttavia, prima di rinunciare definitivamente ad applicare questi modelli, è bene effettuare un confronto con un'altra tipologia di politica, che non provvede in alcun modo né a massimizzare il beneficio, né a conservare la risorsa: una politica banale, da non ottimizzare.
Si supponga di gestire il sistema nel modo seguente:

- Per il serbatoio che non produce alcuna utilità (nel sistema dello Zambesi, quello di Itezhi), ad ogni istante temporale si rilascia sempre l'afflusso totale in ingresso;
- Per gli altri tre serbatoi si rilascia sempre la massima portata turbinabile, in modo da conseguire sempre l'utilità massima; quando i vincoli di capacità imposta sui serbatoi non consentono questo, si rilascia il massimo possibile.

Lo scopo è verificare se le simulazioni condotte con i modelli neurali generano una politica migliore almeno di questa, banale, che tende a mantenere la minore quantità possibile di acqua.
I risultati, espressi sempre in termini di utilità complessiva ottenuta, sono riportati nella Tabella 7.1, a pagina seguente.

MODELLO	UTILITA' MODELLO NEURALE	UTILITA' GESTIONE BANALE	DIFFERENZA [%]
Addestramento unico tutti i rilasci	417280118	441911977	-5,90%
Addestramento unico singoli rilasci	274326029		-61,09%
Addestramenti successivi	243706701		-81,32%
Contatore temporale addestramento completo	431102151		-2,51%
Valori medi giornalieri addestremento completo	474049071,1		6,78%
Valori medi giornalieri addestremento singolo	285368266		-54,86%
Valori medi giornalieri serbatoi non critici	488241413		9,49%

Tabella 7.1.a – Differenze di utilità complessive calcolate con i modelli neurali utilizzati nel Capitolo 6 e il simulatore a politica banale

MODELLO	RILASCIO NEURALE [m^3/mese]	RILASCIO GESTORE BANALE [m^3/mese]	DIFFERENZA [%]
Addestramento unico tutti i rilasci	1502468	1634832	-8,81%
Addestramento unico singoli rilasci	1566913		-4,33%
Addestramenti successivi	1620999		-0,85%
Contatore temporale addestramento completo	1566613		-4,35%
Valori medi giornalieri addestremento completo	1693621,108		3,47%
Valori medi giornalieri addestremento singolo	1627971		-0,42%
Valori medi giornalieri serbatoi non critici	1520004		-7,55%

Tabella 7.1.b – Differenze tra i rilasci totali elaborati dal modello a politica banale e le diverse reti neurali utilizzate

Come si evince dalle precedenti Tabelle, le reti neurali forniscono risultati migliori solamente in quei casi in cui, nelle analisi condotte nel precedente Capitolo 6, non causavano perdite eccessive rispetto ad Aquafun stesso, ossia utilizzando dati medi giornalieri per addestrare un'unica rete per tutto il sistema, e nel caso si limitasse l'addestramento ai soli serbatoi non critici. In tutti gli altri casi, invece, i modelli neurali rimangono peggiori, considerevolmente, anche rispetto al simulatore "a politica banale" qui introdotto. Tali modelli, comunque, non avrebbero potuto essere utilizzati per calcolare una politica effettivamente utilizzabile per il sistema idrico del fiume Zambesi, soprattutto per i rilasci che essi elaboravano: non è pensabile avere, per molti mesi, un rilascio nullo, e poi svuotare i serbatoi in pochissimi periodi, rilasciando enormi portate (anche al di sopra dei livelli di sfioro).
Ai rilasci è legata anche l'oscillazione dei volumi di invaso: se per molti mesi le portate uscenti sono nulle, è ovvio che il bacino tende a riempirsi. Per mantenere il rispetto dei vincoli, inoltre, in un certo momento occorre svuotare il serbatoio, rapidamente, generando delle oscillazioni non accettabili.

Se ci si limita al miglior modello neurale qui ottenuto, quello addestrato con dati medi giornalieri per i soli serbatoi non critici, si osserva che, mediamente, la rete neurale produce dei risultati migliori, soprattutto nei mesi più secchi, nei quali consente di ottenere un'utilità notevolmente superiore a quella calcolata con l'altro simulatore; nei mesi più umidi, pur se molto limitata, la rete neurale calcola dei rilasci che forniscono un'utilità più bassa (Figura 7.1 di seguito).

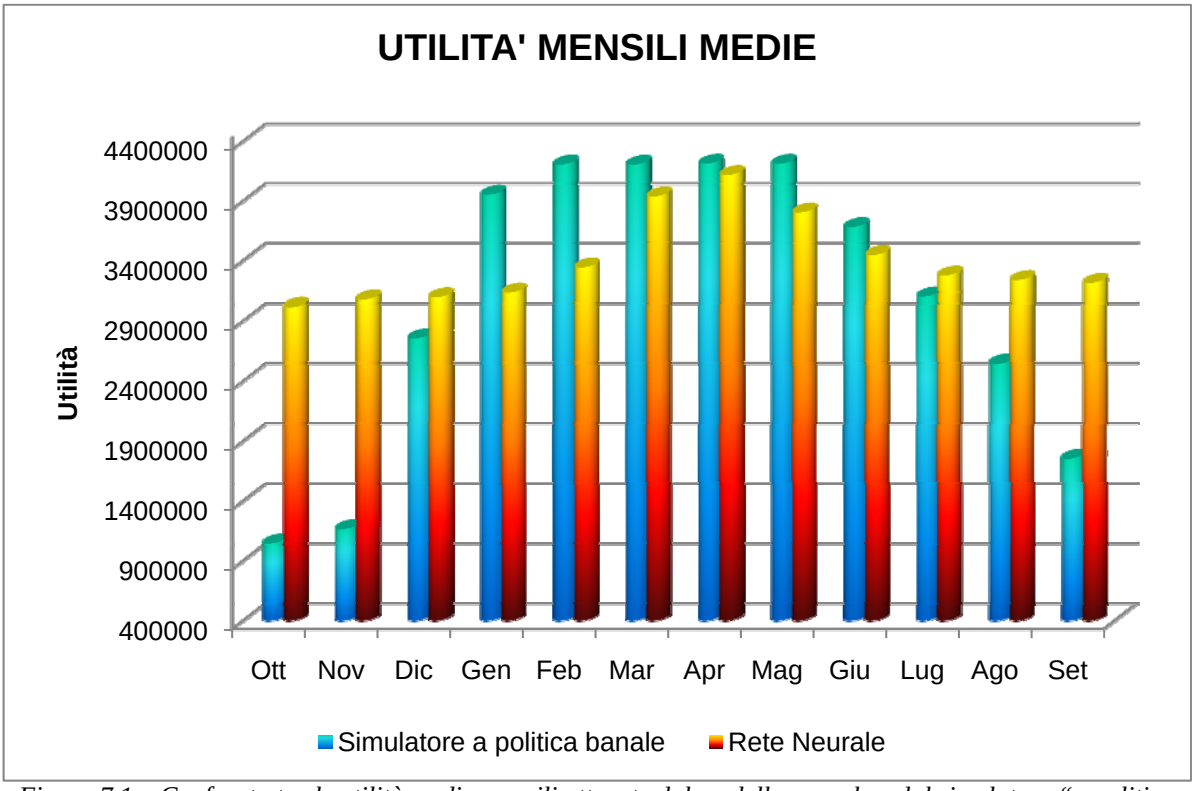

Figura 7.1 – Confronto tra le utilità medie mensili ottenute dal modello neurale e dal simulatore "a politica banale"

Anche l'analisi per serbatoio mostra dei risultati decisamente buoni, relativamente sempre al simulatore "a politica semplice": Kariba e Cabora si comportano, con il modello neurale, molto meglio, mentre Kafue è sempre il peggiore (evidentemente, nella dinamica di questo serbatoio esiste qualche criticità che non si è riuscito a mettere in evidenza). I risultati sono riportati nella seguente Figura 7.2 e nel Grafico 7.1.

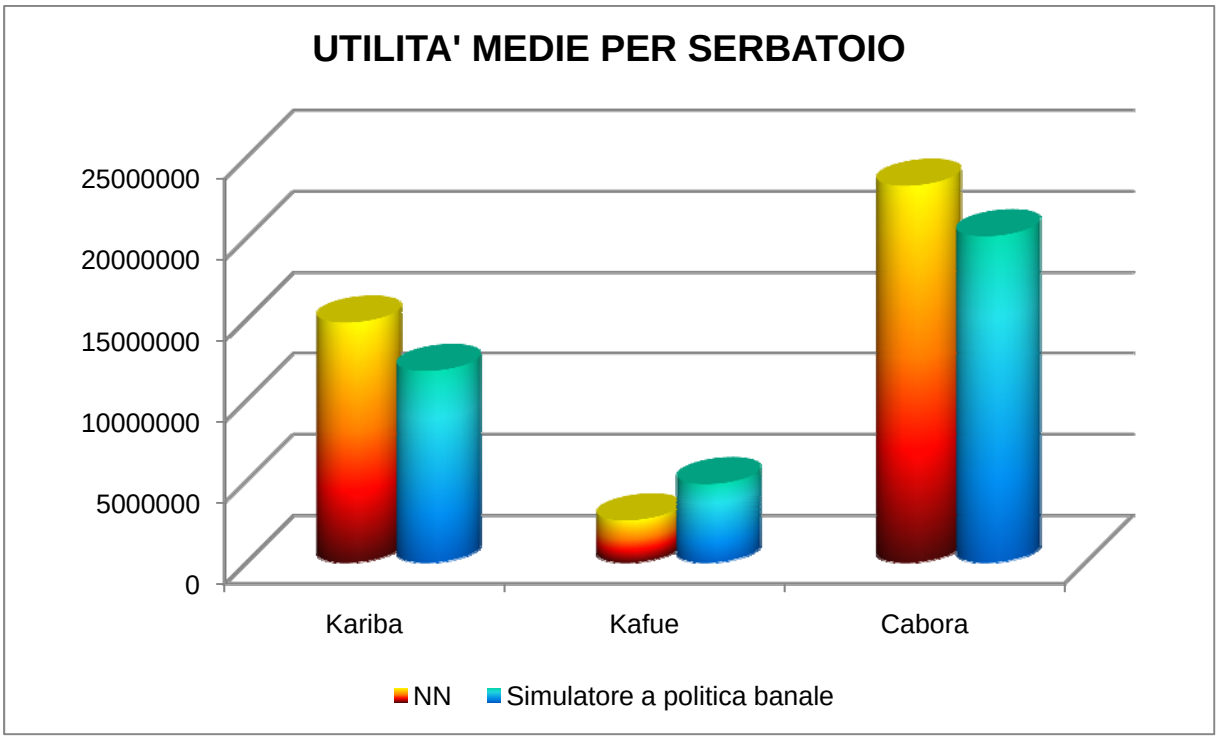

Figura 7.2 – Confronto tra le utilità medie totali per ognuno dei serbatoi produttivi

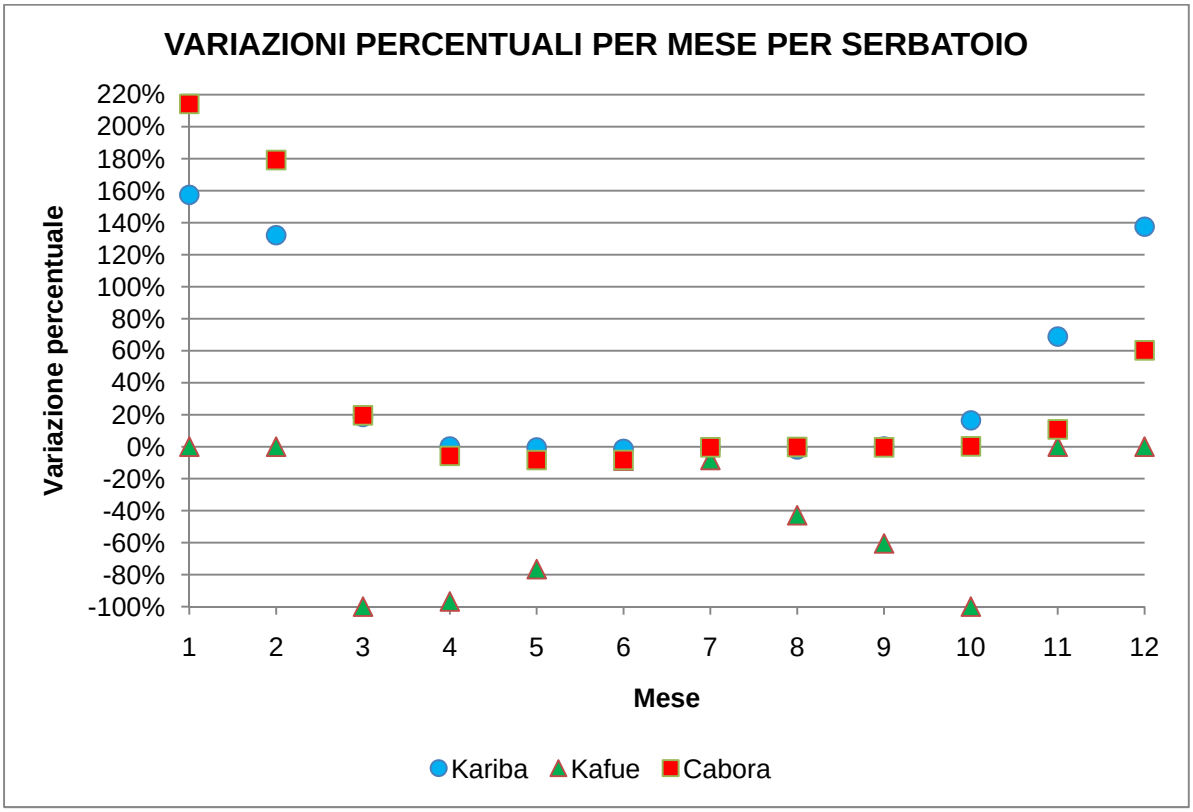

Grafico 7.1 – Differenza tra le utilità medie mensili per ognuno dei serbatoi produttivi calcolate con i due modelli

La scelta di utilizzare le reti neurali, pertanto, per calcolare una politica di gestione di un sistema idrico come quello del fiume Zambesi sembra, nel complesso, rivelarsi infelice.
Le motivazioni che hanno portato a questa conclusione sono da ricercarsi in diversi aspetti:

1) Le serie sintetiche create per addestrare le reti neurali non sono idonee allo scopo, o per via di un intrinseco errore, o per la forma modellistica adottata (che produce serie sintetiche i cui andamenti non rispettano, nella maggior parte dei casi, l'andamento della serie storica), pertanto le reti neurali sono male addestrate;
2) Il sistema del fiume Zambesi è strutturato in modo da essere molto sovradimensionato alle reali esigenze, per cui anche nelle situazioni particolari (anni particolarmente secchi o umidi) riesce a rispondere con un comportamento non molto dissimile da quello che mantiene nelle annualità "normali" dal punto di vista idrologico;
3) I modelli neurali sono inefficaci quando, indipendentemente dai dati in ingresso, devono calcolare uscite sempre costanti (è il caso del rilascio del serbatoio di Kafue verso il pozzo): in fase di addestramento la rete non riesce a percepire l'indipendenza degli output dagli input;
4) Utilizzare i dati, pur normalizzati, mensili che si hanno a disposizione, può essere una delle cause che determinano un cattivo addestramento, giacché la rete neurale non riesce a percepire le differenze nei valori dovute alle diverse durate dei mesi; utilizzare, infatti, i valori medi giornalieri aiuta a risolvere questo problema, e la qualità dell'addestramento migliora;
5) I serbatoi del sistema dello Zambesi hanno dimensioni e comportamenti molto dissimili tra loro, e anche questo può contribuire a peggiorare la qualità dell'addestramento; di contro, utilizzare una rete neurale per calcolare ognuno dei rilasci del sistema non migliora apprezzabilmente i risultati.

La conclusione più immediata cui si giunge è che, per questo specifico sistema idrico, le reti neurali non sono adatte calcolarne la dinamica.
Prima di abbandonare definitivamente la via intrapresa, ossia verificare se è possibile ricorrere a modelli alternativi alla Programmazione Dinamica per la risoluzione di un problema di allocazione ottima delle risorse, occorrerebbe applicare le reti neurali ad altri sistemi idrici, strutturati come quello dello Zambesi, ma più contenuti, in termini di dimensioni, e soggetti ad apporti meteorici con una variabilità assai più marcata.
Occorrerebbe poi procedere a creare nuove serie sintetiche, magari con modelli differenti, che costruiscano dati il cui andamento sia più simile a quello della serie storica a disposizione.
Probabilmente, sebbene di solito architetture neurali con un solo strato nascosto e pochi neuroni siano ampiamente soddisfacenti, può essere che per questo particolare tipo di problemi si debbano sperimentare nuove strutture, rinunciando ad utilizzare la struttura classica di questo tipo di modelli.

A1

FUNZIONAMENTO DEL SOFTWARE "AQUAFUN"

In questa Appendice si spiega come funziona il software di ottimizzazione lineare Aquafun, implementando una piccola rete idrica di esempio al suo interno.

A1.1 DEFINIZIONE DELLA RETE DI ESEMPIO

Al fine di rendere estremamente semplice la comprensione del funzionamento del software Aquafun, si è scelto di implementare una semplicissima rete idrica, costituita da un solo serbatoio (*res1*), e tre periodi temporali di funzionamento. La seguente Figura A1.1 riporta lo schema della rete appena definita.

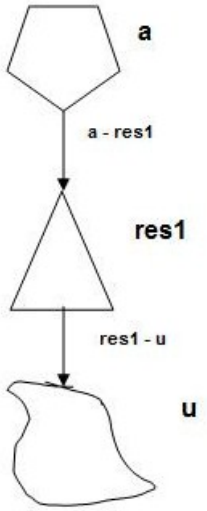

Figura A1.1 – Schema della rete di esempio

Come detto, tale rete si compone di un solo serbatoio chiamato *res1*, il quale riceve acqua da un bacino imbrifero denominato *a*; la decisione di rilascio si effettua unicamente allo scopo di fornire acqua al sottostante distretto irriguo, identificato con *u*. Completa la definizione della rete la presenza di due canali, ossia le vie (non necessariamente reali) che conducono l'acqua da un elemento ad un altro della rete: nel caso rappresentato, esistono sia un canale reale, *res1 – u*, che adduce acqua al distretto irriguo, dal serbatoio, sia un canale fittizio che invece conduce l'acqua proveniente dal bacino imbrifero al serbatoio (canale *a – res1*). Ogni periodo

(tipicamente un mese) è caratterizzato da afflussi diversi, che, nel caso modellizzato, sono pari a volumi di *10*, *20* e *15* metri cubi medi mensili ($\left[\frac{m^3 medi}{mese}\right]$) di acqua.
Il serbatoio deve essere definito mediante la capacità di invaso minima e massima (con valori che eventualmente possono variare in ogni periodo), fissati uguali per tutta la durata della simulazione, pari a *0* e *100*; occorre assegnare anche un valore iniziale ed uno finale al volume idrico contenuto nel serbatoio, fissato pari a *50* sia all'inizio sia alla chiusura della simulazione.
Deve, per ogni periodo, essere fornita l'evaporazione media mensile, fissata, in questo caso, uguale per tutta la durata, pari a *1*, e, per quanto riguarda i canali, devono essere definiti la minima portata circolante (assimilabile ad un Deflusso Minimo Vitale – DMV), l'eventuale numero di periodi di ritardo nel trasferimento del volume idrico da un nodo al successivo, e la funzione di utilità, che rappresenta il beneficio nel trasferire lungo il canale un certo volume idrico. Nel modello della rete qui presentato, si è scelto di porre nullo il ritardo di trasferimento per tutti i periodi, e di porre rispettivamente pari a *2*, *3* e *1* le portate minime circolanti nel canale che unisce il serbatoio al distretto irriguo, mentre si è ritenuta valida, per il primo canale, l'ipotesi che in esso non circoli alcuna portata, simulando, ad esempio, il caso in cui non vi siano afflussi. Le caratteristiche della rete idrica di esempio sono riportate nella seguente Tabella A1.1.

	I PERIODO	II PERIODO	III PERIODO
NUMERO PERIODI	3		
NUMERO SERBATOI	1		
NUMERO POZZI	2		
NUMERO CANALI	2		
NUMERO CICLI	1		
SEQUENZA AFFLUSSI	10	20	15
VOLUME MINIMO INVASO	0		
VOLUME MASSIMO INVASO	100		
VOLUME INIZIALE INVASO	50		
EVAPORAZIONE	1	1	1
RITARDO TRASPORTO (in periodi)	0	0	0
PORTATA MINIMA CIRCOLANTE *a - res1*	0	0	0
PORTATA MINIMA CIRCOLANTE *res1 - u*	2	3	1

Tabella A1.1 – Caratteristiche della rete di esempio

Per quanto riguarda, infine, le funzioni di utilità, esse possono essere definite in un solo modo, ossia a scalino, in modo che per flussi crescenti gli incrementi di beneficio siano decrescenti (come riportato nelle seguenti Tabelle A1.2). Non sono ammessi valori negativi.

PERIODO I			
Δ RIL.	Δ UTIL.	RILASCI	UTILITÀ
0	0	0	0
5	100	5	100
5	70	10	170
5	40	15	210
5	20	20	230
5	10	25	240
5	5	30	245
95	0	35	245

PERIODO II			
Δ RIL.	Δ UTIL.	RILASCI	UTILITÀ
0	0	0	0
10	100	10	100
10	70	20	170
7	50	27	220
4	20	31	240
4	0	35	240

PERIODO III			
Δ RIL.	Δ UTIL.	RILASCI	UTILITÀ
0	0	0	0
5	75	5	75
5	50	10	125
5	30	15	155
5	20	20	175
5	10	25	185
10	0	35	185

Tabelle A1.2 – Funzioni utilità, scalini e progressivi

Di seguito, per semplicità di comprensione, si riportano i grafici delle funzioni di utilità, sia in forma continua, sia con gli scalini che le discretizzano (Grafici A1.1.a – A1.3.b): questi ultimi sono stati ottenuti ricorrendo alla funzione "*indice*" implementata in Excel.

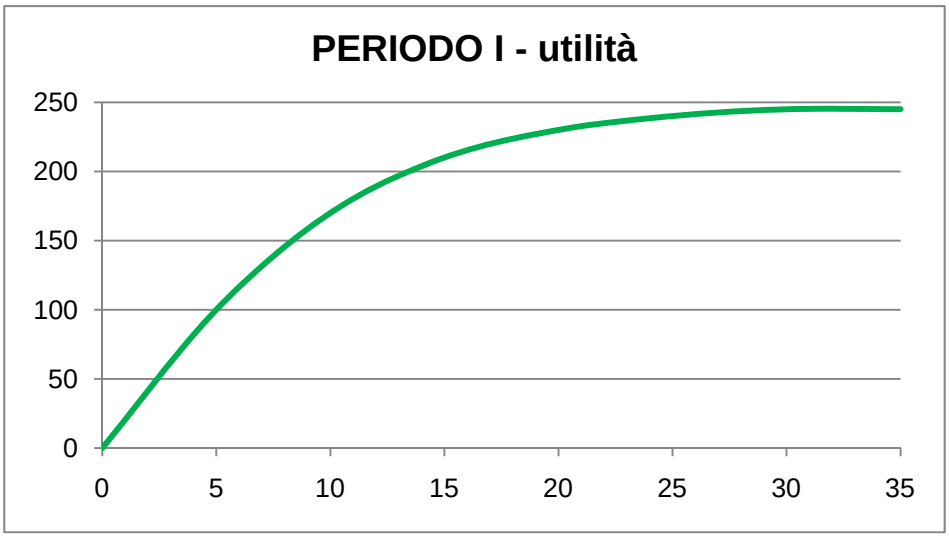

Grafico A1.1.a – Funzione di utilità relativa al periodo 1

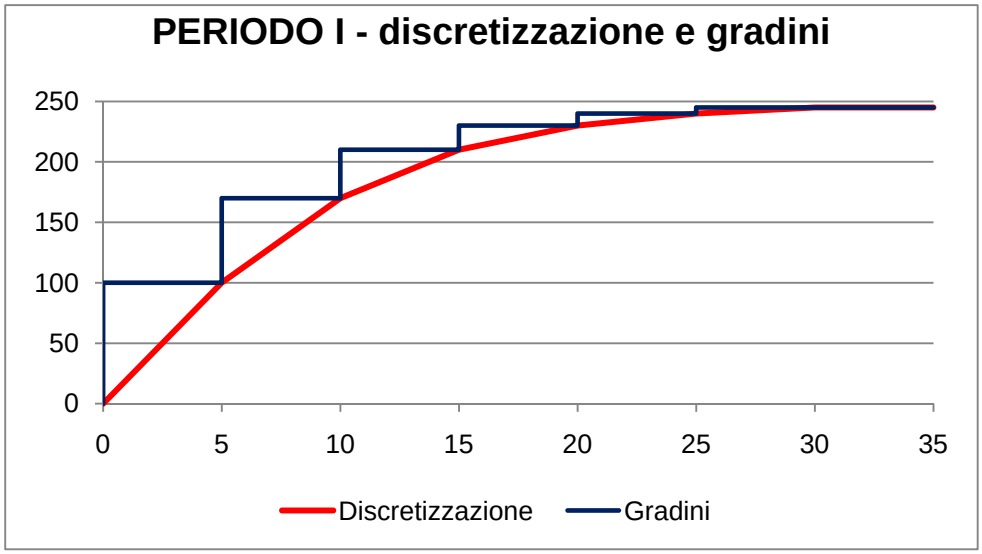

Grafico A1.1.b – Discretizzazione e gradini della funzione di utilità relativa al primo periodo

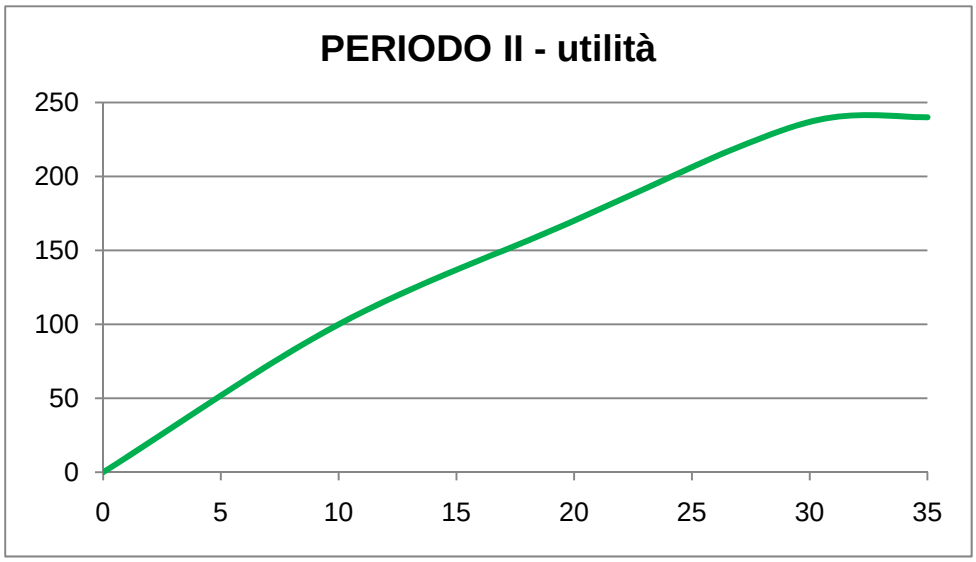

Grafico A1.2.a – Funzione di utilità relativa al periodo 2

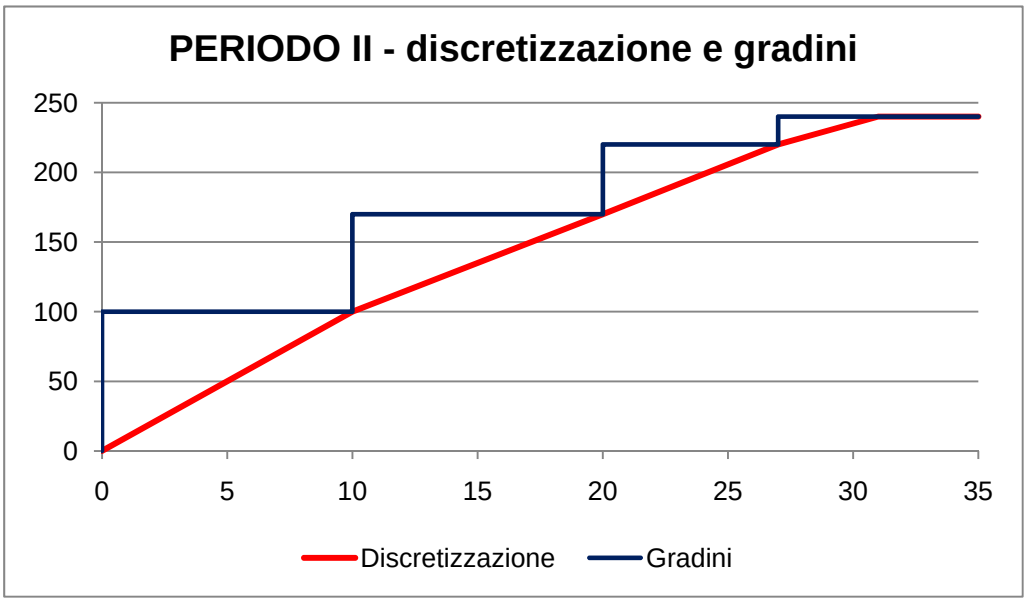

Grafico A1.2.b – Discretizzazione e gradini della funzione di utilità relativa al secondo periodo

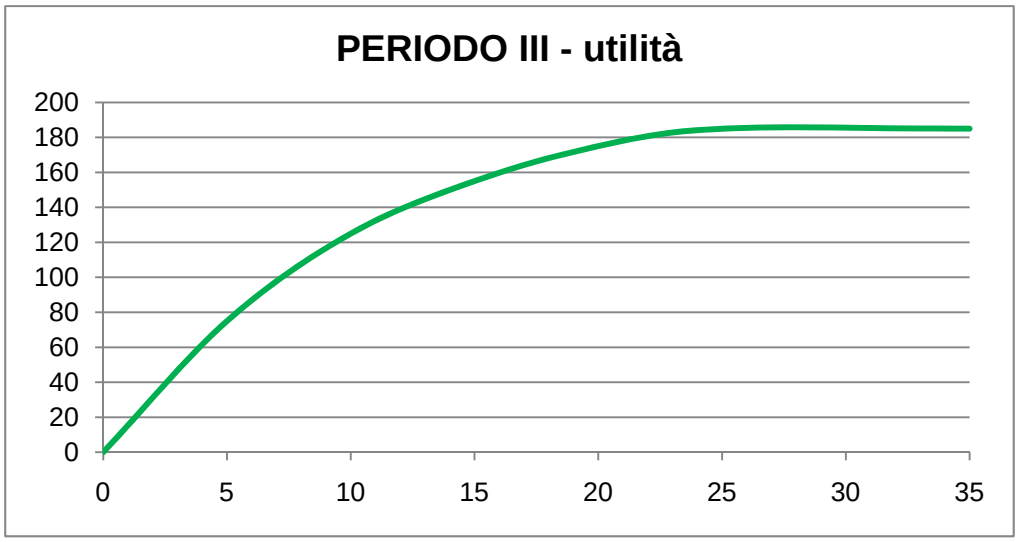

Grafico A1.3.a – Funzione di utilità relativa al Periodo 3

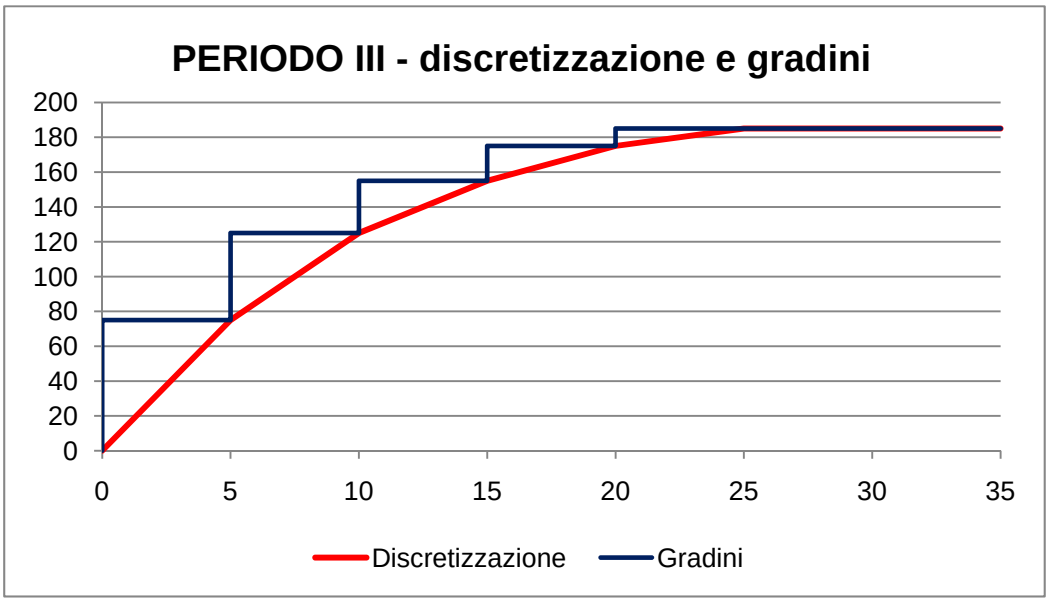

Grafico A1.3.b – Discretizzazione e gradini della funzione di utilità relativa al terzo periodo

È infine possibile definire dei *cicli*, ossia, utilizzando la definizione introdotta da Taddio e Togni nel loro lavoro di Tesi: *"… il numero di periodi dopo il quale tutti i dati riferiti alla struttura topologica della rete, funzioni di costo (utilità) comprese, si ripetono, mentre possono risultare differenti i flussi alla sorgente…"*.
Per il caso in esame si è adottato un solo ciclo, quindi la rete non si ripete più volte nel tempo.

A1.2 STRUTTURA E FUNZIONAMENTO DI AQUAFUN

Aquafun è un semplice software che si utilizza per l'ottimizzazione di una rete idrica. Questo applicativo è stato scritto (Taddio, Togni, 1993) mediante il linguaggio di programmazione *C* nei primi Anni *'90*, e funziona sotto *DOS*, motivo che può causare alcune instabilità, se il software viene caricato ed utilizzato sui più moderni calcolatori. Nell'utilizzo per questo lavoro di Tesi si sono verificati alcuni, banali, rallentamenti, proprio a causa della parziale incompatibilità tra il software e il computer utilizzato.
Nonostante questo, Aquafun si presenta con un'interfaccia piuttosto semplice da utilizzare, che consente all'utilizzatore diverse opzioni, come il caricamento di una rete già definita e strutturata, l'ottimizzazione di questa, oppure la creazione di una nuova rete idrica.
Il software viene fornito direttamente come eseguibile, in un pacchetto compresso: occorre estrarre tutto nella stessa directory, prestando attenzione che vengano create la cartella "NET" (dove Aquafun scrive i file di definizione della rete), e all'interno di questa, la cartella "SOL", dove vengono scritti i file di output.
La schermata di avvio di Aquafun è riportata nella seguente Figura A1.2.

Figura A1.2 – Schermata di avvio di Aquafun

Essa conduce, premendo un qualsiasi pulsante, al menu principale, rappresentato nella seguente Figura A1.3:

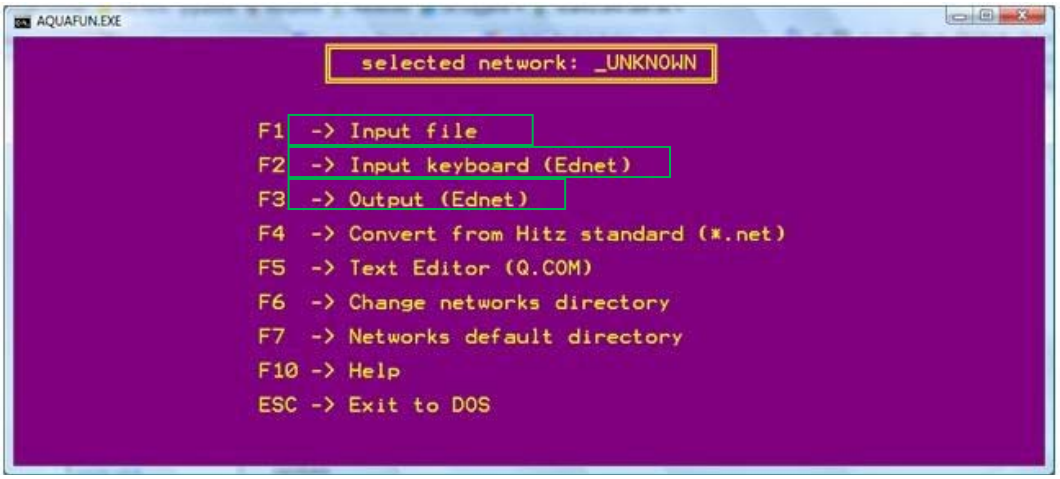

Figura A1.3 – Menu di Aquafun, con rielaborazione personale per evidenziare (in verde) le opzioni principali

Premendo i relativi tasti funzione, si accede alle schermate di interesse. Le opzioni di maggiore interesse sono:

- Il caricamento di una rete già definita in precedenza (“*Input file”*), mediante il tasto F1;
- La definizione di una nuova rete (“*Insert keyboard (Ednet)”*), mediante il tasto F2;
- L’analisi dei risultati ottenuti dall’ottimizzazione di una rete (“*Output (Ednet)”*), premendo il tasto F3;

Esistono poi altre opzioni, che possono essere definite “di servizio”, e consentono di convertire una rete creata mediante altri applicativi in un formato adatto ad essere interpretato da Aquafun (F4), di accedere ad un editor di testo implementato nel software (F5), denominato “Ednet”, di cambiare directory di accesso, di visualizzare una guida del programma, o di uscire da esso.

A1.2.1 INSERIMENTO DI UNA RETE

Per comprendere al meglio come deve essere definita una rete idrica in Aquafun, si prende ad esempio la semplicissima rete descritta nel precedente Paragrafo A1.1. Per attivare l’applicativo che consente di inserire manualmente una rete, dal menu principale del software (Figura A1.3) si deve attivare l’opzione “Insert Keyboard” (F2), mediante la quale si apre Ednet, il programma che guida l’utente nella definizione di una nuova rete. La procedura di definizione si realizza in quattro fasi, al termine di ciascuna delle quali viene scritto, nella directory dove è installato Aquafun all’interno della cartella “NET” (o in una differente, se specificato dall’utente), un file (visualizzabile con un comune editor di testo, ad esempio *Notepad*, utilizzato nel presente esempio) che riporta le informazioni inserite durante la procedura. È necessario premettere che Aquafun legge solamente numeri interi, che devono essere inseriti esclusivamente da tastiera confermando con il tasto “Invio”: gli eventuali errori possono essere corretti o modificando l’intera rete a posteriori, oppure cambiando manualmente i valori scritti sul file generato, avendo cura di non alterare la formattazione di quest’ultimo, pena il non funzionamento del software.
Preliminarmente alla strutturazione della rete, occorre dare un nome a questa, nell’apposita schermata che appare subito dopo la scelta dell’opzione F2 (Figura A1.4): per la rete in esame, si è scelto il nome *Test1s3t*, ad indicare la presenza di tre periodi ed un solo serbatoio.

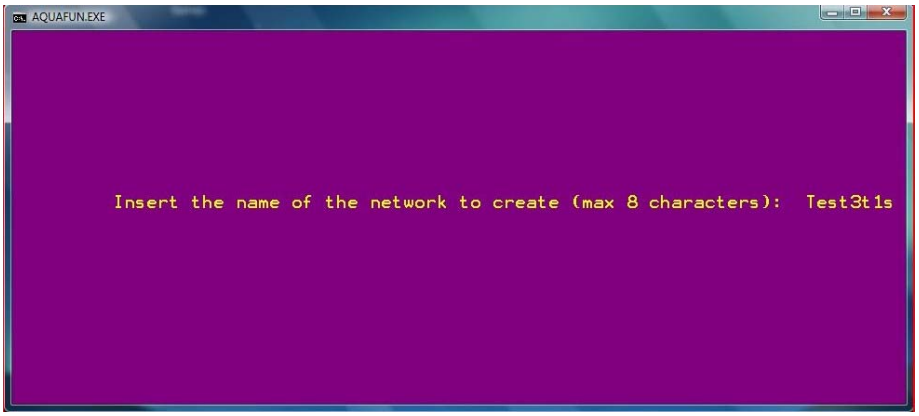

Figura A1.4 – Inserimento del nome della rete

Di seguito si riportano le quattro fasi necessarie alla definizione della rete, e i valori da inserire per il modello definito nel precedente paragrago A1.1. Si indicano in carattere Bookman Old Style i comandi di Ednet, e in *corsivo* gli input da tastiera

Prima fase

GENERAL INFORMATION

Insert the number of periods <int>: *3*

Insert the number of reservoirs <int>: *1*

Insert the number of switches <int>: *2*

In questa fase occorre definire il numero di sorgenti e pozzi presenti, ossia il numero di bacini imbriferi che sono presenti (uno solo, *a*, nel caso in esame), e il numero di elementi cui la risorsa idrica è destinata (uno anch'esso, *u*).

Insert the names of sink and source <str str>: *a u*

Occorre inserire due stringhe di testo, separate necessariamente da uno spazio, relative al nome della sorgente e a quello del pozzo: nel caso in esame sono state chiamate *a* e *u*.

Insert the number of channels <int>: *2*

Si hanno due soli canali, che uniscono rispettivamente il bacino al serbatoio e questo al distretto irriguo.

Al termine di questa fase, viene scritto il file che ha come nome il nome dato alla rete, e come estensione .NT1, per cui "*Test1s3t.NT1*", la cui struttura è mostrata nella seguente Figura A1.5.

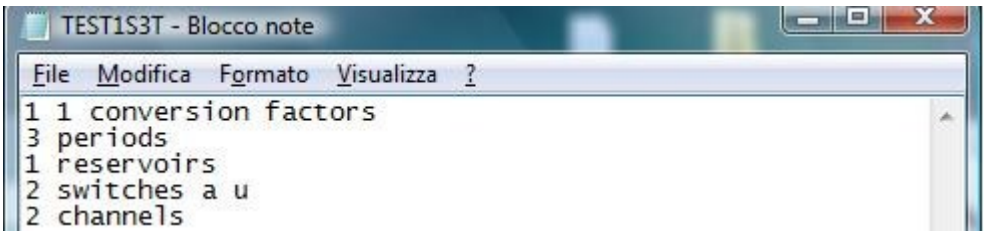

Figura A1.5 – Struttura del file .NT1, aperto con Blocco Note

Seconda fase

INFLOWS INFORMATION

Insert the number of management cycles <int>: *1*

Con cicli si intende, come già introdotto in precedenza, il numero di periodi dopo il quale le informazioni si ripetono uguali.

Insert the 3 inflows for cycle 1 (press enter at the end) <int int …>: *10 20 15*

Si deve aver cura, in questo passo, di inserire tutti gli interi richiesti separandoli da uno spazio, e premendo il tasto Enter al termine dell'input. Occorre osservare altresì che il programma individua automaticamente il numero di afflussi da inserire, e il numero di cicli richiesti, in base ai dati inseriti in precedenza; qualora (non è il caso dell'esempio in esame) siano presenti più cicli, il programma ripete questo input per tutto il numero di cicli necessari.

Al termine della fase corrente, viene scritto un file testuale contenente le informazioni sugli afflussi, con estensione .NT2, per cui, per la rete di esempio "*Test1s3t.NT2*", la cui struttura è riportata nella seguente Figura A1.6.

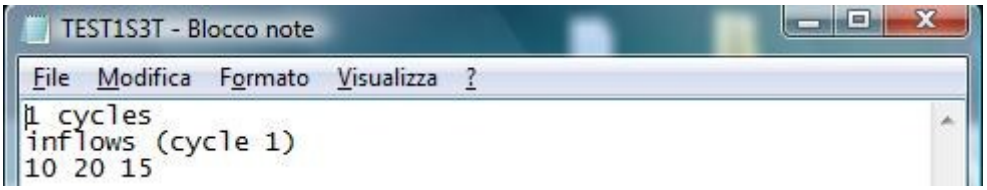

Figura A1.6 – File "Test1s3t.NT2"

Terza fase

RESERVOIRS INFORMATION

Insert reservoir name; evaporation; min., max. and initial storage for each reservoir at period 1 (press enter at the end)

<str int int int>: *res1 1 0 100 50*

Nella definizione del serbatoio per il primo periodo, occorre introdurre anche il valore iniziale dell'invaso.

Insert reservoir name; evaporation; min., and max. storage for each reservoir at period 2 (press enter at the end)

<str int int int>: *res1 1 0 100*

Per tutti i periodi intermedi, ossia quelli compresi tra il primo e l'ultimo, non viene richiesto il valore dell'invaso, ma solo le condizioni al contorno e l'evaporazione, dal momento che il valore di invaso viene calcolato dal software applicando una semplice equazione di bilancio, descritta nel Paragrafo A1.3; nell'esempio discusso in questa sede, il periodo *2* è l'unico intermedio.

Insert reservoir name; evaporation; min., max. and final storage for each reservoir at period 3 (press enter at the end)

<str int int int>: *res1 1 0 100 50*

Occorre inserire il valore finale dell'invaso del serbatoio, per garantire al programma una condizione al contorno mediante cui determinare la soluzione ottima vincolata.

Il risultato della definizione dei serbatoi viene memorizzato in un file con estensione .NT3, che, per il caso in esame, risulta "*Test1s3t.NT3*", come rappresentato nella Figura A1.7, che segue.

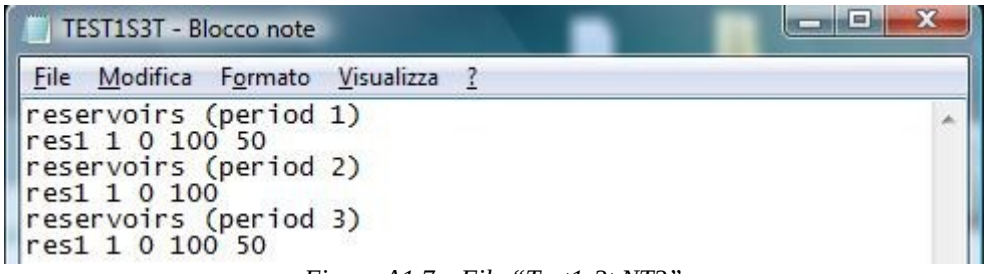

Figura A1.7 – File "Test1s3t.NT3"

Quarta fase

CHANNELS INFORMATION

For each channel (2) at period 1 insert start node, end node, delay time (in periods), minimum flow (press enter at the end) and the utility function (cap., benefits) (press enter at the end)

<str str in tint>

<int int; int int; …>: *a res1 0 0*

1000 0

res1 u 0 2

5 100; 5 70; 5 40; 5 20; 5 10; 5 5; 5 0

Il programma, nell'intestazione, riconosce automaticamente il numero di canali da descrivere ed il periodo di riferimento. Occorre specificare il nodo di partenza e quello di arrivo, l'eventuale ritardo nel trasporto del flusso e la portata minima che deve necessariamente defluire nel canale (assimilabile ad una portata di Deflusso Minimo Vitale); occorre poi definire la funzione di utilità relativa al canale, che, nell'esempio in questione, se riferita al canale di collegamento tra sorgente e serbatoio, è nulla, altrimenti è definita secondo quanto specificato nelle Tabelle A1.2. La stessa schermata si ripete per i periodi successivi, ma in essi i nodi estremi dei canali sono già introdotti automaticamente, memorizzati dalla definizione del primo periodo; per i periodi successivi al primo occorre inserire la struttura delle funzioni di utilità relativa al passo in questione.

Al termine di questa quarta fase, al file scritto viene assegnata l'estensione .NT4: "*Test1s3t.NT4*", come rappresentato nella seguente Figura A1.8.

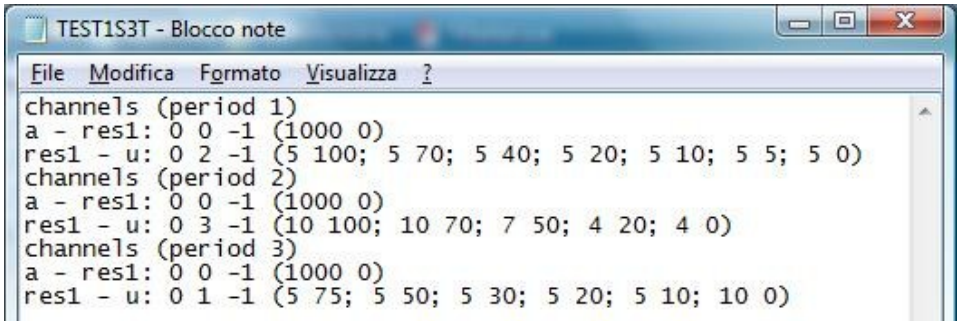

Figura A1.8 – Rappresentazione del file "Test1s3t.NT4", aperto in Blocco note

Può capitare, anche se non accade nell'esempio qui riportato, che il ritardo di trasporto nel flusso sia superiore a *0* durante l'ultimo periodo di simulazione. In tale caso, Aquafun riconosce l'errore di modellizzazione ed interviene con un avviso all'utente, il quale deve imporre una condizione finale sul flusso:

The arc goes out of the horizon. Insert the final condition on flow <int>.

A1.2.2 OTTIMIZZAZIONE DELLA RETE

Terminata la definizione della rete, Aquafun apre una schermata (riportata nella seguente Figura A1.9) mediante cui è possibile scegliere diverse opzioni, che prevedono:

- La possibilità di eseguire e simulare la rete adottando uno dei tre algoritmi di ottimizzazione implementati in Aquafun (Relax, Netflo e Spath, descritti nell'Appendice A3), opzione *"Perform"* che si attiva premendo il tasto F1;
- La possibilità di modificare la rete introdotta, attivando nuovamente Ednet e la procedura descritta nei paragrafi precedenti, opzione *"Modify (Ednet)"* (tasto F2);
- La possibilità di visualizzare (*"View selected network (Ednet)"*), sempre in Ednet, la struttura della rete introdotta (Tasto F3).

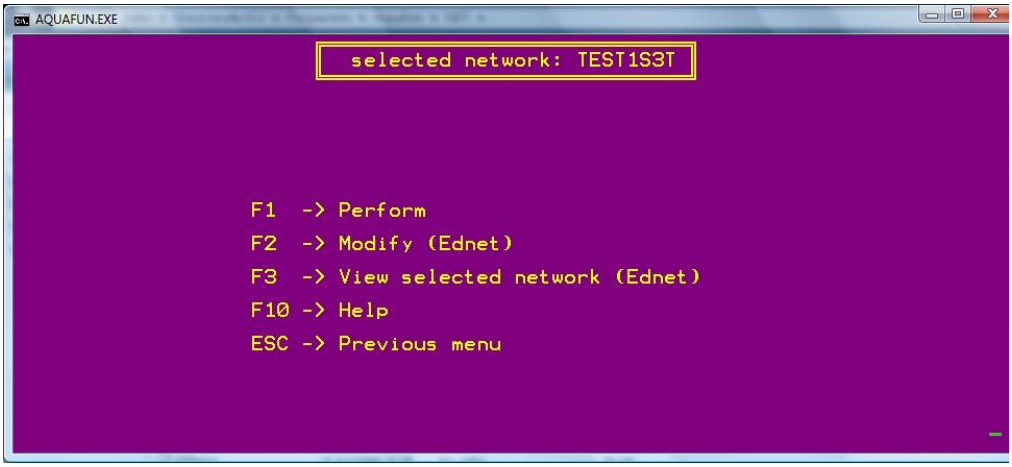

Figura A1.9 – Opzioni di Aquafun possibili sulla rete introdotta

Qualora si voglia visualizzare la rete "Test1s3t", appare una schermata come quella rappresentata nella seguente Figura A1.10:

Figura A1.10 – Visualizzazione della rete; la pressione di un qualsiasi tasto mostra gli altri dati

Maggiormente di interesse, tuttavia, è simulare la rete: pertanto, occorre attivare l'opzione "Perform" che Aquafun mette a disposizione. La schermata che si apre (Figura A1.11, di seguito) consente di scegliere uno dei tre algoritmi di ottimizzazione (Relax con F1, Netflo con F2 e Spath con F3), oppure di introdurne uno nuovo (F4). L'analisi per la rete "Test1s3t" di esempio, si limiterà al solo algoritmo Relax, che, peraltro, è il migliore e quello adottato per l'analisi della rete dello Zambesi, oggetto del presente lavoro di Tesi. La selezione di un algoritmo crea, nella cartella "SOL", un file (anch'esso visualizzabile con un comune editor),

con nome uguale a quello della rete ed estensione ".SO1" se l'algoritmo adottato è Relax, ".SO2" se Netflo e ".SO3" se Spath.

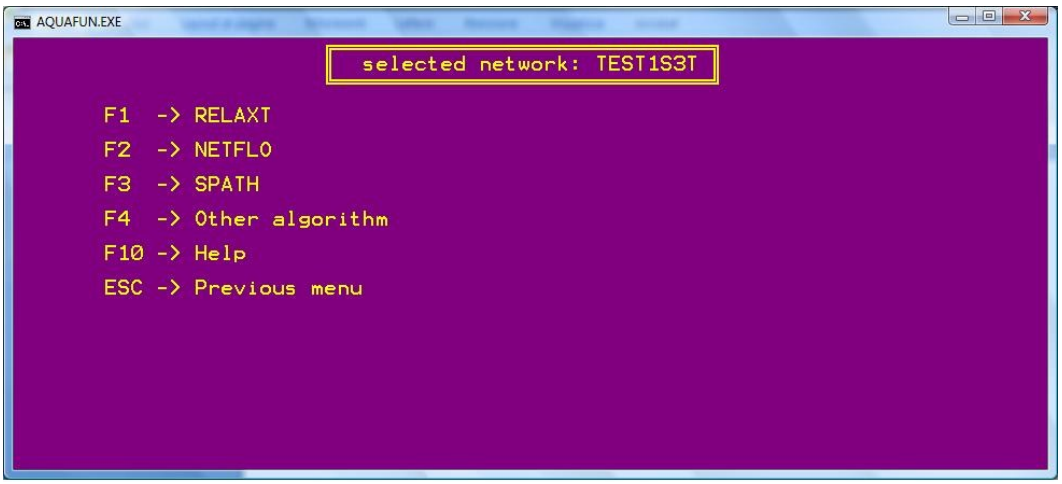

Figura A1.11 – Finestra di scelta dell'algoritmo di ottimizzazione della rete

Per poter utilizzare l'algoritmo Relax (tasto F1), Aquafun deve convertire tutti i dati di input in un unico file, anch'esso visualizzabile con un qualsiasi editor, con estensione .NTP: tale file è provvisorio, viene creato alla pressione del tasto F1 quando si scelgono le opzioni sulla rete (passo precedente, Figura A1.10), e si cancella non appena terminata la procedura di ottimizzazione[1]. La struttura di detto file per la rete di esempio è riportata nella seguente Figura A1.12.

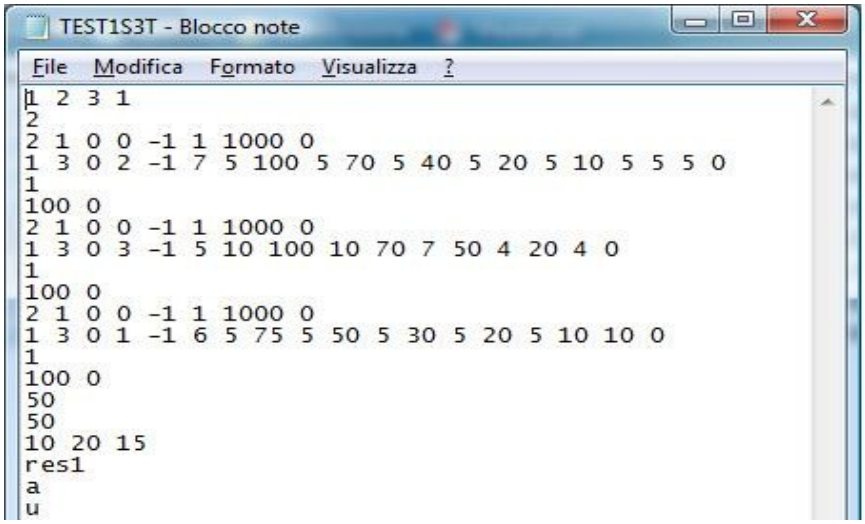

Figura A1.12 – Struttura del file "Test1s3t.NTP", aperto con un editor di testo

La visualizzazione dei risultati prodotti da Relax è possibile tornando al menu principale (Figura A1.3): scegliendo l'opzione "Output" (F3), si seleziona l'algoritmo Relax e poi la rete di interesse. Gli output vengono visualizzati sempre in Ednet, in schermate del tipo di quella riportata nella seguente Figura A1.13.

[1] La scelta dell'opzione "Perform" crea anche un altro file provvisorio, con estensione .NTM, che viene utilizzato da Aquafun per l'ottimizzazione con Spath e Netflo; la struttura di tale file si discosta da quella del medesimo .NTP, ma non viene riportata

Figura A1.13 – Visualizzazione dei risultati di ottimizzazione con l'algoritmo Relax; la pressione di un qualsiasi tasto mostra le altre schermate analoghe

A1.3 IMPLEMENTAZIONE DI AQUAFUN

L'algoritmo Relax calcola la soluzione ottima tramite la stesura e la risoluzione di un problema a minimo costo, che si ottiene cambiando segno alla funzione di utilità.

Più in generale, comunque, Aquafun implementa al suo interno delle "reti fittizie", che replicano, per ogni passo, la struttura topologica della rete di partenza (Figura A1.14 di seguito), come spiegato nel Capitolo 1; inoltre, ad ogni passo, viene risolta la semplice equazione di bilancio

$$s_{t+1} = s_t + a_{t+1} - e_{t+1} - u_t \qquad \text{(A1.1)}.$$

Relax interviene solamente per la scelta del controllo ottimo, tale che sia massimizzata l'utilità, nel rispetto dei vincoli, sia sul periodo, sia sull'intero orizzonte.

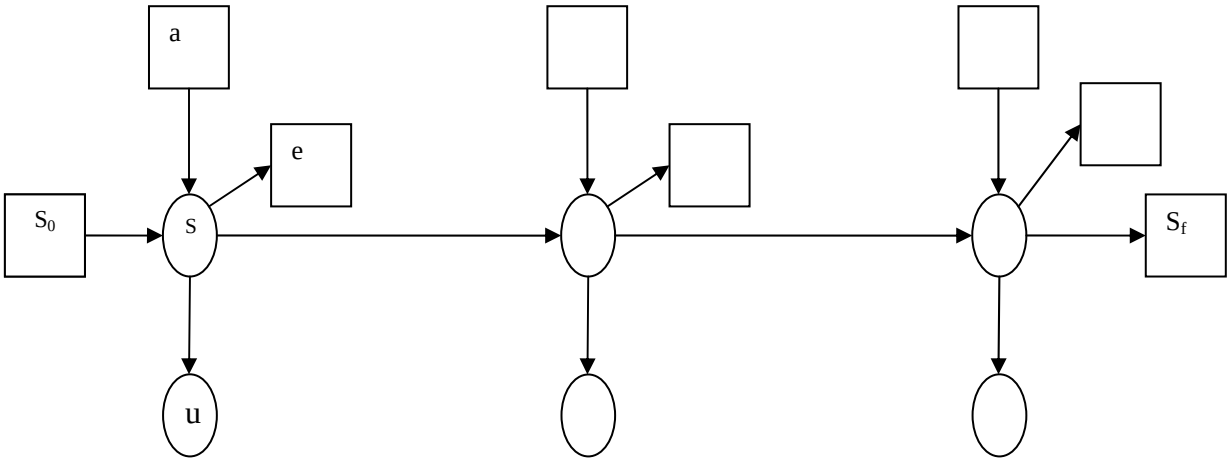

Figura A1.14 – Replicazione per tre periodi della rete di esempio; le frecce orizzontali indicano il passaggio da un periodo al successivo, quelle verticali rappresentano i canali.

Il volume residuo di invaso ad ogni step viene calcolato mediante l'equazione di bilancio (A1.1), mentre il flusso che transita nel canale tra il serbatoio ed il distretto viene calcolato mediante l'ottimizzazione con Relax. A tale proposito, si fa presente che i flussi idrici

vengono ripartiti tra i periodi in modo che la loro somma consenta di chiudere il bilancio nel rispetto dei vincoli, e in modo tale che l'utilità complessiva sia massimizzata (può non essere massima l'utilità sul singolo periodo). Prendendo a riferimento i dati della rete"Test1s3t", si ha che:

$$Evap_{totale} = e_1 + e_2 + e_3 = 1 + 1 + 1 = 3 \quad \text{(A1.2)},$$

$$Afflussi_{Totali} = a_1 + a_2 + a_3 = 10 + 20 + 15 = 45 \quad \text{(A1.3)},$$

$$S_{iniz} = S_{finale} = 50 \text{ quindi } Afflussi_{Totali} - Evap_{Totale} = U_{totali} = 42 \quad \text{(A1.4)}.$$

Alla fine della simulazione, i flussi circolanti nel canale *res1 – u* devono essere *42*. Se si analizza il file di soluzione, si evince che Aquafun alloca al primo periodo un flusso pari a *10*, al secondo un flusso pari a *27* ed al terzo un flusso pari a *5*. Le utilità corrispondenti ai singoli periodi vengono calcolate moltiplicando il valore dell'utilità marginale di ogni gradino per l'ampiezza del gradino stesso, o una sua porzione. Pertanto:

$$\begin{cases} u_1 = 10 \rightarrow utilità_1 = 5 \times 100 + 5 \times 70 = 850 \\ u_2 = 27 \rightarrow utilità_2 = 10 \times 100 + 10 \times 70 + 7 \times 50 = 2050 \\ u_3 = 5 \rightarrow utilità_3 = 5 \times 75 = 375 \end{cases} \quad \text{(A1.5)},$$

dove i valori sono quelli definiti durante la descrizione della rete e delle funzioni di utilità (Paragrafo A1.1). La somma delle utilità relative ai singoli periodi ricavate con la precedente Equazione (A1.5) consentono di ricavare il beneficio totale sull'orizzonte, pari a

$$Utilità_{Totale} = utilità_1 + utilità_2 + utilità_3 = 850 + 2050 + 375 = 3275 \quad \text{(A1.6)}.$$

Al fine di verificare che l'utilità ottenuta fosse realmente quella massima, sono stati eseguiti alcuni test, qui non riportati, nei quali è stata mutata la ripartizione dei flussi nei vari periodi, mantenendo invariati i vincoli: le utilità totali ottenute sono sempre inferiori a quella prodotta da Aquafun, anche se le utilità sui singoli periodi possono essere localmente più alte. Pertanto, il programma alloca i flussi cercandone una distribuzione che renda massimo il beneficio complessivo sull'intero orizzonte.

A2

SCRIPT E FUNZIONI DI MATLAB

In questa Appendice al lavoro si riportano tutti gli script e le funzioni realizzate in Matlab per il presente lavoro di Tesi

A2.1 FUNZIONE “estr_mnvr_year.m”

La presente funzione è stata utilizzata durante l’analisi delle serie di afflussi (Capitolo 5) per estrarre i valori medi e le varianze dei valori di afflusso raggruppati per anno. Viene prodotto anche un grafico che riporta gli andamenti di medie e varianze.
Di seguito si riporta lo script della funzione:

```
function [MVAR] = estr_mnvr_year(fl);
lg_per = 12;
n_per = 12;
j=0;
acc_per=0;
MVAR = [];
MD = [];
VR = [];
%CONSTRUCTION OF THE MATRIX
for j=1:n_per
   calc=fl((1+acc_per):(lg_per+acc_per));
   acc_per=acc_per+lg_per;
   MD= [MD; mean(calc)];
   VR= [VR; var(calc)];
end
%--------------------------------------------------------------------------
%Following instructions simplify visualization of values
MD_RED=MD./(10^3);
VR_RED=VR./(10^7);
'First column vector of mean values, Second column vector of variances'
'BE CAREFUL: MEAN ARE DIVIDED PER 1000!!!'
'BE CAREFUL: VARIANCES ARE DIVIDED PER 10^7!!!'
%--------------------------------------------------------------------------
MVAR=[MD_RED,VR_RED];
%PLOTTING RESULTS
yr=[1:n_per];
figure
subplot(2,1,1),plot(yr,MD_RED,'ro:')
grid
axis([1 n_per (min(MD_RED)-2) (max(MD_RED)+2)])
xlabel('Year')
ylabel('Mean inflows [*10^3]')
title('MEAN INFLOW PER YEAR')
subplot(2,1,2),plot(yr,VR_RED,'mo:')
grid
axis([1 n_per (min(VR_RED)-1) (max(VR_RED)+1)])
```

```
xlabel('Year')
ylabel('Variance of inflows [*10^7]')
title('VARIANCE OF INFLOWS PER YEAR')
```

A2.2 FUNZIONE "estr_mnvr_month.m"

Mediante questa funzione è possibile estrarre valor medio, varianza e produrre un grafico degli afflussi raggruppati per mese (Capitolo 5). Lo script della funzione è il seguente:

```
function [MVAR] = estr_mnvr_month(fl);
lg_per = 12;
j=0;
val_month = [];
MVAR = [];
MD = [];
VR = [];
%CONSTRUCTION OF THE MATRIX
for j=1:lg_per
   val_month=fl(j:lg_per:length(fl));
   MD= [MD; mean(val_month)];
   VR= [VR; var(val_month)];
   val_month=[];
end
%--------------------------------------------------------------------------
%Following instructions simplify visualization of values
MD_RED=MD./(10^3);
VR_RED=VR./(10^7);
'First column vector of mean values, Second column vector of variances'
'BE CAREFUL: MEAN ARE DIVIDED PER 1000!!!'
'BE CAREFUL: VARIANCES ARE DIVIDED PER 10^7!!!'
'Month order on row: Oct Nov Dec Jan Feb Mar Apr May Jun Jul Aug Sep'
'On x axis, 1 corresponds to Oct and 12 to Sep'
%--------------------------------------------------------------------------
MVAR=[MD_RED,VR_RED];
% PLOTTING RESULTS
figure
subplot(2,1,1),plot(MD_RED,'ro:')
grid
axis([1 lg_per (min(MD_RED)-2) (max(MD_RED)+2)])
xlabel('Month')
ylabel('Mean of inflows [*10^3]')
title('MEAN INFLOWS PER MONTH')
subplot(2,1,2),plot(VR_RED,'mo:')
grid
axis([1 lg_per (min(VR_RED)-1) (max(VR_RED)+1)])
xlabel('Month')
ylabel('Variance of inflows [*10^7]')
title('VARIANCE OF INFLOWS PER MONTH')
```

A2.3 FUNZIONE “estr_mnvr_mavg.m”

Tale funzione è utilizzata, sempre nell’ambito dell’analisi degli afflussi (Capitolo 5), per estrarre valori medi e varianze di dati di afflusso raggruppati su finestre mobili di ampiezza fissata (tipicamente in modo tale da riuscire a coprire un intero anno), che si spostano in avanti nel tempo. Lo script è il seguente:

```
function [MVAR] = estr_mnvr_mavg(fl);
op_wind=12;
lg_aff=length(fl);
j=0;
val_window = [];
MVAR = [];
MD = [];
VR = [];
%CONSTRUCTON OF THE MATRIX
for j=(1+(op_wind/2)):(lg_aff-(op_wind/2))
   val_window=fl((j-(op_wind/2)):(j+(op_wind/2)));
   MD= [MD; mean(val_window)];
   VR= [VR; var(val_window)];
   val_window=[];
end
%--------------------------------------------------------------------------
%Following instructions simplify visualization of values
MD_RED=MD./(10^3);
VR_RED=VR./(10^7);
'First column vector of mean values, Second column vector of variances'
'BE CAREFUL: MEAN ARE DIVIDED PER 1000!!!'
'BE CAREFUL: VARIANCES ARE DIVIDED PER 10^7!!!'
%--------------------------------------------------------------------------
MVAR=[MD_RED,VR_RED];
%PLOTTING RESULTS
figure
plot(MD_RED,'r+:')
grid
axis([1 lg_aff (min(MD_RED)-2) (max(MD_RED)+2)])
xlabel('')
ylabel('Mean value of inflow [*10^3]')
title('MOVING WINDOW MEAN VALUES')
figure
plot(VR_RED,'m+:')
grid
axis([1 lg_aff (min(VR_RED)-1) (max(VR_RED)+1)])
xlabel('')
ylabel('Variance of the inflows [*10^7]')
title('MOVING WINDOW VARIANCE')
```

A2.4 FUNZIONE “rd_arzero.m”

La presente funzione è quella utilizzata per creare la maggior parte delle serie sintetiche di afflussi (Capitolo 5), utilizzate per l’addestramento, la taratura e la validazione delle reti neurali. Il richiamo della presente funzione ne attiva anche un’altra (“rapp_chann.m”, Paragrafo A2.6), mediante la quale i valori di afflusso ad ogni periodo vengono ripartiti tra i quattro serbatoi che compongono la rete del fiume Zambesi.

```
function [ARZERO] = rd_arzero(fl, coeff, start , stop, ratioes);
if start>stop
   printf('BE CAREFUL!!! Starting value of the serie cannot be bigger than final value')
   return
else
   range = [3000,10000];
   ARZERO = [];
```

```
   lt_flw = length(fl);
   i = 0; k = 0; j = 0;
   avg_fl = mean(fl);
   std_dev_fl = sqrt(var(fl));
   if coeff<1
      fprintf('<coeff> is not correct')
      return
   elseif coeff>4
      fprintf('<coeff> is not correct')
      return
   else
      %CREATION OF FILE NAMES
      support = sprintf('ARZ%04d.NT2',1); %filename
      length_files = numel(support);
      file_names = mat2cell(sprintf('ARZ%04d.NT2',start:stop),1,(length_files*ones(1,(stop-start+1))));
      names_ratioes = mat2cell(sprintf('ARZ%04d.MRO',start:stop),1,(length_files*ones(1,(stop-start+1))));
      %VIEWING FILENAMES
      for k = 1:numel(file_names)
         disp(file_names{k})
         disp(names_ratioes{k})
      end
      %AR(0) MODEL AND SAVING DATA
      for j = 1:numel(file_names)
         for i=1:lt_flw
            ARZERO(i)=avg_fl+coeff*std_dev_fl*(0.5-rand);
         end
         avg_arzero = mean(ARZERO);
         arzero_int = max(0,round(ARZERO))';
         arzero_rat = rapp_chann(arzero_int,ratioes);
         arzero_rat_int = max(0,round(arzero_rat));
         if ((avg_arzero>range(1))&(avg_arzero<range(2)))
            fid = fopen(file_names{j},'wt');
            fprintf(fid, '1 cycles\ninflows (cycle 1)\n');
            fprintf(fid, '%d\n',arzero_int);
            fclose(fid);
            %CREATION OF .MRO FILE
            fid_rat = fopen(names_ratioes{j},'wt');
            fprintf(fid_rat,'%d\n',arzero_rat_int);
            fclose(fid_rat);
         end
         ARZERO = []; %Deleting existing ARZERO vector
      end
   end
end
```

A2.5 FUNZIONE "rd_arone.m"

Mediante la presente funzione sono state create altre serie sintetiche di afflussi (come descritto nel Capitolo 5), utilizzate prevalentemente per la validazione della Rete Neurale e successive analisi. Anche in questa funzione viene richiamato il codice per la suddivisione degli afflussi nei diversi serbatoi ("rapp_chann.m", Paragrafo A2.6).

```
function [ARONE] = rd_arone(fl, start, stop, ratioes)
range = [3000,10000];
ARONE = [];
lt_flw = length(fl); %the length of the vector in input is necessary to stop cycle <for>
i = 0; k = 0; j = 0;%counter for the cycle
if start>stop
   printf('BE CAREFUL!!! Starting value of the serie cannot be bigger than final value')
   return
else
   %CREATION OF FILE NAMES
   support = sprintf('ARO%04d.NT2',1); %filename
   length_files = numel(support);
   file_names = mat2cell(sprintf('ARO%04d.NT2',start:stop),1,(length_files*ones(1,(stop-start+1))));
```

```
   names_ratioes = mat2cell(sprintf('ARO%04d.MRO',start:stop),1,(length_files*ones(1,(stop-start+1))));
   %VIEWING FILENAMES
   for k = 1:numel(file_names)
      disp(file_names{k})
      disp(names_ratioes{k})
   end
   %AR(1) MODEL AND SAVING DATA
   for j = 1:numel(file_names)
      coeff = rand;
      ARONE(1) = round(mean(fl)+randn);
      for i=2:lt_flw
         ARONE(i) = round(mean(fl) + randn + coeff*(fl(i-1)-mean(fl)));
      end
      avg_arone = mean(ARONE)
      arone_int = max(0, round(ARONE))';
      arone_rat_int = max(0,round(rapp_chann(arone_int,ratioes)));
      if ((avg_arone>range(1))&(avg_arone<range(2)))
         fid = fopen(file_names{j},'wt')
         fprintf(fid, '1 cycles\ninflows (cycle 1)\n')
         fprintf(fid, '%d\n',arone_int)
         fclose(fid)
         %CREATION OF .MRO FILE
         fid_rat = fopen(names_ratioes{j},'wt');
         fprintf(fid_rat,'%d\n',arone_rat_int);
         fclose(fid_rat);
      end
      ARONE = []; %Deleting existing ARONE vector
   end
end
```

A2.6 FUNZIONE “rapp_chann.m”

Questa funzione viene richiamata all’interno delle precedenti (“rd_arzero.m” e “rd_arone.m”, Paragrafi A2.4 e A2.5), e serve a suddividere gli afflussi relativi ad ogni periodo di simulazione in base ai rapporti di ripartizione tra i diversi serbatoi della rete idrica.

```
function [channels_ratioes]=rapp_chann(flow,ratio)
nro_res = 4;
number_flows = repmat(flow,1,nro_res)';
%-------------------------------------------------------------------------
%BUILDING THE INFLOWS VECTOR PER RESERVOIR
channels_ratioes = [];
rows = [];
i=0;
for i=1:length(ratio)
   rows(i)=number_flows(i)*ratio(i);
end
rows(nro_res:nro_res:numel(rows)) = 0;
rows_round = round(rows);
l=1;
%EXTRACTION DIFFERENCE FOR THE LAST INFLOW PER PERIOD
for j=4:4:numel(rows)
   rows_round(j)=flow(l)-(rows_round(j-1)+rows_round(j-2)+rows_round(j-3));
   l=l+1;
end
channels_ratioes = rows_round';
```

A2.7 FUNZIONE "train_setting.m"

Mediante la presente funzione è possibile costruire le matrici di addestramento alle Reti Neurali (Capitolo 3). Con tale funzione, dal file di soluzione di Aquafun vengono estratti i valori di afflusso ad ognuno dei serbatoi, dei volumi di invaso e dei rilasci, per ogni periodo di simulazione. Nella tabella finale, che viene ordinatamente scritta in un apposito file testuale, sono contenuti solamente i valori dei periodi che non sono stati esclusi per evitare gli effetti di bordo. I dati di input alla matrice sono costituiti dal file *.MRO* (ottenuto contestualmente alla generazione delle serie di afflusso, possibile con le Funzioni 006 e 007), dal file *.MRL* (ottenibile dalla funzione "*levels_creator.m*"), e dal file *.MRC* (che può essere ottenuto con la funzione "*releases_creator.m*")

```
function [MAT_TRAIN]=train_setting(flow_per_res,res,rel,name)
num_per_begin = 12;
num_per_end = 12;
nro_res_per = 4;
nro_flow_per = 4;
nro_rel_per = 5;
%--------------------------------------------------------------------------
%STEP 0
flow_mod = load('-ascii', flow_per_res);
res_mod = load('-ascii', res);
rel_mod = load('-ascii', rel);
%--------------------------------------------------------------------------
%STEP 1
flow_cut = flow_mod(((num_per_begin*nro_flow_per)+1):(end-(num_per_end*nro_flow_per)));
res_cut = res_mod(((num_per_begin*nro_res_per)+1):(end-(num_per_end*nro_res_per)));
rel_cut = rel_mod(((num_per_begin*nro_rel_per)+1):(end-(num_per_end*nro_rel_per)));
%--------------------------------------------------------------------------
%STEP 2
flows_KA = []; flows_IT = []; flows_KF = []; flows_CB = [];
res_KA = []; res_IT = []; res_KF = []; res_CB = [];
rel_KA = []; rel_IT = []; rel_KF1 = []; rel_KF2 = []; rel_CB = [];
flows_KA = flow_cut(1:4:length(flow_cut));
flows_IT = flow_cut(2:4:length(flow_cut));
flows_KF = flow_cut(3:4:length(flow_cut));
flows_CB = flow_cut(4:4:length(flow_cut));
res_KA = res_cut(1:4:length(res_cut));
res_IT = res_cut(2:4:length(res_cut));
res_KF = res_cut(3:4:length(res_cut));
res_CB = res_cut(4:4:length(res_cut));
rel_KA = rel_cut(1:5:length(rel_cut));
rel_IT = rel_cut(2:5:length(rel_cut));
rel_KF1 = rel_cut(3:5:length(rel_cut));
rel_KF2 = rel_cut(4:5:length(rel_cut));
rel_CB = rel_cut(5:5:length(rel_cut));
%--------------------------------------------------------------------------
%STEP 3
KA = [flows_KA, res_KA]; IT = [flows_IT, res_IT]; KF = [flows_KF, res_KF]; CB = [flows_CB, res_CB];
MAT_TRAIN = floor(([KA, IT, KF, CB, rel_KA, rel_IT, rel_KF1, rel_KF2, rel_CB]));
%--------------------------------------------------------------------------
%STEP 4
periods = [(1+num_per_begin):(numel(rel_CB)+num_per_begin)];
fid = fopen(name,'wt');
fprintf(fid,'COMPLETE TRAINING MATRIX WITH INFLOWS, LEVELS AND RELEASES PER RESERVOIR PER
PERIOD\n');
fprintf(fid,'P = Period of time; I = inflow; L = level; R = release; KA = Kariba; IT = Itezhitezhi; KF = Kafue; CB =
Cabora\n');
fprintf(fid,'Not considered %d periods of time at the beginning and %d period at the end of the serie\n\n',
num_per_begin, num_per_end);
fprintf(fid,' P \tKA  I\tKA  L\tIT  I\tIT  L\tKF  I\tKF  L\tCB  I\tCB  L\t\tKA  R\tIT  R\tKF1 R\tKF2 R\tCB  R\n');
for k = 1:numel(periods)
   fprintf(fid,'\n%03d\t',periods(k));
   fprintf(fid,'%05d\t',floor(KA(k,1)));
```

```
    fprintf(fid,'%05d\t',floor(KA(k,2)));
    fprintf(fid,'%05d\t',floor(IT(k,1)));
    fprintf(fid,'%05d\t',floor(IT(k,2)));
    fprintf(fid,'%05d\t',floor(KF(k,1)));
    fprintf(fid,'%05d\t',floor(KF(k,2)));
    fprintf(fid,'%05d\t',floor(CB(k,1)));
    fprintf(fid,'%05d\t\t',floor(CB(k,2)));
    fprintf(fid,'%05d\t',floor(rel_KA(k)));
    fprintf(fid,'%05d\t',floor(rel_IT(k)));
    fprintf(fid,'%05d\t',floor(rel_KF1(k)));
    fprintf(fid,'%05d\t',floor(rel_KF2(k)));
    fprintf(fid,'%05d\t',floor(rel_CB(k)));
end
fclose(fid);
%--------------------------------------------------------------------------
%DIMENSIONAL CONTROL
flow_per_res_dim = size(flow_per_res)
mat_mod_dim = [size(flow_mod), size(res_mod), size(rel_mod)]
mat_cut_dim = [size(flow_cut), size(res_cut), size(rel_cut)]
size_flows = [size(flows_KA); size(flows_IT); size(flows_KF); size(flows_CB)]
size_res = [size(res_KA); size(res_IT); size(res_KF); size(res_CB)]
size_rel = [size(rel_KA); size(rel_IT); size(rel_KF1); size(rel_KF2); size(rel_CB)]
mat_train_dim = [size(MAT_TRAIN)]
```

A2.8 FUNZIONE “levels_creator.m”

Utilizzando questa funzione è possibile estrarre, dal file di soluzione di Aquafun, tutti i valori relativi agli invasi per ogni periodo della simulazione. Il file così ottenuto (in formato *.MRL*) può essere utilizzato per la costruzione della matrice di addestramento (funzione *“train_setting.m”*).

```
function [MRL_file] = levels_creator(name_sol, name_mrl)
%STEP FOR READING FILE AND PREPARING FOR EXTRACTING VALUES
read_sol = textread(name_sol, '%s', 'delimiter', '\n');              %Reading file
char_read_sol = char(read_sol);                                      %Transformation in characters
res_string = strfind(read_sol, 'invaso  ');                          %Finding interest rows
no_res_string = cellfun('isempty', res_string);                      %No interests row
number_res_string = find(no_res_string == 0);                        %Number of rows of interests
string_char = [];                                                    %Rows container
%STEP FOR EXTRACTING ROWS CONTAINING RESERVOIRS INFORMATION
for i = 1:numel(number_res_string)
    string_char = char(string_char, char_read_sol(number_res_string(i),:));
end
strchar_size = size(string_char);
%STEP FOR BUILDING RESERVOIRS VOLUMES VECTOR IN NUMERIC FORMAT
%Remember first line of <string_char> is empty: character informations
%start from line 2.
spaces = [];                                                         %Vector of blank spaces position
lev_num = [];                                                        %Vector containing numeric reservoirs data.
for j=2:strchar_size(1)
    spaces = find(string_char(j,:) == ' ');                          %Finds the blank spaces in each line of the vector
    lev_num(j-1) = str2num(string_char(j,(spaces(3)+1:spaces(4)-1)));   %The value of interest is between the third
and the fourth blank space
end
%STEP FOR WRITING TEXT FILE
fid = fopen(name_mrl, 'wt')
for k = 1: length(lev_num)
    fprintf(fid, '%d\n', lev_num(k));
end
fprintf(fid, '43140\n3045\n860\n36250\n');                            %Final values of reservoirs volume: contour
conditions, the same for each network
fclose(fid)
%CONTROL TESTS
sprintf('<lev_num> elements: %d', numel(lev_num))
```

A2.9 FUNZIONE "releases_creator.m"

Analogamente alla precedente, la presente funzione consente di estrarre dal file di soluzione prodotto da Aquafun, i valori dei rilasci dai serbatoi, per poter costruire la matrice di addestramento ed eseguire tutte le operazioni necessarie all'addestramento ed alle simulazioni con le Reti Neurali. Il file prodotto in uscita da questa funzione ha formato *.MRC*.

```
function [MRC_file] = releases_creator(name_sol,name_mrc)
%Reading file and converting it into a character array
read_sol = textread(name_sol, '%s', 'delimiter', '\n');
char_read_sol = char(read_sol);
%Finding rows containing period reference (i.e. the word "mese")
period_ref = strfind(read_sol, 'mese');
no_period_ref = cellfun('isempty',period_ref);
rows_per = find(no_period_ref == 0);
%FIRST NUMEL TEST
numel(rows_per)
%Preparing arrays for releases data containing
IT_rel = zeros(144,1);
KF1_rel =zeros(144,1);
KF2_rel = zeros(144,1);
CB_rel = zeros(144,1);
KA_rel = zeros(144,1);
spaces = [];
Rel = {}; Rel_empty = {}; Rel_full = {};
%Cycle which operates from Period 1 to Period 142 (not operates on last two
%periods)
for i=1:(numel(rows_per)-2)
   %Extracting rows of interestd
   extraction = char_read_sol(rows_per(i)+1:rows_per(i+1)-1,:);
   extr_str = cellstr(extraction);
   %Extraction characters of interests
   Rel{1,1} = strfind(extr_str, 'Itezhi-Kafue (ritardo 2): ');
   Rel{1,2} = strfind(extr_str, 'Kafue-pozzo: ');
   Rel{1,3} = strfind(extr_str, 'Kafue-Chirundu: ');
   Rel{1,4} = strfind(extr_str, 'Kariba-Chirundu: ');
   Rel{1,5} = strfind(extr_str, 'Cabora-pozzo: ');
   Rel_empty{1,1} = cellfun('isempty', Rel{:,1});
   Rel_empty{1,2} = cellfun('isempty', Rel{:,2});
   Rel_empty{1,3} = cellfun('isempty', Rel{:,3});
   Rel_empty{1,4} = cellfun('isempty', Rel{:,4});
   Rel_empty{1,5} = cellfun('isempty', Rel{:,5});
   Rel_full{1,1} = find(Rel_empty{:,1} == 0);
   Rel_full{1,2} = find(Rel_empty{:,2} == 0);
   Rel_full{1,3} = find(Rel_empty{:,3} == 0);
   Rel_full{1,4} = find(Rel_empty{:,4} == 0);
   Rel_full{1,5} = find(Rel_empty{:,5} == 0);
   %If the rows of interests exists, the value written in these rows is
   %converted into numeric format and written into the array, else in the
   %vector is written 0
   if Rel_full{1,1}
      row = extraction(Rel_full{1,1},:);
      spaces = find(row == ' ');
      IT_rel(i)= str2num(row(((spaces(4)+1):(spaces(5)-1))));
      row = []; spaces = [];
   else IT_rel(i) = 0;
   end
   if Rel_full{1,2}
      row = extraction(Rel_full{1,2},:);
      spaces = find(row == ' ');
      KF1_rel(i)= str2num(row(((spaces(2)+1):(spaces(3)-1))));
      row = []; spaces = [];
   else KF1_rel(i) = 0;
   end
   if Rel_full{1,3}
      row = extraction(Rel_full{1,3},:);
```

```
        spaces = find(row == ' ');
        KF2_rel(i)= str2num(row(((spaces(2)+1):(spaces(3)-1))));
        row = []; spaces = [];
    else KF2_rel(i) = 0;
    end
    if Rel_full{1,4}
        row = extraction(Rel_full{1,4},:);
        spaces = find(row == ' ');
        KA_rel(i)= str2num(row(((spaces(2)+1):(spaces(3)-1))));
        row = []; spaces = [];
    else KA_rel(i) = 0;
    end
    if Rel_full{1,5}
            row = extraction(Rel_full{1,5},:);
            spaces = find(row == ' ');
            CB_rel(i)= str2num(row(((spaces(2)+1):(spaces(3)-1))));
            row = []; spaces = [];
        else CB_rel(i) = 0;
    end
    clear Rel Rel_empty Rel_full
end
%Last but one period (143 if used a common Aquafun Zambesi Watershed
%simulation)
extraction = char_read_sol(rows_per(143)+1:rows_per(144)-1,:);
extr_str = cellstr(extraction);
Rel{1,1} = strfind(extr_str, 'Itezhi-futuro: ');
Rel{1,2} = strfind(extr_str, 'Kafue-pozzo: ');
Rel{1,3} = strfind(extr_str, 'Kafue-Chirundu: ');
Rel{1,4} = strfind(extr_str, 'Kariba-Chirundu: ');
Rel{1,5} = strfind(extr_str, 'Cabora-pozzo: ');
Rel_empty{1,1} = cellfun('isempty', Rel{:,1});
Rel_empty{1,2} = cellfun('isempty', Rel{:,2});
Rel_empty{1,3} = cellfun('isempty', Rel{:,3});
Rel_empty{1,4} = cellfun('isempty', Rel{:,4});
Rel_empty{1,5} = cellfun('isempty', Rel{:,5});
Rel_full{1,1} = find(Rel_empty{:,1} == 0);
Rel_full{1,2} = find(Rel_empty{:,2} == 0);
Rel_full{1,3} = find(Rel_empty{:,3} == 0);
Rel_full{1,4} = find(Rel_empty{:,4} == 0);
Rel_full{1,5} = find(Rel_empty{:,5} == 0);
if Rel_full{1,1}
    row = extraction(Rel_full{1,1},:);
    spaces = find(row == ' ');
    IT_rel(143)= str2num(row(((spaces(2)+1):(spaces(3)-1))));
    row = []; spaces = [];
else IT_rel(143) = 0;
end
if Rel_full{1,2}
    row = extraction(Rel_full{1,2},:);
    spaces = find(row == ' ');
    KF1_rel(143)= str2num(row(((spaces(2)+1):(spaces(3)-1))));
    row = []; spaces = [];
else KF1_rel(143) = 0;
end
if Rel_full{1,3}
    row = extraction(Rel_full{1,3},:);
    spaces = find(row == ' ');
    KF2_rel(143)= str2num(row(((spaces(2)+1):(spaces(3)-1))));
    row = []; spaces = [];
else KF2_rel(143) = 0;
end
if Rel_full{1,4}
    row = extraction(Rel_full{1,4},:);
    spaces = find(row == ' ');
    KA_rel(143)= str2num(row(((spaces(2)+1):(spaces(3)-1))));
    row = []; spaces = [];
else KA_rel(143) = 0;
end
if Rel_full{1,5}
```

```
        row = extraction(Rel_full{1,5},:);
        spaces = find(row == ' ');
        CB_rel(143)= str2num(row(((spaces(2)+1):(spaces(3)-1))));
        row = []; spaces = [];
    else CB_rel(143) = 0;
end
clear Rel Rel_empty Rel_full
%Last period
extraction = char_read_sol(rows_per(144)+1:end,:);
extr_str = cellstr(extraction);
Rel{1,1} = strfind(extr_str, 'Itezhi-futuro: ');
Rel{1,2} = strfind(extr_str, 'Kafue-pozzo: ');
Rel{1,3} = strfind(extr_str, 'Kafue-Chirundu: ');
Rel{1,4} = strfind(extr_str, 'Kariba-Chirundu: ');
Rel{1,5} = strfind(extr_str, 'Cabora-pozzo: ');
Rel_empty{1,1} = cellfun('isempty', Rel{:,1});
Rel_empty{1,2} = cellfun('isempty', Rel{:,2});
Rel_empty{1,3} = cellfun('isempty', Rel{:,3});
Rel_empty{1,4} = cellfun('isempty', Rel{:,4});
Rel_empty{1,5} = cellfun('isempty', Rel{:,5});
Rel_full{1,1} = find(Rel_empty{:,1} == 0);
Rel_full{1,2} = find(Rel_empty{:,2} == 0);
Rel_full{1,3} = find(Rel_empty{:,3} == 0);
Rel_full{1,4} = find(Rel_empty{:,4} == 0);
Rel_full{1,5} = find(Rel_empty{:,5} == 0);
if Rel_full{1,1}
    row = extraction(Rel_full{1,1},:);
    spaces = find(row == ' ');
    IT_rel(144)= str2num(row(((spaces(2)+1):(spaces(3)-1))));
    row = []; spaces = [];
else IT_rel(144) = 0;
end
if Rel_full{1,2}
    row = extraction(Rel_full{1,2},:);
    spaces = find(row == ' ');
    KF1_rel(144)= str2num(row(((spaces(2)+1):(spaces(3)-1))));
    row = []; spaces = [];
else KF1_rel(144) = 0;
end
if Rel_full{1,3}
    row = extraction(Rel_full{1,3},:);
    spaces = find(row == ' ');
    KF2_rel(144)= str2num(row(((spaces(2)+1):(spaces(3)-1))));
    row = []; spaces = [];
else KF2_rel(144) = 0;
end
if Rel_full{1,4}
    row = extraction(Rel_full{1,4},:);
    spaces = find(row == ' ');
    KA_rel(144)= str2num(row(((spaces(2)+1):(spaces(3)-1))));
    row = []; spaces = [];
else KA_rel(144) = 0;
end
if Rel_full{1,5}
        row = extraction(Rel_full{1,5},:);
        spaces = find(row == ' ');
        CB_rel(144)= str2num(row(((spaces(2)+1):(spaces(3)-1))));
        row = []; spaces = [];
    else CB_rel(144) = 0;
end
clear Rel Rel_empty Rel_full
%DIMENSIONALLY CONTROLS
sprintf('Dimensional control of periods\n')
MRC_file = [IT_rel, KF1_rel, KF2_rel, CB_rel, KA_rel];
sprintf('Numel IT = %d\t', numel(IT_rel))
sprintf('Numel KF1 = %d\t',numel(KF1_rel))
sprintf('Numel KF2 = %d\t',numel(KF2_rel))
sprintf('Numel CB = %d\t', numel(CB_rel))
sprintf('Numel KA = %d\t', numel(KA_rel))
```

```
%Error abort
if    (not(logical(numel(IT_rel) == numel(KF1_rel)))) && (      ...
       not(logical(numel(IT_rel) == numel(KF2_rel)))) && (      ...
       not(logical(numel(IT_rel) == numel(KA_rel))))  && (      ...
       not(logical(numel(IT_rel) == numel(CB_rel)))) && (       ...
       not(logical(numel(KF1_rel) == numel(KF2_rel)))) && (     ...
       not(logical(numel(KF1_rel) == numel(CB_rel)))) && (      ...
       not(logical(numel(KF1_rel) == numel(KA_rel)))) && (      ...
       not(logical(numel(KF2_rel) == numel(KA_rel)))) && (      ...
       not(logical(numel(KF2_rel) == numel(CB_rel)))) && (      ...
       not(logical(numel(KA_rel) == numel(CB_rel))))
   sprintf('Error in compiling files: retry please')
   return
else
   %Writing text file
   fid = fopen(name_mrc, 'wt')
   for k = 1:numel(IT_rel)
      fprintf(fid, '%d\n', KA_rel(k));
      fprintf(fid, '%d\n', IT_rel(k));
      fprintf(fid, '%d\n', KF1_rel(k));
      fprintf(fid, '%d\n', KF2_rel(k));
      fprintf(fid, '%d\n', CB_rel(k));
   end
   fclose(fid)
end
%OTHER CONTROLS
sum_elem = (numel(IT_rel)+numel(KF1_rel)+numel(KF2_rel)+numel(CB_rel)+numel(KA_rel));
sprintf('<MRC> elements: %d', (numel(IT_rel)+numel(KF1_rel)+numel(KF2_rel)+numel(CB_rel)+numel(KA_rel)))
if (mod(sum_elem,144)==0)
   sprintf('OK')
else sprintf('NO')
end
```

A2.10 FUNZIONE “cutting.m”

Con la presente funzione i dati contenuti nei file di afflusso, invaso e rilascio (quelli con dominio, rispettivamente, .*MRO*, .*MRL* e .*MRC*), vengono depurati degli effetti di bordo (sostanzialmente, vengono eliminati dati relativi ad un certo numero di periodi all’inizio ed alla fine della simulazione, per ridurre il più possibile gli effetti dovuti alle condizioni al contorno).

```
function [cut_vectors] = cutting(mro_file, mrl_file, mrc_file, name_mro_cut, name_mrl_cut, name_mrc_cut)
num_per_begin = 12;
num_per_end = 12;
nro_res_per = 4;
nro_flow_per = 4;
nro_rel_per = 5;
i = 0; j = 0; k = 0;
%--------------------------------------------------------------------------
flow_mod = load('-ascii', mro_file);
res_mod = load('-ascii', mrl_file);
rel_mod = load('-ascii', mrc_file);
%--------------------------------------------------------------------------
flow_cut = flow_mod(((num_per_begin*nro_flow_per)+1):(end-(num_per_end*nro_flow_per)));
res_cut = res_mod(((num_per_begin*nro_res_per)+1):(end-(num_per_end*nro_res_per)));
rel_cut = rel_mod(((num_per_begin*nro_rel_per)+1):(end-(num_per_end*nro_rel_per)));
%--------------------------------------------------------------------------
fid_mro = fopen(name_mro_cut, 'wt');
fid_mrl = fopen(name_mrl_cut, 'wt');
fid_mrc = fopen(name_mrc_cut, 'wt');
for i=1:numel(flow_cut)
   fprintf(fid_mro, '%d\n', flow_cut(i));
end
for j=1:numel(res_cut)
   fprintf(fid_mrl, '%d\n', res_cut(j));
```

```
end
for k=1:numel(rel_cut)
   fprintf(fid_mrc, '%d\n', rel_cut(k));
end
fclose(fid_mro);
fclose(fid_mrl);
fclose(fid_mrc);
```

A2.11 FUNZIONE "sysdinNN.m"

La presente funzione è stata costruita per simulare il sistema con le reti neurali. Quella riportata di seguito fa riferimento ad architetture neurali addestrate per elaborare direttamente tutti e cinque i controlli per il sistema dello Zambesi; sono state create altre funzioni, simili per struttura a quella qui presentata, ma non riportate in questa sede, che eseguono le simulazioni per le altre tipologie di reti neurali descritte nel Capitolo 6 (ad esempio reti addestrate per singoli controlli).

```
function [s,r] = sysdinNN(weights_internal,biases_internal,weights_output,biases_output,Nneuron)
%INPUT PARAMETERS
Nt = 144; %Number of periods
Nres = 4; %Number of reservoirs
Nrel = 5; %Number of releases
%According to Aquafun Linear Optimizer and Standard conditions of Zambesi
%System, evaporation data are repeated every year. At the beginning it is
%built a [12,4] matrix; each row represents a Month (from October to
%September, and each column represents a reservoir (Kariba, Itezhi, Kafue,
%Cabora). This is replied for <Nt/12>, in order to obtain the complete
%evaporation matrix
evap = [40 277 833 308; 10 121 577 224; 0 24 0 0; 0 0 0 0; 0 0 0 0; 2 94 37 73; 23 169 453 178; 25 169 554 202;
19 129 495 182; 25 172 431 156; 29 201 554 204; 36 244 731 265];
e = repmat(evap, (Nt/12), 1);
%Vector of minimal and maximal values allowed for the reservoirs (both in
%order Kariba, Itezhi, Kafue, Cabora)
S_min = [50 780 140 4000];
S_MAX = [64800 5800 1040 68000];
%-------------------------------------------------------------------------
%Vectors and matrixes for normalized data
%Mean and Standard Deviation for reservoirs and releases (x_res and x_rel),
%in training data set. They should be the same also in simulation
Mean_res = [34843.241344, 3020.722583, 615.209823, 19067.973760];
Dev_res = [11943.732412, 1381.038104, 328.860249, 16005.554098];
Mean_rel = [3610.585156, 586.260646, 39.333333, 603.147000, 6403.426688];
Dev_rel = [269.136560, 317.517258, 1.105547, 144.234002, 1240.599525];
w_int = zeros(Nneuron,Nres);
b_int = zeros(Nneuron,1);
w_out = zeros(Nneuron,Nrel);
b_out = zeros(Nrel,1);
s = zeros(Nt+1,Nres);
r = zeros(Nt,Nrel);
rnorm = zeros(Nt,Nrel);
snorm = zeros(Nt+1,Nres);
s(1,:) = [43140 3045 860 36250];
s(Nt+1,:) = s(1,:); %Condition on final state
aff = load('-ascii', 'STD%.MRO');
rel = load('-ascii', 'STD%.MRC');
AQ_rels = zeros(Nt,Nrel);
for pp = 1:Nrel
   AQ_rels(:,pp) = rel(pp:Nrel:end);
end
a = zeros(Nt,Nres);
for qq = 1:Nres
   a(:,qq) = aff(qq:Nres:end);
end
a = [a; 0 0 0 0];
w_int = weights_internal;
```

```
b_int = biases_internal;
w_out = weights_output';
b_out = biases_output;
atot = zeros(Nt+1,Nres)+NaN;
atot(1,:) = a(1,:) + 0;
support = zeros(Nt+1,Nrel);
%Support variable to evaluate the performance of the network
counter = zeros(Nres,1);
counter_rel = zeros(Nrel,1);
for t = 1:Nt
   for zx = 1:Nres
      snorm(t,zx) = (s(t,zx) - Mean_res(zx))/Dev_res(zx);
   end
   for j = 1:Nrel
      for k = 1:Nneuron
         support(t,j) = support(t,j) + w_out(k,j)*(1/exp((-1)*(b_int(k)+ (snorm(t,:)*(w_int(k,:)'))))));
         %It is correct because in the reservoirs matrix the row t
         %refers to the previous period of time (whereas support at the
         %row t refers to the next period.
      end
   end
   for jhg = 1:Nrel
      rnorm(t,jhg) = b_out(jhg) + support(t,jhg);
   end
   for dd = 1:Nrel
      r(t,dd) = Mean_rel(dd) + (rnorm(t,dd)*Dev_rel(dd));
   end
   for ddd = 1:Nrel
      if r(t,ddd) < 0
         r(t,ddd) = 0;
      end
   end
   %Even if time indicator t can give misunderstandings in the reader, the
   %following formula is correct, because row t in the reservoirs matrix
   %refers to the previous period of time respect to releases, evaporation
   %and inflows matrixes.
   s(t+1,1) = s(t,1) - r(t,1) + atot(t,1) - e(t,1);
   s(t+1,2) = s(t,2) - r(t,2) + atot(t,2) - e(t,2);
   s(t+1,3) = s(t,3) - r(t,3) - r(t,4) + atot(t,3) - e(t,3);
   s(t+1,4) = s(t,4) - r(t,5) + atot(t,4) - e(t,4);
   if s(t+1,1) < S_min(1)
      s(t+1,1) = S_min(1);
      r(t,1) = s(t,1) + atot(t,1) - e(t,1) - s(t+1,1);
      if r(t,1) < 0
         r(t,1) = 0;
      end
      counter(1) = counter(1) + 1;
      counter_rel(1) = counter_rel(1) + 1;
   elseif s(t+1,1) > S_MAX(1)
      s(t+1,1) = S_MAX(1);
      r(t,1) = s(t,1) + atot(t,1) - e(t,1) - s(t+1,1);
      if r(t,1) < 0
         r(t,1) = 0;
      end
      counter(1) = counter(1) + 1;
      counter_rel(1) = counter_rel(1) + 1;
   end
   if s(t+1,2) < S_min(2)
      s(t+1,2) = S_min(2);
      r(t,2) = s(t,2) + atot(t,2) - e(t,2) - s(t+1,2);
      if r(t,2) < 0
         r(t,2) = 0;
      end
      counter(2) = counter(2) + 1;
      counter_rel(2) = counter_rel(2) + 1;
   elseif s(t+1,2) > S_MAX(2)
       s(t+1,2) = S_MAX(2);
       r(t,2) = s(t,2) + atot(t,2) - e(t,2) - s(t+1,2);
       if r(t,2) < 0
```

```
            r(t,2) = 0;
        end
        counter(2) = counter(2) + 1;
        counter_rel(2) = counter_rel(2) + 1
    end
    if s(t+1,3) < S_min(3)
        s(t+1,3) = S_min(3);
        r(t,3) = AQ_rels(t,3);
        r(t,4) = s(t,3) + atot(t,3) - e(t,3) - s(t+1,3) - r(t,3);
        if r(t,4) < 0
            r(t,4) = 0;
        end
        counter(3) = counter(3) + 1;
        counter_rel(3) = counter_rel(3) + 1;
        counter_rel(4) = counter_rel(4) + 1
    elseif s(t+1,3) > S_MAX(3)
        s(t+1,3) = S_MAX(3);
        r(t,3) = AQ_rels(t,3);
        r(t,4) = s(t,3) + atot(t,3) - e(t,3) - s(t+1,3) - r(t,3);
        if r(t,4) < 0
            r(t,4) = 0;
        end
        counter(3) = counter(3) + 1;
        counter_rel(3) = counter_rel(3) + 1;
        counter_rel(4) = counter_rel(4) + 1;
    end
    if s(t+1,4) < S_min(4)
        s(t+1,4) = S_min(4);
        r(t,5) = s(t,4) + atot(t,4) - e(t,4) - s(t+1,4);
        if r(t,5) < 0
            r(t,5) = 0;
        end
        counter(4) = counter(4) + 1;
        counter_rel(5) = counter_rel(5) + 1;
    elseif s(t+1,4) > S_MAX(4)
        s(t+1,4) = S_MAX(4);
        r(t,5) = s(t,4) + atot(t,4) - e(t,4) - s(t+1,4);
        if r(t,5) < 0
            r(t,5) = 0;
        end
        counter(4) = counter(4) + 1;
        counter_rel(5) = counter_rel(5) + 1;
    end
    atot(t+1,:) = a(t+1,:);
    atot(t+1,3) = atot(t+1,3) + r(t,2);
    atot(t+1,4) = atot(t+1,4) + r(t,1) + r(t,3);
end
r = [r; 0 0 0 0 0];
fid = fopen('Releases.txt', 'wt');
fid2 = fopen('Reservoirs.txt','wt');
fid3 = fopen('Performance.txt','wt');
for zz = 1:Nt+1
    fprintf(fid,'%f\n',r(zz,1));
    fprintf(fid2,'%f\n',s(zz,1));
    fprintf(fid,'%f\n',r(zz,2));
    fprintf(fid2,'%f\n',s(zz,2));
    fprintf(fid,'%f\n',r(zz,4));
    fprintf(fid2,'%f\n',s(zz,3));
    fprintf(fid,'%f\n',r(zz,5));
    fprintf(fid2,'%f\n',s(zz,4));
end
for qw = 1:Nres
    fprintf(fid3,'\tPerformance of reservoir %d in percent\n',qw);
    fprintf(fid3,'\t\t%5.3f\n\n\n',(1-(counter(qw)/Nt))*100);
end
fprintf(fid3,'\n\n');
for qw2 = 1:Nrel
    fprintf(fid3,'\tPerformance of release %d in percent\n',qw2);
    fprintf(fid3,'\t\t%5.3f\n\n\n',(1-(counter_rel(qw2)/Nt))*100);
```

```
end
fclose(fid);
fclose(fid2);
fclose(fid3);
```

A2.12 FUNZIONE “sysdinNN_time.m”

Questa funzione è stata utilizzata per simulare il sistema con le reti neurali, tenendo conto anche del tempo.

```
%INPUT PARAMETERS
Nt = 144; %Number of periods
Nres = 4; %Number of reservoirs
Ninfl = 4; %Number of inflows
Nrel = 5; %Number of releases
%According to Aquafun Linear Optimizer and Standard conditions of Zambesi
%System, evaporation data are repeated every year. At the beginning it is
%built a [12,4] matrix; each row represents a Month (from October to
%September, and each column represents a reservoir (Kariba, Itezhi, Kafue,
%Cabora). This is replied for <Nt/12>, in order to obtain the complete
%evaporation matrix
evap = [40 277 833 308; 10 121 577 224; 0 24 0 0; 0 0 0 0; 0 0 0 0; 2 94 37 73; 23 169 453 178; 25 169 554 202;
19 129 495 182; 25 172 431 156; 29 201 554 204; 36 244 731 265];
e = repmat(evap, (Nt/12), 1);
%Vector of minimal and maximal values allowed for the reservoirs (both in
%order Kariba, Itezhi, Kafue, Cabora)
S_min = [50 780 140 4000];
S_MAX = [64800 5800 1040 68000];
%--------------------------------------------------------------------------
%Vectors and matrixes for normalized data
%Mean and Standard Deviation for reservoirs, inflows and releases (x_res and x_rel),
%in training data set. They should be the same also in simulation
Mean_res = [33289.949125, 3073.316333, 614.296583, 17308.458042];
Mean_infl = [3963.361500, 655.687042, 203.758042, 2573.599667];
Dev_res = [9153.362219, 1287.728227, 342.860741, 14122.589054];
Dev_infl = [2347.894611, 541.854959, 168.936165, 1494.910313];
Mean_rel = [3513.911875, 617.377500, 39.333333, 648.609708, 6307.843333];
Dev_rel = [107.647629, 333.535129, 1.105565, 168.446979, 1005.051793];
w_int = zeros(Nneuron,Nres+Ninfl+2);
b_int = zeros(Nneuron,1);
w_out = zeros(Nneuron,Nrel);
b_out = zeros(Nrel,1);
s = zeros(Nt+1,Nres);
r = zeros(Nt,Nrel);
rnorm = zeros(Nt,Nrel);
snorm = zeros(Nt+1,Nres);
s(1,:) = [43140 3045 860 36250];
s(Nt+1,:) = s(1,:); %Condition on final state
aff = load('-ascii', 'STD%.MRO');
rel = load('-ascii', 'STD%.MRC');
AQ_rels = zeros(Nt,Nrel);
for pp = 1:Nrel
   AQ_rels(:,pp) = rel(pp:Nrel:end);
end
a = zeros(Nt,Nres);
for qq = 1:Nres
   a(:,qq) = aff(qq:Nres:end);
end
a = [a; 0 0 0 0];
w_int = weights_internal;
b_int = biases_internal;
w_out = weights_output';
b_out = biases_output;
atot = zeros(Nt+1,Nres)+NaN;
atot(1,:) = a(1,:) + 0;
support = zeros(Nt+1,Nrel);
```

```
time = 1:Nt;
T = Nt/12;
time_sin = sin(2*pi*time/T);
time_cos = cos(2*pi*time/T);
input_net = zeros(Nt+1,Nres+Ninfl+2);
%Support variable to evaluate the performance of the network
counter = zeros(Nres,1);
counter_rel = zeros(Nrel,1);
for t = 1:Nt
   for zx = 1:Nres
      snorm(t,zx) = (s(t,zx) - Mean_res(zx))/Dev_res(zx);
   end
   for fw = 1:Ninfl
      anorm(t,fw) = (atot(t,fw) - Mean_infl(fw))/Dev_infl(fw);
   end
   input_net(t,:) = [snorm(t,:), anorm(t,:), time_sin(t), time_cos(t)];
   for j = 1:Nrel
      for k = 1:Nneuron
         support(t,j) = support(t,j) + w_out(k,j)*(1/exp((-1)*(b_int(k)+ (input_net(t,:)*(w_int(k,:)'))))));
         %It is correct because in the reservoirs matrix the row t
         %refers to the previous period of time (whereas support at the
         %row t refers to the next period.
      end
   end
   for jhg = 1:Nrel
      rnorm(t,jhg) = b_out(jhg) + support(t,jhg);
   end
   for dd = 1:Nrel
      r(t,dd) = Mean_rel(dd) + (rnorm(t,dd)*Dev_rel(dd));
   end
   for ddd = 1:Nrel
      if r(t,ddd) < 0
         r(t,ddd) = 0;
         %counter_rel(ddd) = counter_rel(ddd) + 1;
      end
   end
   %Even if time indicator t can give misunderstandings in the reader, the
   %following formula is correct, because row t in the reservoirs matrix
   %refers to the previous period of time respect to releases, evaporation
   %and inflows matrixes.
   s(t+1,1) = s(t,1) - r(t,1) + atot(t,1) - e(t,1);
   s(t+1,2) = s(t,2) - r(t,2) + atot(t,2) - e(t,2);
   s(t+1,3) = s(t,3) - r(t,3) - r(t,4) + atot(t,3) - e(t,3);
   s(t+1,4) = s(t,4) - r(t,5) + atot(t,4) - e(t,4);
   if s(t+1,1) < S_min(1)
      s(t+1,1) = S_min(1);
      r(t,1) = s(t,1) + atot(t,1) - e(t,1) - s(t+1,1);
      if r(t,1) < 0
         r(t,1) = 0;
      end
      counter(1) = counter(1) + 1;
      counter_rel(1) = counter_rel(1) + 1;
   elseif s(t+1,1) > S_MAX(1)
      s(t+1,1) = S_MAX(1);
      r(t,1) = s(t,1) + atot(t,1) - e(t,1) - s(t+1,1);
      if r(t,1) < 0
         r(t,1) = 0;
%          error( sprintf(                    ...
%          'release %d at time %g is negative' , ...
%          1 , t                          ...
%         ))
      end
      counter(1) = counter(1) + 1;
      counter_rel(1) = counter_rel(1) + 1;
   end
   if s(t+1,2) < S_min(2)
      s(t+1,2) = S_min(2);
      r(t,2) = s(t,2) + atot(t,2) - e(t,2) - s(t+1,2);
      if r(t,2) < 0
```

```
        r(t,2) = 0;
      end
      counter(2) = counter(2) + 1;
      counter_rel(2) = counter_rel(2) + 1;
    elseif s(t+1,2) > S_MAX(2)
      s(t+1,2) = S_MAX(2);
      r(t,2) = s(t,2) + atot(t,2) - e(t,2) - s(t+1,2);
      if r(t,2) < 0
        r(t,2) = 0;
%       error( sprintf(                    ...
%         'release %d at time %g is negative' , ...
%         2 , t                          ...
%       ))
      end
      counter(2) = counter(2) + 1;
      counter_rel(2) = counter_rel(2) + 1
    end
    if s(t+1,3) < S_min(3)
      s(t+1,3) = S_min(3);
      r(t,3) = AQ_rels(t,3);
      r(t,4) = s(t,3) + atot(t,3) - e(t,3) - s(t+1,3) - r(t,3);
      if r(t,4) < 0
        r(t,4) = 0;
%       error( sprintf(                    ...
%         'release %d at time %g is negative' , ...
%         3 , t                          ...
%       ))
      end
      counter(3) = counter(3) + 1;
      counter_rel(3) = counter_rel(3) + 1;
      counter_rel(4) = counter_rel(4) + 1
    elseif s(t+1,3) > S_MAX(3)
      s(t+1,3) = S_MAX(3);
      r(t,3) = AQ_rels(t,3);
      r(t,4) = s(t,3) + atot(t,3) - e(t,3) - s(t+1,3) - r(t,3);
      if r(t,4) < 0
        r(t,4) = 0;
%       error( sprintf(                    ...
%         'release %d at time %g is negative' , ...
%         4 , t                          ...
%       ))
      end
      counter(3) = counter(3) + 1;
      counter_rel(3) = counter_rel(3) + 1;
      counter_rel(4) = counter_rel(4) + 1;
    end
    if s(t+1,4) < S_min(4)
      s(t+1,4) = S_min(4);
      r(t,5) = s(t,4) + atot(t,4) - e(t,4) - s(t+1,4);
      if r(t,5) < 0
        r(t,5) = 0;
%       error( sprintf(                    ...
%         'release %d at time %g is negative' , ...
%         5 , t                          ...
%       ))
      end
      counter(4) = counter(4) + 1;
      counter_rel(5) = counter_rel(5) + 1;
    elseif s(t+1,4) > S_MAX(4)
      s(t+1,4) = S_MAX(4);
      r(t,5) = s(t,4) + atot(t,4) - e(t,4) - s(t+1,4);
      if r(t,5) < 0
        r(t,5) = 0;
      end
      counter(4) = counter(4) + 1;
      counter_rel(5) = counter_rel(5) + 1;
    end
    atot(t+1,:) = a(t+1,:);
    atot(t+1,3) = atot(t+1,3) + r(t,2);
```

```
    atot(t+1,4) = atot(t+1,4) + r(t,1) + r(t,3);
end
r = [r; 0 0 0 0 0];
fid = fopen('Releases_time.txt', 'wt');
fid2 = fopen('Reservoirs_time.txt','wt');
fid3 = fopen('Performance_time.txt','wt');
for zz = 1:Nt+1
    fprintf(fid,'%f\n',r(zz,1));
    fprintf(fid2,'%f\n',s(zz,1));
    fprintf(fid,'%f\n',r(zz,2));
    fprintf(fid2,'%f\n',s(zz,2));
    fprintf(fid,'%f\n',r(zz,4));
    fprintf(fid2,'%f\n',s(zz,3));
    fprintf(fid,'%f\n',r(zz,5));
    fprintf(fid2,'%f\n',s(zz,4));
end
for qw = 1:Nres
    fprintf(fid3,'\tPerformance of reservoir %d in percent\n',qw);
    fprintf(fid3,'\t\t%5.3f\n\n\n',(1-(counter(qw)/Nt))*100);
end
fprintf(fid3,'\n\n');
for qw2 = 1:Nrel
    fprintf(fid3,'\tPerformance of release %d in percent\n',qw2);
    fprintf(fid3,'\t\t%5.3f\n\n\n',(1-(counter_rel(qw2)/Nt))*100);
end
fclose(fid);
fclose(fid2);
fclose(fid3);
```

A2.13 FUNZIONE "nt4_builder.m"

La presente funzione è utile per costruire una parte degli input ad Aquafun, in particolare il file *.NT4*, che varia per ognuna delle serie di afflusso. Senza l'ausilio di questa funzione, infatti, tale file si sarebbe dovuto ricostruire manualmente, con facilità notevole di incorrere in errori che ne avrebbero precluso la possibilità di lettura da parte dell'ottimizzatore lineare. Tale funzione deve essere eseguita in Octave anziché in Matlab.

```
function [txt] = nt4_builder( nt4_template , inflow , n_periods , n_reservoirs , n_arches )
where = sprintf(  '(in function %s)'  , mfilename );

usage_msg = sprintf( [                                  ...
   'Usage:  [txt] = nt4_builder( nt4_template , inflow , n_periods , n_reservoirs , n_arches )\n'  ] );
 if nargin < 5
   fprintf( 2,  'Error: not enough input arguments\n'  );
   fprintf( 2,  usage_msg  );
   error(  ' '  );
end
 check_is(                                          ...
   nt4_template , 'string' ,                          ...
   '%s the first argument <nt4_template> must be a string.' , ...
   where                                            ...
);
 fid = fopen( nt4_template , 'rt' );
check_is(                                           ...
   fid > 0 , 'true' ,                               ...
   [ '%s the file path passed as <nt4_template> cannot '      ...
     'be read' ] ,                                  ...
   where                                            ...
);
 txt = char( fread(fid)' );
fclose(fid);
 check_is(                                          ...
   inflow , 'natural' ,                             ...
   [ '%s the second argument <inflow> must be a vector '      ...
```

```
    'of naturals' ] ,                                ...
   where                                             ...
);
 check_is(                                           ...
   n_periods , 'scalar_numel' ,                      ...
   [ '%s the third argument <n_periods> must be a natural '   ...
     'scalar' ] ,                                    ...
   where                                             ...
);
 check_is(                                           ...
   n_reservoirs , 'scalar_numel' ,                   ...
   [ '%s the fourth argument <n_reservoirs> must be a natural'...
     'scalar' ] ,                                    ...
   where                                             ...
);
 check_is(                                           ...
   n_arches , 'scalar_numel' ,                       ...
   [ '%s the fifth argument <n_arches> must be a natural '    ...
     'scalar' ] ,                                    ...
   where                                             ...
);
 n_inflow = n_periods * n_reservoirs;
check_is(                                            ...
   numel(inflow) >= n_inflow , 'true' ,              ...
   [ '%s the second argument <inflow> is expected to have '   ...
   'at least %d elements' ] ,                        ...
   where ,                                           ...
   n_inflow                                          ...
);
 inflow     = reshape( inflow(1:n_inflow) , n_reservoirs , [] );
ret        = sprintf( '\n' );
endlid     = find( txt == ret );
n_txt_row  = numel(endlid);
 period_len =       ...
   1 +            ...  % header: "channels (period <i>)"
   n_reservoirs +  ...  % reservoir inflow
   n_arches;           % other physical system arches
 check_is(                                           ...
    n_txt_row == n_periods * period_len ,            ...
    'true' ,                                         ...
    [ '%s number of rows must be coherent with the '          ...
   'structure of the .NT4 file' ] ,                  ...
   where                                             );
 ctxt       = msplit( txt , ret );
lid        = logical( zeros(1,period_len) );
lid( 1 + [1:n_reservoirs] ) = 1;
id         = find( repmat( lid(:), n_periods , 1 ) );
for i=1:numel(id)
   r       = id(i);
   fprintf( 1 , 'line %5d  %-40s ' , r , ctxt{r} );
   if exist('OCTAVE_VERSION')
      fflush(1);
   end
   tmp     = mreg_replace(                 ...
      ctxt{r}                              , ...
      '^([^(]*)\(\d+'                      , ...
      [ '\1(' sprintf('%d' , inflow(i) ) ]  ...
   );
   ctxt{r} = tmp;
   fprintf( 1 , '| %-40s\n' , ctxt{r} );
end
 txt        = sprintf('%s\n', ctxt{:} );
```

A3

ALGORITMI DI OTTIMIZZAZIONE LINEARE

Nella presente Appendice, si descrivono gli algoritmi di ottimizzazione lineare più comuni, Relax, Netflo e Spath.
Il maggior dettaglio viene riservato all'algoritmo Relax, che è quello più noto, ampiamente validato ed utilizzato, anche per ottimizzare la rete del Fiume Zambesi, ed avere dei vettori di confronto utili per l'addestramento della Rete Neurale individuata.
Si riporta anche una breve descrizione di altre due procedure di ottimizzazione, perché utilizzate per analizzare la stessa Rete di serbatoi in passato, con lo scopo di confrontare le prestazioni dei medesimi algoritmi

A3.1 INTRODUZIONE ALL'OTTIMIZZAZIONE

Sebbene non essenziale alla comprensione del problema trattato nel presente lavoro, o al funzionamento di Aquafun, si ritiene comunque opportuno esporre brevemente come opera l'algoritmo di ottimizzazione Relax, al fine di avere un quadro chiaro e sintetico dell'intera procedura.
Scopo dell'ottimizzazione è calcolare l'utilità media risultante dall'applicazione di una determinata politica al sistema dei serbatoi preso in considerazione.
Detta utilità viene calcolata facendo ricorso ad una *funzione di utilità*, rispondente a precise caratteristiche di forma e struttura (come descritto nel Capitolo 1), che consentano di esprimerla come

$$\min_{\{f(p,q)\}} \sum -c(p,q) \times f(p,q) \qquad \text{(A3.1)}$$

dove il termine $-c(p,q)$ rappresenta la *funzione di costo* per ognuno degli archi che compongono la rete del sistema idrico, mentre il termine $f(p,q)$ rappresenta invece il flusso di risorsa che scorre nell'arco. Ogni arco è da intendersi come orientato dal nodo *p* al nodo *q*.
L'essenziale caratteristica delle funzioni di utilità di linearità a tratti consente di ricondursi, per l'ottimizzazione, ad un caso particolare della programmazione lineare, che viene risolta mediante l'*algoritmo del simplesso di Dantzing*.

A3.2 ALGORITMO RELAX

Questo algoritmo si basa sulla teoria della dualità. È un algoritmo piuttosto rapido nella ricerca della soluzione ottima, ed è quello maggiormente testato e validato.
Relax trova le sue basi nell'algoritmo cosiddetto "fuori – fase" (*Out – of – kilter algorihtm*). [Aashtiani, Magnanti, 1976]
Se si considera una rete generica composta da *N* nodi e *A* archi, ad ogni arco che congiunge due nodi (per esempio a_{ij} è l'arco che congiunge il nodo *i* al nodo *j*) viene associato un determinato numero che rappresenta il costo del passaggio di una unità di flusso (in questo caso l'acqua nei canali della rete, ma non necessariamente, data la generalità dell'impostazione) all'interno del medesimo arco. Indicando con c_{ij} il costo e con f_{ij} il flusso transitante nell'arco, l'algoritmo consente di trovare il minimo della funzione utilità costruita moltiplicando tra loro i due termini precedentemente introdotti, ossia

$$\min_{i,j} \sum_i \sum_j c_{ij} f_{ij} \qquad \text{(A3.2).}$$

Il sistema è sottoposto al seguente sistema di equazioni vincolari

$$\begin{cases} \sum_m f_{mi} - \sum_m f_{im} = 0, \forall i \in N \\ l_{ij} \leq f_{ij} \leq u_{ij} \end{cases} \qquad \text{(A3.3.a e A3.3.b),}$$

dove la prima equazione (A3.3.a) rappresenta la *conservazione del flusso* (si presti attenzione: nel primo termine il contatore *m* è indicato come primo pedice, quindi indica tutti i flussi entranti nel nodo generico *i*, mentre nel secondo termine dell'equazione il contatore compare come secondo pedice, ad indicare tutti i flussi che escono dallo specifico nodo *j*), mentre la seconda equazione (A3.3.b) impone il rispetto di una fascia di *capacità degli archi* (essendo l_{ij} la minima capacità dell'arco, e u_{ij} invece quella massima). Un altro vincolo "implicito", nel senso che non è espresso sotto forma di equazione, richiede che i dati in ingresso siano interi (è la condizione che soddisfa la prima relazione vincolare): qualora non lo siano, affinché l'algoritmo possa essere eseguito, è sufficiente modificare le unità di misura in modo opportuno.
Si assume, come ipotesi di partenza, che il sistema abbia una soluzione ammissibile.
Si introduce un termine, che si indica con p_i, che rappresenta il "prezzo ombra" (*shadow price*) necessario per avere la risorsa nello specifico nodo *i*, in modo che la prima equazione del sistema vincolare (A3.3.a) sia rispettata (un prezzo ombra deve essere definito per ogni arco che collega un nodo *i* ad un altro nodo *j*, dal momento che le equazioni vincolari devono essere definite su tutti gli archi). Per semplicità il prezzo ombra definisce il costo ridotto:

$$\overline{c_{ij}} = c_{ij} + p_j - p_i. \qquad \text{(A3.4)}$$

In generale, il termine c_{ij} rappresenta il costo di trasferimento, mentre il termine negativo p_i rappresenta il prezzo nel non disporre più la risorsa al nodo *i*, laddove il positivo p_j rappresenta quello nel disporre della stessa risorsa al nodo *j*.
L'insieme delle equazioni (A3.3) diviene quindi il sistema da risolvere, e la sua soluzione si trova calcolando la funzione lagrangiana associata, scrivibile come

$$L_{(fp)} = \sum_{(i,j)\in A}\left(\overline{c_{ij}}\right) \times f_{ij} \qquad \text{(A3.5),}$$

dove i termini *f* e $\bar{c}$ rappresentano i vettori contenenti rispettivamente i flussi tra due nodi *i* e *j* nel loro insieme di definizione *A* e i costi ridotti che possono assumere valori nell'insieme dei

numeri naturali. Tutti i costi sono per unità di flusso, quindi occorre moltiplicarli per il valore del flusso effettivo in transito.
Relax consente di risolvere il problema duale, esprimibile nella forma

$$\max_f q(p) \qquad \text{(A3.6)},$$

sul quale non vengono instaurati vincoli di alcun tipo. La funzione $q(p)$ può essere scritta in termini della sua complementare, ovverosia

$$(\blacksquare) \rightarrow \min_f L(f,p) \qquad \text{(A3.7)}.$$

In modo esteso si scrive

$$q(p) = \min_{l\leq f\leq u} \sum L(f,p) = \sum_{(i,j)\in A} \min_{l\leq f\leq u} \left(\left(c_{ij} + p_j - p_i\right) \times f_{ij} \right) = \sum_{(i,j)\in A} \left(q_{ij} \left(p_i - p_j\right. \right. \qquad \text{(A3.8)}$$

Per meglio chiarire il significato della differenza si definisce il *vettore tensione*, cioè

$$t_{ij} = p_i - p_j, \forall (i,j) \in A \qquad \text{(A3.9)}$$

che definisce la differenza di prezzi tra due nodi in cui avviene il trasferimento di flusso. Tale vettore concorre nel dare una determinazione ad un arco, tramite il confronto tra prezzi e costi, definiti dalla variabile *c* già introdotta precedentemente. Le determinazioni possibili per un arco sono (per semplicità sono stati omessi i pedici $_{ij}$, di cui bisogna sempre tenere conto:

1) *Inattivo*, se $t < c$;
2) *Bilanciato*, se $t = c$;
3) *Attivo*, se $t > c$.

Si definisce poi il *deficit del nodo i*, come

$$d = \sum_m f_{im} - \sum_m f_{mi}\,, \forall i \in N \qquad \text{(A3.10)}$$

Entrambi i problemi, sia quello primale sia quello duale hanno una soluzione che vada bene per entrambi, ossia (f,p), se e solo se sono verificate delle *condizioni di ottimalità*, riconoscibili in

$$\begin{cases} d = 0 \text{ per tutti i nodi } i & \text{CONDIZIONE AMMISSIBILE PRIMALE } (CPA) \\ \begin{cases} f_{ij} = L_{ij} \text{ se inattivo } \forall i,j \\ L_{i,j} < f_{i,j} < u_{i,j} \text{ se bilanciato } \forall i,j \\ f_{i,j} = u_{i,j} \text{ se attivo } \forall i,j \end{cases} & \text{CONDIZIONI DI SCARTO COMPLEMENTARE } (CSC) \end{cases} \qquad \text{(A3.11)}.$$

Il significato che può essere dato a queste condizioni è che se il costo per spostare la risorsa dal nodo *i* al nodo *j* risulta inferiore alla differenza tra i prezzi nei due nodi, allora conviene tenere la risorsa nel nodo in cui si trova (è il caso dell'arco inattivo), quindi il flusso che dovrà transitare nell'arco deve essere quello minimo possibile (non essendoci guadagno, non ha senso spostare risorsa). Un ragionamento analogo può essere implementato anche se l'arco è bilanciato o attivo.

Se sono soddisfatte queste condizioni, la funzione (f, p) è *soluzione ottima per entrambi i problemi*, sia quello primale sia quello duale (*min* e *max*), come introdotto in precedenza.
L'algoritmo procede ad aggiornare, ad ogni step il vettore dei prezzi, già indicato con *p*, mantenendo però i flussi in modo da soddisfare le condizioni di ottimalità *CSC*, e giunge a terminazione quando sono soddisfatte sia la condizione primale sia quelle complementari.
Vi sono molti algoritmi che si basano sul concetto di problema primale e duale, ma viene scelto Relax dal momento che individua la direzione di crescita del funzionale duale in modo molto semplice, eseguendo una transizione a coordinata singola.
Infatti, assegnato un qualsiasi vettore dei prezzi, si sceglie un nodo che abbia deficit non nullo e si verifica se è possibile un'ascesa del funzionale lungo la direzione che individua il nodo *i*. Se questo non è possibile, si prova a ridurre il deficit del nodo con un aumento del flusso. Quando nemmeno questa soluzione sortisca l'effetto desiderato, ossia di far crescere il funzionale, si cambia nodo, scegliendone uno, *i'*, adiacente al precedente, e si ripete la medesima procedura in modo iterativo, scegliendo nodi vicini finché non si ottiene il risultato desiderato, senza perdere in termini di efficienza metodologica.
La procedura di calcolo della direzione di ascesa è relativamente semplice, dal momento che ogni modifica al vettore tensione *t* produce un cambiamento a tutti e soli quei nodi che appartengono all'insieme *S* cui il vettore fa riferimento, e non mutano in alcun modo i flussi negli altri nodi.
A titolo esplicativo, per meglio chiarire come funziona la procedura di "ascesa", si definisce un sottoinsieme *S* dell'insieme *N* in cui possono assumere valori le variabili. Tale insieme prende la forma

$$v = \{v_{ij} | (i,j) \in A\} \qquad \text{(A3.12)},$$

dove i termini che lo compongono sono definiti secondo la seguente relazione:

$$v_{ij} = \begin{cases} 1 \text{ se } i \notin S \text{ e } j \in S \\ -1 \text{ se } i \in S \text{ e } j \notin S \\ 0 \text{ altrimenti} \end{cases} \qquad \text{(A3.13)}.$$

Ora è più chiaro il significato del precedente capoverso.
Precedentemente si è definito il problema duale *q(p)*, tramite le Equazioni (A3.6 – A3.8), cui è ovviamente associato un *costo duale*, la cui derivata *C(v,t)* rispetto al vettore tensione *t* lungo la direzione di crescita scelta (ossia *v*), rappresenta la diminuzione (sempre di una quantità uguale) dei prezzi in *S* senza toccare gli altri in alcun modo. Ciò è possibile se valgono le seguenti condizioni:

1) Il flusso degli archi inattivi è posto uguale al minimo possibile;
2) Il flusso degli archi attivi è posto uguale al massimo possibile;
3) Il flusso degli archi bilanciati assume:
    a) Il valore massimo se tali archi sono entranti in *S*;
    b) Il valore minimo se sono uscenti.

Con la precedente Equazione (A3.8) è stato costruito il funzionale *q(p)*, *q(p)*, da cui è possibile ricavare il costo duale, ottenibile mediante

$$w(t) = \sum_{i,j} q_{ij}(t_{ij}) \qquad \text{(A3.14)}$$

e la sua derivata direzionale nella direzione scelta

$$C(v,t) = \sum_{(i,j)\in A} \lim_{\alpha\to 0^+} \frac{q_{ij}(t_{ij}+\alpha v_{ij}) - q_{ij}(t_{ij})}{\alpha} = \sum_{(i,j)\in A} e_{ij}\left(v_{ij}, t_{ij}\right) \qquad \text{(A3.15).}$$

Avendo definito le due grandezze precedenti (mediante le Equazioni A3.14 e A3.15), si può scrivere

$$\begin{cases} w(t+\gamma v) = w(t) + \gamma C(v,t), \forall \gamma \in [0,\delta), \forall t, \forall S \subset N, S \neq \emptyset \\ \delta = min\{\{t_{im} - c_{im} | i \in S, m \notin S, (i,m)\text{: attivo}\}, \{c_{mi} - t_{mi} | i \in S, m \notin S, (m,i)\text{: inattivo}\}\} \end{cases}$$

(A3.16.a e A3.16.b),

proprio una relazione che descrive come la variazione del costo *w* al variare di *t* lungo *v* sia proprio la sua derivata, ossia *C(v,t)*. Finché tale relazione si mantiene lineare a tratti, la variazione del costo *w* lungo *v* si mantiene lineare partendo da γ fino a che γ diventa grande a sufficienza in modo tale che la coppia di punti $\{(w(t+\gamma v)); (t+\gamma v)\}$ incontri una nuova faccia del grafico del costo *w*.
Il valore di γ in questo punto è proprio quello che rende bilanciato un arco incidente in *S*. Si assume $\gamma = \delta$.
Usualmente, δ corrisponde alla prima discontinuità nella derivata della funzione di costo duale lungo la direzione di ascesa. È possibile anche scegliere un δ tale da massimizzare la funzione duale nella direzione prescelta, in modo da accelerare la convergenza. Quest'ultima soluzione è quella adottata dall'algoritmo Relax.
Il nome Relax deriva dal fatto che per mantenere valide le condizioni *CSC* (che risultano sempre verificate applicando questo metodo), l'algoritmo può aumentare liberamente il deficit negli altri nodi (in modo che sia soddisfatto un vincolo senza poterne però violare degli altri), effettuando quindi un "rilassamento" delle condizioni.
Di seguito si descrivono i passi che l'algoritmo segue per ricercare la soluzione.

Passo 0: inizializzazione

Si ha una coppia *(f, t)* che soddisfa le CSC all'inizio di ogni iterazione; dopo aver eseguito l'iterazione si ha una nuova coppia che soddisfa le stesse condizioni.

Passo 1: scelta del nodo candidato

Si scelga un nodo *s* con deficit positivo ($d_s > 0$, ma se si sceglie un nodo con deficit negativo, l'algoritmo rimane del tutto simile). In assenza di nodi che non soddisfano la condizione l'algoritmo termina dal momento che la soluzione trovata è quella ottima; in caso contrario si etichetti con il valore *0* il nodo corrente *s* e si assegni all'insieme *S* il valore di vuoto (in *S* sono contenuti i nodi analizzati, *scanned* in terminologia anglosassone); si vada al passo 2 ($\{s = 0, S = \emptyset\}$).

Passo 2: scelta della direzione di ascesa

Si scelga un nodo *k* già etichettato (*labeled*) ma non analizzato quindi $k \notin S$, si ponga il nodo scelto nell'insieme *S*, che diventa quindi $S = S \cup \{k\}$.

Passo 3: analisi del nodo *k*

Si analizzi il nodo scelto al passo precedente. Si assegni l'etichetta *k* a tutti i nodi *m* che non sono già etichettati e tali per cui l'arco *(m,k)* risulta bilanciato e il flusso entrante è minore del massimo possibile; si assegni la stessa etichetta a quei nodi non già etichettati e tali per cui il flusso uscente è maggiore del minimo. Si vada al passo 5 se *C(v,t)>0*, si vada al passo 4 se per ogni nodo etichettato si ha un deficit negativo; altrimenti si vada al passo 2.

Passo 4: aggiornamento del flusso

Al precedente passo 3 si è individuata una direzione (*P*) che consente di passare dal nodo *s* al nodo *m* con deficit calcolato. Semplicisticamente, questa fase "fornisce risorsa dal nodo in deficit togliendola da quello ad esso collegato". Computazionalmente, la procedura di costruzione di *P* consiste nel partire dal nodo di arrivo (*m*), individuando tutti gli archi bilanciati da *s* a *m* con $l_{k,n} < f_{k,n}$, e tutti gli archi bilanciati da *m* a *s* con $f_{k,n} > u_{k,n}$: si ricostruisce il percorso a ritroso. Viene calcolato il valore:

$$\epsilon = \min_{s,m,f,l,u}\{d_s, -d_m, (f_{k,n} - l_{k,n}) \leftarrow \text{da } s \text{ a } m, (u_{k,n} - f_{k,n}) \leftarrow \text{da } m \text{ a } s\} \quad \text{(A3.17)}$$

e si diminuisce di tale valore ϵ il flusso da tutti gli archi diretti da *m* a *s*, si aumenta dello stesso valore il flusso degli archi opposti. Si vada al passo 1.

Passo 5: aggiornamento del prezzo

Preso l'insieme *S* dei nodi definiti al passo 2, si calcola il valore di δ come dall'Equazione (A3.15.b) che lo definisce. Ricordando che *L* è l'insieme dei nodi etichettati, si definiscono due variabili $\begin{cases} i \in S \\ m \in L \end{cases}$. Si impone che tutti gli archi diretti da *i* a *m* prendano il flusso minimo e tutti quelli diretti in modo opposto prendano il flusso massimo. Il vettore tensione viene modificato come segue:

$$t_{im} = \begin{cases} t_{km} + \delta \text{ se } k \notin S, m \in S \\ t_{km} - \delta \text{ se } k \notin S, m \notin S \\ t_{km} \text{ altrimenti} \end{cases} \quad \text{(A3.18)},$$

con il significato che si aumentano di una stessa quantità i flussi che entrano in *S* e si diminuiscono della stessa quelli che ne escono. Si torni al passo 1.

Ogni iterazione termina dunque o con l'aumento del flusso allocato, oppure con l'aggiornamento del costo duale, a seconda del fatto che si applichino i passi 4 o 5. Inoltre, perché l'algoritmo risulti correttamente definito occorre che, tornando iterativamente dal passo 3 al passo 2, esista sempre un nodo già etichettato ma non ancora analizzato al fine di poter proseguire nella ricerca.

Tutti i dati in ingresso sono interi, conseguentemente le variabili intermedie e gli incrementi (o i decrementi) di flusso sono anch'essi interi (in particolare, è intero l'incremento al passo 5 e invece rimane invariato al passo 4). Il fatto che tutte le grandezze in gioco siano intere, garantisce la convergenza dell'algoritmo se la coppia iniziale $(f,)$ è formata da interi. Dopo un numero finito di iterazioni, infatti, anche la coppia finale (f', t') sarà formata da interi.

È stato dimostrato che Relax ha un peso computazionale non differente di quello di altri metodi primale – duale, tranne che per il calcolo della derivata del costo duale $C(v,t)$. Essa si può tuttavia ottenere facilmente in modo incrementale partendo dall'iterazione precedente:

$$C(v,t) = \sum_{k \in S} d_k \quad \text{(A3.19)}.$$

L'ampiezza δ calcolata con l'Equazione (A3.16.b) equivale alla prima delle discontinuità della derivata della funzione costo duale, che è lineare a tratti, lungo la direzione di ascesa. Relax implementa nella sua soluzione la scelta di un δ che massimizza la funzione duale lungo la direzione prescelta, in modo da accelerare la convergenza.

A3.3 ALTRE OTTIMIZZAZIONI LINEARI: CENNI

A3.3.1 ALGORITMO *NETFLO*

Questo algoritmo deriva dal metodo del simplesso primale, che non prevede il riaggiornamento della matrice di base mediante la quale si esprime il sistema. Come ipotesi di partenza considera quella che non possano esserci archi paralleli, e nella matrice ogni arco è rappresentato da una colonna.
Detto *G* il sistema di interesse, si definisce con *T* un qualsiasi sottografo di *G*, che abbia almeno due nodi. In questo caso la matrice *A* che rappresenta il grafo contiene colonne tutte linearmente indipendenti, ma *A* è di rango n – *1*. Affinché sia possibile risolvere il problema di programmazione lineare, la matrice va resa di rango massimo, aggiungendo quindi delle colonne linearmente dipendenti tra loro, a capacità nulla (ciò equivale ad aggiungere degli archi non caricati). L'algoritmo NETFLO provvede a triangolizzare la matrice, con lo scopo di risolvere il sistema.
La condizione iniziale è una politica sub ottima che va comunque trovata ed inserita. Il programma provvede a trovare flussi e costi ottimi, utilizzando appunto il simplesso.

A3.3.2 ALGORITMO *SPATH*

Questo ultimo algoritmo si basa invece sui *cammini minimi*, che consistono sostanzialmente nell'individuare il percorso a minimo costo (temporale, economico o di qualsiasi altro genere) tra una sorgente ed un pozzo attraverso l'algoritmo di Bellman. Il massimo flusso ammissibile viene allocato proprio sul cammino a costo minimo. Al raggiungimento dell'allocamento l'algoritmo termina con successo, e viene aggiornata l'utilità.
Al fine di poter utilizzare il metodo, occorre definire l'arco inverso *(q,p)*, che viene creato all'atto della prima assegnazione all'arco diretto e viene aggiornato ad ogni successiva modifica del flusso all'interno di detto arco. Dal punto di vista pratico, per poter definire il flusso all'arco diretto (flusso denominato f_{pq}, come già ampiamente detto in precedenza), occorre imporre all'arco inverso una capacità pari a tale flusso, e un costo opposto a quello dell'arco diretto, in modo che l'arco inverso possa, ad una generica successiva distribuzione, recuperare tale flusso e distribuirlo qualora ne possa trarre un vantaggio. In simboli si ha:

$$\begin{cases} \begin{cases} u_{qp} = f_{pq} \\ l_{qp} = 0 \end{cases} \text{condizioni sulla capacità} \\ c_{qp} = -c_{pq} \text{condizione sul costo} \end{cases} \qquad \text{(A3.20).}$$

Di seguito si riportano i passi che definiscono l'algoritmo.

Inizio

Passo 0: inizializzazione ➔ si inseriscono gli archi inversi, tutti con capacità nulla e si inizializzano le variabili *flusso = 0* e *utilità = 0*.
Passo 1: determinazione del cammino minimo ➔ si determina il cammino minimo, chiamato *r*, definito da *O* a *D*. Se esso non esiste, non c'è soluzione ammissibile.
Passo 2: valutazione del flusso assegnabile ➔ si determina la massima quantità allocabile di flusso lungo il cammino *(p,q)*, applicando

$$f = \min \left\{ \min \left\{ c_{p,q} | \forall (p,q) \in r \right\}, F - flusso \right\} \qquad \text{(A3.21),}$$

e si vada al passo 3.

Passo 3: aggiornamento dei vincoli ➔ per ognuno degli archi del cammino occorre aggiornare il valore del flusso allocato, la capacità residua e quella dell'arco inverso:

$$\begin{cases} f_{p,q} = f_{p,q} + f \\ u_{p,q} = u_{p,q} - f \quad \forall (p,q) \in r \\ u_{q,p} = u_{q,p} + f \end{cases} \qquad \text{(A3.22).}$$

Passo 4: aggiornamento del flusso ➔ il valore del flusso complessivamente allocato viene aumentato della quantità determinata al passo 2, applicando semplicemente

$$flusso = flusso + f \qquad \text{(A3.23).}$$

Se si denota con *F* la richiesta di flusso alla sorgente, se *flusso* < *F* si torna al passo 1, altrimenti è possibile determinare soluzione ottima e l'utilità complessiva assume il valore

$$u = \sum_{(p,q)\in A} f_{p,q} \times c_{p,q} \qquad \text{(A3.24).}$$

Termine

È da notare che (Passo 4) l'utilità non viene allocata ad ogni iterazione, ma solamente al termine, quando l'algoritmo ha allocato tutto il flusso

Dei tre algoritmi testati (con un PC Hp Vectra 486 a 25 MHz [Taddio, Togni, 1993], oggi assolutamente vetusto ed in alcun modo applicabile ai test pratici), il migliore risulta sempre essere Relax, mentre Spath viene eliminato quasi subito (i test sono stati eseguiti aumentando il numero di cicli, fino al raggiungimento della massima dimensione accettabile dal sistema, aumentando il numero degli archi aumentando il numero di serbatoi e aumentando la dimensione della funzione utilità.
Spath, nonostante la sua inefficacia rilevata rapidamente in tutti i test, non viene escluso a priori, ma viene utilizzato solamente in fase di validazione dei modelli, mentre Netflo e Relax anche in taratura.

BIBLIOGRAFIA

AA. VV., software “Octave Forge”, ultima visita 30 giugno 2010 (http://octave.sourceforge.net/)

Aashtiani H. A., Magnanti T. L., June 1976, “Implementing primal – dual network flow algorithms”, *Operation Research Center*, MIT

Baglietto M., Parisini T., Zoppoli R., May 2001, “Distribuited – Information Neural Control: the case of Dynamic Routing in Traffic Networks”, *IEEE Transaction on Neural Network*, Vol. 12, No. 3

Bellman R., 1957, “The theory of Dynamic Programming”, Princeton University Press

Bertsekas D. P., Hosein P., Tseng, P., 1989, “Relaxation methods for network flow problem with convex arc cost”, *Control and Optimization 25*

Bertsekas D. P., Tseng P., 1994, “Relax IV: A faster version of the RELAX code for solving minimum cost flow problems”, *LIDS Reports*, P – 2276, M.. I. T.

Bertsekas D. P., Tseng P., 1985, “Relaxation methods for minimum cost – ordinary and generalized network flow problems”, *LIDS Reports*, LIDS – P – 1462, M. I. T.

Bertsekas D. P., Tseng P., 1988, “The Relax codes for linear minimum flow network flow problems”, *Annals of Operations Research 13*, pp. 125 – 190

Bertsekas D.P., Tseng P., software “Relax”, ultima visita 30 giugno 2010 (http://www.di.unipi.it/optimize/Software/MCF.html#RelaxIV)

Bertsekas D. P., Tsitsiklis J. N., 1996, “Neuro – Dynamic programming”, Athena Scientific, Belmont, Massachusetts

Bolzern P., Scattolini R., Schiavoni N., 2008, “Fondamenti di controlli automatici”, McGraw-Hill

Bryson A. E., Ho Y., Taylor, Francis, 1975, “Applied optimal control: optimization, estimation, and control”, New York

Burattini E., Cordeschi R., “Intelligenza Artificiale”, Carocci Editore, Roma, *consultato nel 2010*

Busacker R. G., Saaty T. L., 1965, “Finite Graphs and Networks – An introduction with applications”, Mc Graw Hill

Cammarata S., 1990, “Reti Neuronali. Una introduzione all’altra intelligenza artificiale”, ETAS Libri, Milano

Clerico R., Fabbri P., Ortenzio F., Giugno 2009, “Charing Cross Road”, *Rubrica “Rudi Matematici”*, Le Scienze numero 490, pp. 106 – 107

Collins M., July 2002, "Discriminative training methods for hidden Markov models: Theory and experiments with the perceptron algorithm", *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pp. 1 – 8, Philadelphia

Colorni A., 1984, "Ricerca operativa", Edizioni CLUP, Milano

Cormen T. H., Leiserson C. E., Rivest R. L., 1990, "Introduction to algorithms", MIT Press and McGraw-Hill

Cybenko G. V., 1989, "Approximation by Superpositions of a Sigmoidal Function", *Mathematics of Control, Signals and Systems*, pp. 203 – 2014, Springler – Verlag, New York

De Rigo D., software "Mastrave Project", ultima visita 30 giugno 2010
(http://www.mastrave.org/)

Demuth H., Beadle M., Hagan M., 1992 e successive aggiornamenti, "Neural Network Toolbox – User's Guide", The Matworks Inc.
(www.mathworks.com)

Di Franco G., 1998, "Reti Neurali artificiali e analisi dei dati per la ricerca sociale: un nuovo paradigma?", *Sociologia e Ricerca sociale*, numero 56, pagine 35 – 75

Dung Doan C., Liong S., 2004, "Generalization for Multilayer Neural Network Bayesian regularization or Early Stopping", Department of Civil Engineering, National University of Singapore, Singapore

Eaton J. W., 2002, "GNU Octave Manual", Network Theory Limited

Eaton J. W. *et al.*, software GNU Octave, ultima visita 30 giugno 2010
(http://www.gnu.org/software/octave/)

Fabbri G., Orsini R., 1993, "Reti Neurali per le scienze economiche", Franco Muzzio Editore

Floreano D., Mattiussi C., 2002, "Manuale sulle Reti Neurali", Il Mulino, Bologna

Free Software Foundation, Software "GNU Project", ultima visita 30 giugno 2010
(http://www.gnu.org/)

Freund Y., Schapire R. E., 1999, "Large margin classification using the Perceptron algorithm", *Machine Learning*, No. 37, pp. 277 – 296, Kluver Academic Publisehers, The Netherlands

Funahashi K., 1989, "On the Approximate Realization of continuous mapping by Neural Networks", *Neural Networks*, Vol. 2, pp. 183 – 192, Pergamon Press

Gallant S. I., June 1990, "Perceptron-based learning algorithms", *IEEE Transactions on Neural Network*, Vol. 1, No. 2, pp. 179 – 191, June 1990

Gallo C., 2007, "Reti Neurali artificiali: Teoria ed Applicazioni", *Quaderno n. 28/2007*, Dipartimento di Scienze Economiche, Matematiche e Statistiche, Università degli Studi di Foggia

Gandolfi C., Guariso G. e Togni D., 1997, “Optimal Flow allocation in the Zambesi River System”, *Water Resource Management 11*, pp. 377 – 393, Kluver Academic Publishers, Netherlands

Giuliano G., Cancelliere A., Rossi G., “Regole di esercizio di sistemi di serbatoi basate su reti neurali e sistemi inferenziali fuzzy”, Dipartimento di Ingegneria Civile ed Ambientale, Università di Catania, *consultato nel 2010*

Gomory R. E., Hu T. C., June 1962, “An application of generalized linear programming to network flows”, *Journal of the Society for Industrial and Applied Mathematics*, Vol. 10, No. 2, pages 260 – 283

Greppi M., 2005, “Idrologia”, Seconda edizione, Editore Ulrico Hoepli Milano

Grossman W., Guariso G., Hitz M., Werthner H., January 1995, “A Min Cost flow solution for Dynamic Assignment Problems in Network with storage devices”, *Management Science*, Vol. 41, No. 1, pp. 83 – 93, published by www.jstor.org

Hillier F. S., Lieberman G. J., 2001, “Introduction to operations research”, McGraw – Hill, 7th edition

Karul C., Soyupak S., Çilesiz A. F., Akbay N., Germen E., 2000, “Case studies on the use of Neural Networks in eutrophication modeling”, *Ecological Modeling 134*, pp. 145 – 152, Elsevier

Lang K. J., Waibel A. W., Hinton G. E., 1990, “A time – delay Neural Network architecture for isolated word recognition”, *Neural Networks*, Vol. 3, pp. 23 – 43, Pergamon Press

Levenberg K., 1944“A Method for the Solution of Certain Non-Linear Problems in Least Squares”, *The Quarterly of Applied Mathematics*, pp. 164 – 168

Macchiavello C., 1992, “Introduzione alle Reti Neurali – *seminario tenuto il 17 dicembre 1992*”, Dipartimento di Fisica A. Volta, Università degli Studi di Pavia

Maier H. R., Dandy G. C., 2000, “Neural Network for the prediction and forecasting of water resources variables: a review of modelling issues and applications”, *Environmental Modelling and Software*, nr. 15, pp. 101 – 124

Marquardt D. W., June 1963, “An algorithm for Least-Squares estimation of nonlinear parameters”, *Journal of the Society for Industrial and Applied Mathematics*, Vol. 11, No. 2, pp. 431 – 441, published by www.jstor.org

Meraviglia C., 2001, “Le Reti neurali nella ricerca sociale”, Il Mulino, Bologna

Ogata K., 2002, “Modern Control Engineering”, Pearson Education International

Onida M., Tesi di Laurea, Anno Accademico 2001 – 2002, “Un approccio basato su reti neurali per la gestione ottimale di sistemi idrici”, Relatore Prof. Giorgio Guariso, Correlatore Ing. Giorgio Corani, Dipartimento di Elettronica ed Informazione, Politecnico di Milano

Patarnello S., 1991, “Le Reti neuronali”, Franco Angeli, Milano

Pessa E., 2004, "Statistica con le Reti Neurali", Di Rienzo Editore, Roma

Rosenblatt F., 1958, "The perceptron: a probabilistic model for information storage and organization in the brain", *Psychological Review*, Vol. 65, No. 6, pp. 386 – 408, Cornell Areonautical Laboratory

Solomatine D. P., Avila Torres L. A., September 1996, "Neural Network Approximation of a Hydrodynamic Model in optimizing Reservoirs operation", presented in *International Conference on Hydroinformatics*, Zurich

Soncini Sessa R., 2004, DVD – ROM allegato a "Il Progetto Verbano", Mc Graw – Hill

Soncini Sessa R., 2004, "Il Progetto Verbano – Modellistica integrata e decisione partecipata in pratica, *Pianificazione e gestione delle risorse idriche*", Mc Graw – Hill, collana Ambiente e Territorio

Soncini Sessa R., , 2004, "MODSS – Per decisioni integrate e pianificate, *Pianificazione e gestione delle risorse idriche*", Mc Graw – Hill, collana Ambiente e Territorio

Soncini Sessa R., Anni Accademici 2006 – 2007 e 2007 – 2008, Lucidi dei Corsi di Gestione delle Risorse Naturali 1 con Laboratorio e Gestione delle Risorse Naturali 2, DEI, Politecnico di Milano

Soncini Sessa R., Weber E., Castelletti A., 2007, "Appendix A8 – Neural Netwoks", from "Integrated and Partecipatory Water Resources Management – *Theory*", Elsevier LTD.

Southern Africa Development Coordination Conference – SADCC, 3.0.4 – 240 – 3

Stallman R. M., 2010, "GNU Emacs Manual", version 23.2, 16th edition, Free Software Foundation Press, Boston

Stam A., Salewicz K. A., Aronson J. E., 1998, "An interactive reservoir management system for Lake Kariba", *European Journal of Operational Research 107*, pp. 119 – 136, Elsevier

Taddio M., Togni D., Tesi di Laurea, Anno Accademico 1992 – 1993, "Il problema di assegnamento ottimo del flusso nelle reti: applicazioni al caso delle reti idriche", Relatore Prof. Giorgio Guariso, Correlatore Ing. Claudio Gandolfi, Dipartimento di Elettronica ed Informazione, Politecnico di Milano

Taddio M., Togni D., 1993, "Manuale d'uso del pacchetto Aquafun", Appendice alla Tesi di Laurea, Politecnico di Milano, 1993

Taddio M., Togni D., 1993, software "Aquafun"

Taha H. A., 2007, "Operations research: an introduction", Pearson Prentice Hall, 8th edition, New Jersey

The Matworks, Inc., software "Matlab", ultima visita 30 giugno 2010 (http://www.mathworks.com)

Vapnyarskii I. B., Hazewinkel M., 2001, "Lagrange multipliers", *Encyclopaedia of Mathematics*, Kluwer Academic Publishers

Widrow B., Lehr M. A., September 1990, "30 years of Adaptive Neural Networks: Perceptron, Madaline, and Backpropagation", *Proceedings of the IEEE*, Vol. 78, No. 9, pp. 1415 – 1442

Pao Y., 1989, “Adaptive pattern recognition and neural networks”, Addison Wesley

Dizionario online inglese – italiano (www.wordreference.com)

Per numerose foto, sito Panoramio (www.panoramio.com)

Matlab Help offline e online
(www.mathworks.com/support e www.mathworks.com/access/helpdesk)

Wikipedia per i dati anagrafici italiani (http://it.wikipedia.org/wiki/Italia)

Wikipedia per le informazioni geografiche sullo Zambesi
(http://it.wikipedia.org/wiki/Zambesi, http://en.wikipedia.org/wiki/Zambezi)

Per i dati sulle Cascate Vittoria, sito World Waterfall Database
(http://www.world-waterfalls.com/waterfall.php?num=147)

Sito web dell’Autorità di Bacino del Fiume Zambesi (*Zambesi Water Authority*)
(http://www.zaraho.org.zm/bmd.html)