UNIVERSITÀ
POLITECNICA
DELLE MARCHE

Facoltà di Ingegneria
Corso di Laurea in Ingegneria Meccanica

Valutazione dell'utilizzo di algoritmi genetici per identificare le proprietà meccaniche di acciai con metodi inversi

Evaluation on the use of genetic algorithms to identify the steels' mechanical properties by inverse strategies

Candidato:
Alessio Bonfitto

Relatore:
Prof. Marco Rossi

Correlatore:
Dott. Ric. Attilio Lattanzi

Anno Accademico 2019-2020

UNIVERSITÀ
POLITECNICA
DELLE MARCHE

Facoltà di Ingegneria
Corso di Laurea in Ingegneria Meccanica

Valutazione dell'utilizzo di algoritmi genetici per identificare le proprietà meccaniche di acciai con metodi inversi

Evaluation on the use of genetic algorithms to identify the steels' mechanical properties by inverse strategies

Candidato:
Alessio Bonfitto

Relatore:
Prof. Marco Rossi

Correlatore:
Dott. Ric. Attilio Lattanzi

Anno Accademico 2019-2020

Università Politecnica delle Marche
Facoltà di Ingegneria
Corso di Laurea in Ingegneria Meccanica
Via Brecce Bianche – 60131 Ancona (AN), Italy

Ringraziamenti

Ancona, Ottobre 2020

Alessio Bonfitto

Abstract

Evolutionary algorithms are inspired by Darwin's theory of evolution of the species and are based on the genetic inheritance and natural selection concept. The aim is to arrive at an optimal solution to a given problem, through the creation of new generations of individuals/solutions.

Genetic algorithms constitute the simplest variant, as the representation of individuals occurs through a binary string of fixed length. They start from a randomly generated initial population of solutions and then develop new generations through mutation and crossover genetic operators.

The fields of application of this class of algorithms are varied, in particular the attention will be placed on the evaluation of their use in an optimization problem concerning the identification of the constituent parameters of the material, which belong to a certain reference model.

Finally, experimental tests will be carried out with genetic algorithms, varying the optimization options, to characterize the parameters of *Yld2000-2d* model.

Sommario

Gli algoritmi evolutivi si ispirano alla teoria dell'evoluzione della specie di Darwin e si basano sui concetti dell'ereditarietà genetica e della selezione naturale. Lo scopo è quello di arrivare ad una soluzione ottima ad un dato problema, attraverso la creazione di nuove generazioni di individui/soluzioni.

Gli algoritmi genetici ne costituiscono la variante più semplice, poichè la rappresentazione degli individui avviene mediante una stringa binaria di lunghezza fissa. Essi partono da una popolazione iniziale di soluzioni generata casualmente, per poi sviluppare nuove generazioni attraverso gli operatori genetici di *crossover* e *mutazione*.

I campi di applicazione di questa classe di algoritmi sono svariati, in particolare l'attenzione verrà posta sulla valutazione del loro utilizzo in un problema di ottimizzazione riguardante l'identificazione dei parametri costitutivi del materiale, che appartengono ad un certo modello di riferimento.

Infine verranno svolte delle prove sperimentali con algoritmi genetici, variando le opzioni di ottimizzazione, per caratterizzare i parametri del modello *Yld2000-2D*.

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Introduzione

Gli Algoritmi Evolutivi sono una vasta classe di algoritmi stocastici di ricerca per problemi di ottimizzazione. Si basano sull'imitazione dei processi biologici che permettono a popolazioni di organismi di adattarsi all'ambiente in cui vivono, quali l'ereditarietà genetica e la sopravvivenza del più adatto. Tali concetti sono stati introdotti nel secolo scorso da Charles Darwin e ancora oggi sono ampiamente riconosciuti come validi.

Essenzialmente gli Algoritmi Evolutivi fanno uso di una metafora in cui il problema oggetto prende il posto dell'ambiente; le soluzioni ammissibili sono viste come individui che vivono in questo ambiente; il grado di adattamento di un individuo all'ambiente è infine il corrispettivo della funzione obiettivo calcolata su una soluzione ammissibile [1].

Nella metafora, insiemi di soluzioni ammissibili prendono il posto delle popolazioni di organismi. Le soluzioni differiscono tra loro per qualità, cioè per costo o merito, che si riflettono nella valutazione della funzione obiettivo, così come gli individui di una popolazione di organismi differiscono tra di loro per grado di adattamento all'ambiente. La selezione naturale, che permette a una popolazione di adattarsi all'ambiente che la circonda, sarà in grado, applicata a una popolazione di soluzioni a un problema, di far evolvere soluzioni sempre migliori ed eventualmente, con il tempo, ottime.

In base a questa metafora, il modello computazionale prende in prestito dalla biologia alcuni concetti e i relativi termini: ogni soluzione è codificata in uno o più *cromosomi*; i *geni* sono i pezzi della codifica responsabili di uno o più tratti di una soluzione; gli *alleli* sono le possibili configurazioni che un gene può assumere.

Gli algoritmi genetici fanno parte di quelli evolutivi e ne costituiscono la variante più semplice e diffusa.

Infatti questi rapprensentano l'indivuo/soluzione attraverso una stringa di bit in linguaggio binario, familiare alla maggior parte dei calcolatori.

L'algoritmo procede in modo iterativo, mantenendo una popolazione di individui che rappresentano possibili soluzioni per il problema che deve essere risolto e facendola evolvere attraverso l'applicazione di operatori stocastici, che si ispirano alla genetica.

La popolazione iniziale, da cui parte il processo di ottimizzazione, può provenire da un campionamento casuale dello spazio delle soluzioni oppure da un nucleo di

soluzioni iniziali, trovate da semplici procedure di ricerca locale.

Un esempio di problema di ottimizzazione è il caso dell'identificazione dei parametri costitutivi del materiale, che verrà illustrato nei capitoli seguenti, in cui l'algoritmo lavora sui parametri per minimizzare una funzione di costo.

Esistono diversi tipi di algoritmi, tra cui quelli *derivate free*, quelli *a gradiente*, quelli *trust region* e gli *algoritmi genetici*.

Ogni metodo è caratterizzato da punti di forza e di debolezza, in particolare quelli *a gradiente* utilizzano le informazioni riguardanti le derivate di primo e secondo ordine. Essi sono ottimi ricercatori di minimi in uno spazio di ricerca ristretto e arrivano a convergenza generalmente in poco tempo e con un numero non elevatissimo di iterazioni. Il punto di debolezza di questi algoritmi è la loro forte dipendenza dai parametri iniziali, essendo di natura deterministica.

Gli *algoritmi genetici*, invece, sono di natura stocastica, quindi la loro dipendenza dai parametri di input è nettamente più bassa rispetto agli altri, inoltre sono ottimi ricercatori di minimi in uno spazio ampio e hanno una bassa probabilità di convergere verso quelli locali. Lo svantaggio è che necessitano di un alto tempo di calcolo e di un numero molto elevato di iterazioni. Inoltre nell'implementazione di questi algoritmi è richiesto all'utente di selezionare le impostazioni che desidera, come la grandezza della popolazione iniziale, il numero di generazioni da creare, il tasso di crossover e il metodo secondo cui verranno selezionati gli inidividui genitori in ogni generazione. L'utilizzo scorretto di alcune di queste opzioni, in alcune circostanze, può mettere a dura prova la stabilità dell'algoritmo e può portarlo a convergenza verso minimi locali, andando a compromettere la prestazione dell'intero processo.

Nel Cap.7 sono illustrate delle prove sperimentali, e i risultati ottenuti, riguardanti l'implementazione degli algoritmi genetici nell'identificazione dei parametri costitutivi del materiale. Sono state svolte sei diverse ottimizzazioni, nelle quali sono state variate le impostazioni di ogni algoritmo, in modo da avere delle indicazioni generali sul suo modo di operare.

Le ottimizzazioni sono state svolte all'interno del software *Matlab*, che offre una serie di metodi per gestire il funzionamento dell'algoritmo e per crearne delle varianti.

Lo scopo di queste prove è quello di valutare l'utilizzo degli algoritmi genetici in questo specifico problema di ottimizzazione, considerando il risultato numerico, il tempo di calcolo, anche le opzioni necessarie da inserire per ottenere un algoritmo performante e anche la validità fisica dei risultati ottenuti.

Quest'ultima è valutata confrontando le curve del *Normalized flow stress* e del *R-value*, ottenute con i parametri ottimizzati, con quelle di riferimento, ottenute quindi a partire da un set di riferimento, come illustrato nel Cap.7.

Capitolo 2

Algoritmi evolutivi

2.1 Cenni storici

L'idea di usare selezione e mutazione casuale per un compito di ottimizzazione risale agli anni cinquanta, con il lavoro dello statistico George E. P. Box [2], che tuttavia non fece uso dell'elaboratore elettronico. Contemporaneamente diversi studiosi, concepirono l'idea di simulare l'evoluzione su un elaboratore elettronico: Barricelli e Fraser utilizzarono simulazioni al calcolatore per studiare i meccanismi dell'evoluzione naturale, il biomatematico Hans J. Bremermann [3] fu il primo a riconoscere nell'evoluzione biologica un processo di ottimizzazione.

Gli *algoritmi evolutivi*, nelle loro varianti originarie, furono inventati allo stesso tempo, a metà degli anni Sessanta, in tre distinti gruppi di ricerca: in America, Lawrence Fogel e colleghi, dell'Università di California a San Diego, posero le basi della *programmazione evolutiva* [4], mentre presso l'Università del Michigan John Holland proponeva i primi *algoritmi genetici* [5]; in Europa Ingo Rechenberg e colleghi, studenti presso il Politecnico di Berlino, idearono quelle che battezzarono *strategie evolutive* (Evolutionsstrategien) [6].

I tre filoni si svilupparono separatamente fino alla prima edizione del congresso PPSN (Parallel Problem Solving from Nature), tenutasi a Dortmund nel 1990, dopo il quale venne messo in atto uno sforzo organizzativo per farli convergere e si formò un'unica comunità scientifica, comprendente tutti i ricercatori interessati al calcolo evoluzionistico.

2.2 I tre filoni

2.2.1 Algoritmi genetici

Gli *algoritmi genetici* sono una delle versioni più semplici di quelli evolutivi, poiché le soluzioni sono rappresentate come stringhe binarie di lunghezza fissa. Questo tipo di rappresentazione è sicuramente la più generale, tuttavia non sempre è anche la più conveniente: infatti, qualsiasi struttura dati, per quanto complessa e articolata, sarà sempre codificata in alfabeto binario nella memoria dell'elaboratore elettronico.

2.2.2 Strategie evolutive

I primi sforzi verso la formulazione del concetto di *Strategia Evolutiva* ebbero luogo nel 1964 presso il Politecnico di Berlino, in Germania, da parte di Ingo Rechenberg e Hans-Paul Schwefel. Le applicazioni erano principalmente sperimentali e avevano a che fare con problemi di idrodinamica come l'ottimizzazione di tubi piegati e di ugelli per motori a getto.

Le strategie evolutive affrontano l'ottimizzazione di una funzione obiettivo reale di variabili reali in uno spazio a l dimensioni. La rappresentazione utilizzata per le variabili indipendenti della funzione è un vettore di numeri reali.

Oltre a codificare le variabili indipendenti, tuttavia, le *strategie evolutive* includono nell'individuo anche delle informazioni sulla distribuzione di probabilità da utilizzare per la loro perturbazione.

2.2.3 Programmazione evolutiva

All'inizio degli anni sessanta Lawrence J. Fogel, studente all'Università di California a Los Angeles, pensò di sfruttare un'evoluzione simulata per fare emergere dell'intelligenza artificiale in una popolazione di algoritmi in competizione tra di loro.

Quest'idea fu esaminata in una serie di studi e, qualche anno più tardi, Fogel e colleghi descrissero una tecnica per l'evoluzione di automi a stati finiti per compiti di predizione, che chiamarono *Programmazione Evolutiva*. Ogni automa di una popolazione genera per mutazione un discendente per la generazione successiva, quindi la metà migliore dei genitori e dei figli sopravvive andando a formare una nuova popolazione.

2.3 Il problema della rappresentazione e la programmazione genetica

Gli *algoritmi evolutivi* manipolano una versione codificata delle soluzioni di un problema e lo schema di codifica può limitare in modo significativo l'efficacia dell'algoritmo. Gli *algoritmi genetici* di Holland, i quali lavorano su sequenze di cifre binarie di lunghezza fissa, pur permettendo di sviluppare una serie di risultati teorici, si adattano con difficoltà a certi tipi di problemi, come quelli di ordinamento.

Per questi motivi John Koza introdusse la *programmazione Genetica* [7], un'estensione degli algoritmi genetici dove ogni elemento della popolazione rappresenta un programma per computer anziché una stringa di bit.

Questi programmi si evolvono combinandosi, riproducendosi o mutando, per dar luogo ad altri programmi che si adattano meglio a risolvere un determinato problema.

Capitolo 3

Algoritmi genetici

3.1 Introduzione

Gli algoritmi genetici sono strumenti utili per risolvere problemi di ricerca e di ottimizzazione che seguono un procedimento ispirato alla genetica e alla teoria della selezione naturale di Charles Darwin.

L'insieme delle soluzioni evolve in più generazioni, al fine di ricercare la soluzione ottima. Ad ogni soluzione viene associato un valore chiamato *fitness* che costituisce una stima della bontà della soluzione. Mediante questo valore vengono confrontati gli individui della generazione attuale e, selezionati quelli migliori, questi vengono usati per creare la popolazione della generazione successiva.

Con questo procedimento si cerca di ricavare un insieme di individui sempre migliore che, con l'avanzare delle generazioni, contenga la soluzione ottima del problema o una sua approssimazione. Per creare le nuove soluzioni si utilizzano gli stessi operatori di mutazione e crossover presenti nella genetica, per questo motivo ci si riferisce agli individui anche con il nome di cromosomi.

L'algoritmo ha un funzionamento ciclico che simula il processo evolutivo di una popolazione. Ogni ciclo rappresenta una generazione e al suo interno vengono svolte le operazioni per generare una nuova popolazione formata da individui sempre migliori.

All'inizio viene creata la popolazione iniziale in maniera casuale, le fasi successive, che si ripetono ad ogni ciclo, consistono nel selezionare gli individui migliori e generarne di nuovi mediante gli operatori genetici.

Inoltre, si definisce un criterio di terminazione, che stabilisce quando l'algoritmo si deve fermare.

Alcuni esempi di criteri di terminazione possono essere:

- il passaggio di un numero prefissato di generazioni o di una certa quantità di tempo;
- il rinvenimento di una soluzione soddisfacente secondo qualche misura;
- la mancanza di miglioramenti per un certo numero prefissato di generazioni.

3.1.1 Fitness

La *fitness* di un individuo è un valore che si calcola con una funzione e indica quanto il cromosoma e la sua codifica sia una buona soluzione del problema. Tale funzione permette il confronto tra gli individui e quindi l'ordinamento della popolazione.

È importante che, nelle prime generazioni, la popolazione sia composta da individui con caratteristiche eterogenee, in modo che gli operatori di ricombinazione possano esplorare in modo esauriente lo spazio delle soluzioni ed evitare di convergere verso minimi locali.

3.1.2 Gestione della popolazione

Esistono diverse varianti di algoritmo genetico, dipendenti dalla gestione della popolazione.

Se l'intera popolazione è sostituita interamente dai nuovi elementi si parla di *generazionale*, in caso contrario si ha l'algoritmo genetico *steady-state* (a stato fissato). Si ha un *modello elitarista* se l'elemento o gli elementi migliori sono conservati durante l'evoluzione della popolazione. Nel *modello a isole*, invece, si hanno una serie di popolazioni che evolvono in maniera autonoma, con occasionali migrazioni di individui da un'isola all'altra.

Ogni individuo della popolazione è rappresentato da una stringa di lunghezza prefissata di bit. La scelta di utilizzare prevalentemente un alfabeto binario è dovuta ad alcuni importanti risultati teorici, come il *teorema degli schemi* di Holland, che indica in questa come una rappresentazione molto vicina all'ottimo.

3.2 Operatori genetici

Negli algoritmi genetici si utilizzano principalmente tre operatori: la *riproduzione*, la *mutazione* e il *crossover*. I primi due metodi si applicano ad un solo individuo, mentre il crossover ne coinvolge due. Prima di applicare uno di questi operatori è necessario selezionare uno o due individui della popolazione, a seconda del caso.

I principali metodi di selezione sono illustrati in seguito. In generale, però, tutti i metodi hanno lo scopo di far sì che gli individui con fitness migliore abbiano maggiori probabilità di essere selezionati e, quindi, di trasmettere i propri geni alla generazione successiva, rispettando i meccanismi evolutivi di sopravvivenza dei più adatti

3.2.1 Riproduzione

La riproduzione ricopia l'individuo nella nuova popolazione, lasciando intatto tutto il suo patrimonio genetico. Questo operatore agisce con probabilità molto bassa e non è alla base della convergenza dell'algoritmo.

3.2.2 Ricombinazione

Per applicare il *crossover*, gli individui genitori vengono accoppiati a due a due, quindi, con una certa probabilità *pcross*, chiamata "tasso di crossover", che è un parametro dell'algoritmo, ciascuna coppia subisce la ricombinazione vera e propria.

Il crossover combina i patrimoni genetici dei due genitori in modo da costruire due "figli", che possiedono metà dei geni di uno e metà dell'altro.

Esistono diverse tipologie di crossover. Quello *ad un punto*, Fig.3.1, sceglie in modo casuale una posizione della stringa e genera i figli nel modo seguente: supponendo che la stringa sia di lunghezza L, si sceglie K estraendolo a caso nell'intervallo 1...L, quindi il primo figlio avrà i geni da 1...K del primo genitore e da K+1 dell'altro e il secondo l'inverso.

Altre tipologie largamente usate di crossover sono quello *a due punti*, in cui le stringhe sono scambiate, con procedimento analogo al precedente, fra due punti scelti a caso e quello *uniforme*, in cui ogni bit è scambiato con posizioni selezionate a caso.

Il crossover è la forza trainante dell'algoritmo genetico ed è l'operatore che maggiormente influenza la convergenza. Infatti, ogni nuova popolazione viene costruita con gli individui migliori della precedente per propagare nelle generazioni il patrimonio genetico migliore, per cui è chiaro che, se i figli non fossero simili ai genitori, non avremmo alcuna garanzia sulla convergenza.

Nonostante ciò, un suo utilizzo troppo spregiudicato potrebbe portare ad una convergenza prematura della ricerca su qualche minimo locale.

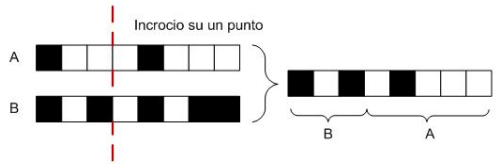


Figura 3.1: Funzionamento cross-over ad un punto.

3.2.3 Mutazione

La *mutazione* introduce dei cambiamenti nel patrimonio genetico per generare nuovi cromosomi, quindi nuove soluzioni. In particolare, vengono modificate una o più cifre binarie della stringa che rappresenta la soluzione. Un'esempio di mutazione in una stringa binaria è in Fig.3.2.

L'obiettivo di questo operatore è quello di esplorare zone dello spazio delle soluzioni non ancora esplorate ed evitare la convergenza verso ottimi locali.

Infatti l'operatore di mutazione può generare nuovi individui molto diversi dai precedenti e, di conseguenza, assicurare che vengano esplorate nuove regioni dello spazio di ricerca.

Tuttavia, un eccessivo utilizzo della mutazione rende instabile l'algoritmo e mette a rischio la sua convergenza. Per questi motivi questo operatore genetico è considerato

di secondo piano, applicato con probabilità molto basse, dell'ordine di una cifra binaria modificata su mille copiate [8].

Un processo evolutivo di mutazione si può rivelare molto utile, infatti assicura che il coefficiente di mutazione cambi con la riduzione della funzione di costo e che la probabilità di mutazione assuma un valore accettabile.

Figura 3.2: Esempio di mutazione binaria.

3.3 Altri operatori di base genetica

3.3.1 Selezione

Una volta calcolato il valore di *fitness* di tutta la popolazione vengono selezionati gli individui che andranno a finire in quella chiamata *mating pool* e andranno a generare nuove soluzioni. Il concetto di base è che un individuo migliore deve avere una probabilità di sopravvivere e di subire ricombinazioni maggiore di quella di un individuo peggiore.

Esistono varie tecniche di selezione. La più semplice consiste nell'ordinare gli n individui della popolazione in base al valore della funzione di fitness e scegliere le prime k soluzioni (con $k < n$). Se l'algoritmo genetico deve risolvere un problema di minimo è sufficiente selezionare i primi k elementi, altrimenti gli ultimi, se è un problema di massimo. Un esempio di questo metodo di selezione è in Fig 3.3.

Un'altra tecnica, chiamata *fitness-proportionate selection*, prevede di selezionare gli individui in base ad una probabilità proporzionale alla fitness. Prima di tutto viene calcolata la fitness totale F della popolazione P(t);

$$Fmax(x) = \sum_{k=1}^{n} f(x_i) \tag{3.1}$$

$$Fmin(x) = \sum_{k=1}^{n} \frac{1}{f(x_i)} \tag{3.2}$$

dove $f(x_i)$ rappresenta la fitness dell'individuo x_i, e i due casi rappresentano i problemi di minimo e di massimo.

È possibile ricavare la probabilità con cui le soluzioni saranno selezionate tramite queste formule;

$$p_i = \frac{1}{f(x_i)} \frac{1}{Fmin} \tag{3.3}$$

$$p_i = \frac{f(x_i)}{Fmax} \tag{3.4}$$

Il risultato di queste due espressioni può essere rappresentato con una ruota suddivisa in spicchi, Fig.3.4 e Fig.3.5, la cui grandezza è proporzionale alla probabilità associata alle soluzioni. La ruota viene fatta girare per k volte e vengono estratte le soluzioni corrispondenti.

I software usati per le ottimizzazioni usano diversi criteri di selezione, ad esempio il software *Matlab* mette a disposizione i seguenti:

- *selectionstochunif*, imposta una linea in cui ogni genitore ne occupa una porzione proporzionale al proprio valore di fitness e l'algoritmo avanza a passi costanti, selezionando gli individui che creeranno la nuova generazione;
- *selectionremainder*, conta gli individui come genitori un numero di volte pari al valore intero della propria fitenss, mentre i restanti valori decimali si usano come proporzione per la selezione dei genitori restanti;
- *selectionroulette*, simula la ruota di una roulette, cui le porzioni di questa sono assegnate agli individui proporzionalmente alla loro fitness, mentre il lancio della pallina è simulato da un numero causuale che seleziona una delle porzioni;
- *selectiontournament*, simula una competizione tra un numero, in genere uguale a 4, di individui scelti casualmente, tra i quali emerge quello con valore di fitness più alto.

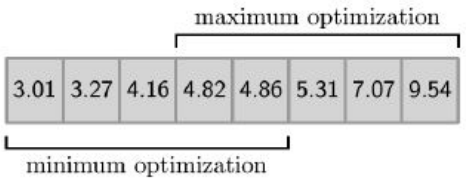


Figura 3.3: Selezione lineare.

3.3.2 Rimpiazzamento

L'operazione di *rimpiazzamento* serve per determinare la popolazione della generazione successiva, scegliendo quali e quanti individui devono farne parte, stabilendo il numero di cromosomi figli da generare e scegliendo i cromosomi padre da rimpiazzare.

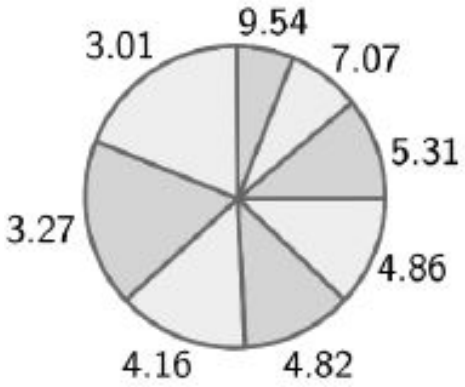


Figura 3.4: Selezione proporzionale in un problema di minimo.

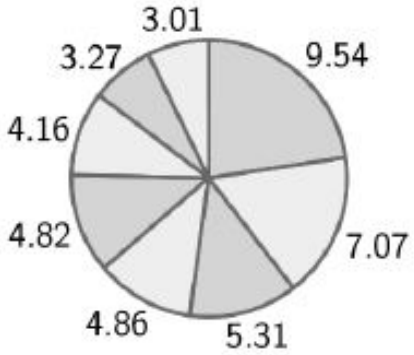


Figura 3.5: Selezione proporzionale in un problema di massimo.

Ipotizzando che la popolazione sia composta da n individui, il rimpiazzamento consiste nel generare r nuove soluzioni tramite cross-over o mutazione, con r minore di n, e nel selezionare le r-n soluzioni rimanenti tra gli individui della generazione precedente.

La scelta del parametro r è molto importante, poiché influenza la velocità con cui l'algoritmo andrà a convergenza. Infatti con valori molto piccoli si ha un basso ricambio generazionale, il che porta a rallentare la ricerca della soluzione ottima.

Tuttavia, è sconsigliato scegliere valori troppo alti di r, poiché mantenere delle soluzioni della popolazione precedente produce dei vantaggi, soprattutto in termini di stabilità.

In Fig.3.6 è illustrato lo schema generale di funzionamento dell'algoritmo genetico, con tutti gli step che si susseguono durante il processo di ottimizzazione.

Figura 3.6: Schema generale di ottimizzzione mediante algoritmo genetico.

3.4 Parallelizzazione

Gli algoritmi genetici hanno bisogno di grandi popolazioni per ottenere una buona convergenza, questo aspetto porta ad uno sforzo computazionale notevole.

Se, ad esempio consideriamo una popolazione M e facciamo girare il nostro algoritmo per un numero di generazioni G, allora il tempo totale di valutazione sarà dell'ordine di GM per il tempo di valutazione di una singola *fitenss*, il che può essere

inaccettabile anche su workstation molto potenti. Da qui la necessità di realizzare un'implementazione parallela di questo tipo di algoritmi. Inoltre questi ultimi si prestano molto bene a questo tipo di utilizzo, infatti spesso si parla di *parallelismo implicito* degli algoritmi genetici.

I principali modelli per l'implementazione di algoritmi paralleli evolutivi sono il modello *globale*, ad *isole* e *diffusivo*.

3.4.1 Il modello globale

In questo modello viene mantenuta una singola popolazione, ma la valutazione della *fitness* degli individui è eseguita in parallelo, dividendo il carico fra i processori disponibili, mentre le fasi di *cross-over* e *mutazione* sono eseguite in sequenziale.

La selezione è quella classica che agisce su tutta la popolazione, da qui il nome *globale* per il modello.

Questo tipo di modello ha il principale vantaggio di preservare il comportamento dell'algoritmo sequenziale anche in parallelo.

3.4.2 Il modello a isole

Il modello a isole rappresenta una valida alternativa alla precedente soluzione. Questo modello divide la popolazione complessiva in sottopopolazioni e le associa ai processori, i quali lavorano in maniera indipendente e isolata. L'esecuzione di tutti gli operatori e la valutazione della *fitness* viene eseguita a livello locale, in tal modo l'algoritmo presenta una scalabilità quasi lineare e risulta di facile implementazione.

Inoltre possono verificarsi fenomeni di migrazione da una sottopopolazione all'altra, così da introdurre delle diversità nelle varie isole e garantire una migliore convergenza. Nonostante ciò, questi fenomeni non sono, in genere, molto frequenti e coinvolgono pochi elementi per generazione.

3.4.3 Il modello diffusivo

Nel modello diffusivo, la popolazione è distribuita spazialmente, su una griglia bidimensionale. Ogni individuo k è sostituito nella successiva generazione da un nuovo elemento generato applicando l'operatore di crossover o mutazione ad elementi appartenenti al vicinato di k stesso, Fig.3.7.

La selezione ristretta ad un vicinato locale rende l'implementazione parallela molto efficiente, dal momento che le comunicazioni sono limitate solo ai vicini.

Questo modello si presta ad una naturale ed efficiente implementazione parallela, inoltre ottiene una convergenza migliore, grazie alla selezione locale che facilita lo sviluppo di nicchie che esplorano spazi di ricerca diversi.

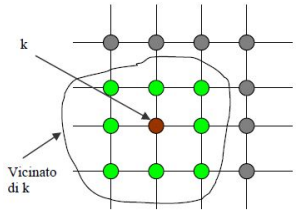


Figura 3.7: Selezione locale.

3.5 Algoritmi genetici multiobiettivo

3.5.1 Generalità

L'ottimizzazione *multiobiettivo* è un'estensione di quella a singolo obiettivo, infatti molti metodi utilizzano una trasformazione per convertire il problema in una forma equivalente a singolo obiettivo, con il vantaggio di poter utilizzare i numerosi strumenti già presenti per questo tipo di ottimizzazione.

I problemi di ottimizzazione *multiobiettivo* sono caratterizzati da più funzioni che devono essere ottimizzate simultaneamente.

In questi problemi, spesso, gli obiettivi sono in contrasto tra di loro e l'ottimizzazione di uno produce risultati insoddisfacenti per gli altri. Un tipico esempio di questa situazione è la volontà di produrre un oggetto minimizzandone i costi e massimizzandone le prestazioni, tale problema non ha una soluzione univoca, ma un insieme di soluzioni, ciascuna delle quali soddisfa al meglio gli obiettivi.

Un altro esempio riguarda il caso in cui si hanno i punti A e B, in cui si vuole minimizzare la distanza e il costo del tragitto. Ipotizzando che il valore della funzione di costo sia legato al percorso e non alla distanza, poiché, ad esempio, il tragitto più breve non è il più facile da percorrere e quello meno costoso non è il più corto, non esiste alcun valore per il quale entrambe le espressioni assumano valore minimo e le due funzioni sono in contrasto perché, migliorando il valore di una, si peggiora il risultato dell'altra. Consideriamo $d(\phi)$ funzione obiettivo della distanza e $c(\phi)$ funzione obiettivo del costo definite dalle Eq.3.5 e 3.6:

$$d(\phi) = \phi^2 + 1 \tag{3.5}$$

$$c(\phi) = (\phi - 2)^2 - 1 \tag{3.6}$$

Quest'esempio mostra chiaramente il concetto di conflitto tra due funzioni, Fig. 3.8, per cui non è possibile individuare un'unica soluzione ottima, visto che abbiamo due criteri di scelta tra loro contrastanti e quindi non è possibile capire se una soluzione sia meglio dell'altra.

Per risolvere questo problema bisogna considerare valido un intero insieme di soluzioni per il quale non siamo in grado di definire una preferenza. Questo insieme sarà formato da soluzioni ritenute tutte valide come soluzioni del sistema.

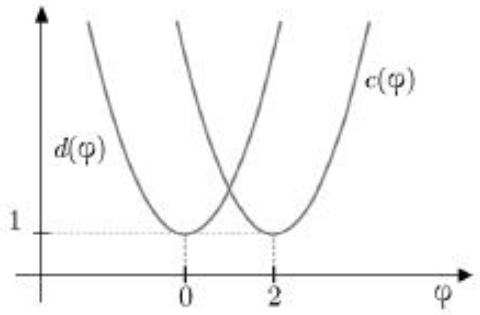


Figura 3.8: Esempio di funzioni obiettivo legate al costo e alla distanza.

La cardinalità delle soluzioni è la principale differenza tra i problemi multiobiettivo e i problemi a singolo obiettivo dove la soluzione, se esiste, è unica. Nell'esempio del tragitto le soluzioni ottime appartengono all'intervallo [0,2] perché sono un buon compromesso tra la distanza e il costo.

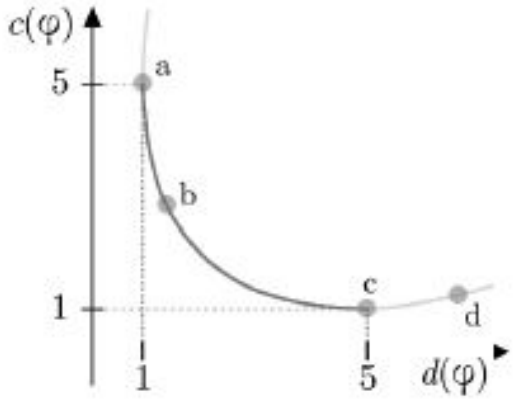


Figura 3.9: Soluzioni ottime.

La Fig 3.9 mostra la composizione dell'insieme delle soluzioni dal punto di vista delle funzioni obiettivo. Il punto a di coordinate (1,5) è la soluzione ottima per la distanza, mentre il punto c è la soluzione ottima del costo. Anche la soluzione intermedia b (1.49, 2.69) è ancora valida rispetto alle altre due. Infatti se consideriamo il tragitto associato al punto a si nota che ha un costo maggiore rispetto a b, ma allo stesso tempo ha una distanza minore. Non potendo scegliere tra il costo e la distanza queste due soluzioni sono entrambe valide.

Queste considerazioni sono le stesse che permettono di scegliere i punti corretti e ricavare l'insieme delle soluzioni. Per determinare questo esistono delle tecniche ben precise, tuttavia possiamo fare una prima analisi delle soluzioni mediante la Fig.3.9. Pertanto consideriamo il punto d di coordinate (6.76,1.16) e notiamo che entrambi gli obiettivi sono maggiori del punto c. In questo caso si può dire che il tragitto identificato dal punto d è peggiore del tragitto identificato da c e per questo motivo non viene inserito nell'insieme delle soluzioni.

Questo insieme viene formato da i soli punti per i quali non è possibile trovare soluzioni che siano migliori, quindi non esistono soluzioni che siano migliori delle soluzioni contenute nell'insieme ottimo.

3.5.2 La risoluzione

La non esistenza di una soluzione univoca richiede all'utente di fare una scelta, che può esser fatta in base a parametri più soggettivi, che sono estranei alle funzioni obiettivo.

Esistono due tipi di informazioni che permettono la risoluzione del problema. Le prime sono generali e si possono quantificare in una funzione obiettivo, mentre le seconde chiamate informazioni ad alto livello, sono soggettive e specifiche di un determinato settore di utilizzo.

Questi due tipi di informazioni sono usate entrambe, ma l'ordine di utilizzo delle stesse definisce due modi per trovare le soluzioni.

La prima tecnica di ricerca si basa sull'analisi diretta delle funzioni obiettivo con metodi di ottimizzazione multiobiettivo. Grazie a questi viene trovato l'insieme delle soluzioni ottime e viene estratta la soluzione desiderata con le informazioni ad alto livello possedute dall'utente, Fig.3.10. Le informazioni generali e meno soggettive vengono utilizzate nella prima fase per definire le funzioni obiettivo.

Figura 3.10: Risoluzione con la prima strategia.

La seconda tecnica utilizza le informazioni ad alto livello per assegnare delle preferenze degli obiettivi all'inizio del procedimento. Queste preferenze vengono quantificate in pesi in modo che si possa trasformare il problema in uno equivalente a singolo obiettivo.

Figura 3.11: Risoluzione con la seconda strategia.

Questa strategia ha degli evidenti limiti. I principali sono la necessità di assegnare dei valori alle informazioni di secondo livello, che sono qualitative, e quella di rieseguire completamente tale strategia nel caso in cui cambi il contesto di utilizzo e le preferenze siano differenti.

Nella prima strategia, invece, si crea l'insieme di tutte le soluzioni ottime e poi, in base alle informazioni dell'utente, viene scelta la soluzione corretta. Per cui se le

preferenze sugli obiettivi dovessero cambiare non occorre ripetere di nuovo i calcoli, ma è sufficiente scegliere una nuova soluzione.

3.5.3 Ottimo paretiano

L'*ottimo paretiano* o *efficienza paretiana* è un concetto introdotto dall'ingegnere ed economista italiano Vilfredo Pareto, applicato in economia, teoria dei giochi, ingegneria e scienze sociali. Si realizza quando l'allocazione delle risorse è tale che non è possibile apportare miglioramenti paretiani al sistema, cioè non si può migliorare la condizione di un soggetto senza peggiorare la condizione di un altro.

Nel caso delle ottimizzazioni *multiobiettivo* si è visto come spesso non si arriva ad una soluzione univoca, ma piuttosto ad un insieme di punti ottimali.

Si consideri un problema di minimizzazione di due obiettivi f_1 e f_2, che hanno minimi distinti m_1 e m_2. I punti compresi tra m_1 e m_2 sono i punti definiti *ottimi di Pareto*, ovvero quei punti che sono il miglior compromesso tra i due obiettivi che hanno minimi diversi.

Rappresentando le funzioni in un piano che ha in ascissa e in ordinata i valori delle funzioni obiettivo f_1 e f_2, rispettivamente, si ottiene il fronte di Pareto, Fig.3.12. Quest'ultimo rappresenta l'insieme delle soluzioni che sono il miglior compromesso tra gli obiettivi in conflitto.

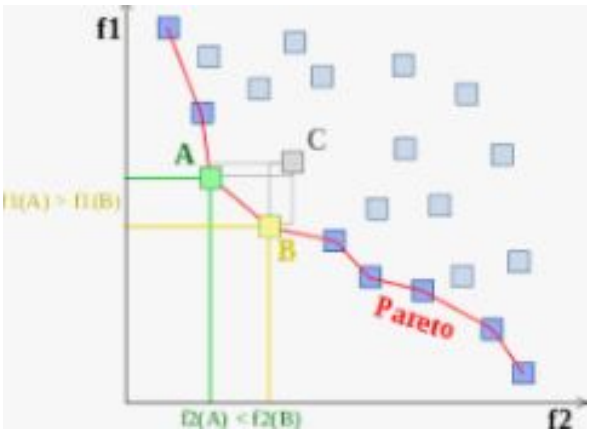


Figura 3.12: Fronte di Pareto.

In generale il *fronte di Pareto* è un insieme di soluzioni ottime, ovvero è costituito da tutti i punti non dominati, cioè da quei punti per i quali non ne esiste nessun altro che sia migliore contemporaneamente per tutti gli obiettivi considerati nella funzione di ottimizzazione.

Quindi un algoritmo genetico è in grado di calcolare le soluzioni di un problema, le quali sono contenute nell'insieme ottimo di Pareto, che in genere ha dimensione infinita. Tuttavia, un algoritmo genetico non è in grado di rappresentare tutte le soluzioni ottime di Pareto perché ha a disposizione solo un numero finito di individui. Da ciò deriva che il compito di un algoritmo genetico non è solamente cercare alcune

soluzioni ottime, ma fare in modo che queste rappresentino al meglio l'insieme ottimo di Pareto.

Questo strumento viene spesso usato nelle soluzioni ingegneristiche, dove un progettista deve valutare un insieme di diverse soluzioni ottime e dà un'indicazione su come comportarsi per raggiungere il compromesso desiderato.

Capitolo 4

Impiego degli algoritmi genetici: la strategia inversa

4.1 Introduzione

Al giorno d'oggi la comunità scientifica e quella industruiale si trovano spesso nella situazione di confrontarsi con complessi problemi di ingegneria meccanica, come la formatura di lamiere. Questi non possono essere risolti con metodi per tentativi, comunemente usati in passato, soprattutto per ragioni economiche.

In questo contesto l' *Analisi agli Elementi Finiti* (FEM) ha assunto un ruolo importante, poichè permette test virtuali. Inoltre aiuta a progettare processi e prodotti di qualità e con alte prestazioni strutturali.

Tuttavia la *FEM*, per ottenere accurati campi di tensione e deformazione, ha bisogno di dati di input certi, come la geometria, i casi di carico, la legge di attrito e quella non lineare che regola il comportamento del materiale.

Questo approccio è definito *diretto*, in quanto la qualità dei risultati dipende dalla quantità dei dati in ingresso, che non sono sempre disponibili, e dal modello costitutivo utilizzato.

4.2 Il metodo inverso

Un possibile approccio alternativo sono i problemi inversi, nei quali sono utilizzati parametri di input, basati su dati sperimentali, da inserire all'interno di modelli costitutivi e geometrici, al fine di implementarli in simulazioni numeriche.

Esempi di metodi inversi sono la FEMU, *Finite Element Model Updating*, e il VFM, *Virtual Fields Method*, i quali verrano illustrati in seguito.

Possono essere formulate diverse categorie di problemi inversi, tra le quali ci sono quelli che riguardano l'identificazione di parametri del materiale, all'interno di un modello costitutivo.

Nel tempo, lo sviluppo di nuovi materiali e gli sforzi impiegati per caratterizzare quelli esistenti, hanno portato a modelli costitutivi molto complessi, i quali richiedono l'identificazione di una grande quantità di parametri, adatti a descrivere il comportamento del materiale che si vuole analizzare.

Sono stati sviluppati, ad esempio, molti modelli costitutivi capaci di spiegare e predirre i dettagli del comportamento plastico durante un processo di deformazione. Quelli più semplici, riguardanti l'incrudimento isotropo, restituiscono una stima accettabile delle forze che si sviluppano durante il processo e sono molto usati nell'industria.

Tuttavia i modelli più sofisticati, come ad esempio quelli che coinvolgono incrudimento non lineare e criteri di snervamento più raffinati, danno una migliore valutazione dell'evoluzione dei processi di deformazione. Generalmente però, hanno bisogno di un gran numero di parametri.

Per questo motivo, quando i parametri da determinare sono in numero troppo elevato è necessario risolvere il problema come un problema di ottimizzazione non lineare, in cui si utilizzano risultati matematici e sperimentali.

Quindi, risolvere un problema inverso, consiste nel cercare un set di parametri di un modello costitutivo, per il quale i risultati ottenuti dalla realtà sperimentale e quelli ottenuti con simulazioni numeriche, siano il quanto più possibile simili.

La distanza tra il modello matematico e quello sperimentale diventa, così, una funzione da sviluppare, la quale è la funzione oggettivo di questi problemi.

4.3 Identificazione dei parametri

Le simulazioni *FEM*, dipendono fortmente dal tipo di modello costitutivo del materiale. Quindi l'affidabilità dei risultati dipende strettamente dall'accuratezza del modello, il quale dipende dai valori attribuiti ai parametri coinvolti.

Dagli inizi degli anni 90 sono state sviluppate molte tecniche per l'identificazione di tali parametri.

Per esempio nel 1991 Chaboche [9] ha utilizzato una procedura di prova per identificare i parametri del materiale, nel 1992 Schnur e Zabras [10] hanno utilizzato un accoppiamento tra un codice agli elementi finiti e un metodo di ottimizzazione.

L'identificazione dei parametri del materiale è un problema di minimizzazione dell'errore, in cui questo è determinato dalla differenza tra i risultati dell'approccio sperimentale e quello numerico. Per questi motivi è definita una funzione di costo.

La corretta definizione della funzione di costo è essenziale per l'intero processo di ottimizzazione e per la determinazione dei parametri costitutivi.

4.3.1 La funzione di costo

Secondo [11] la funzione di costo ideale dovrebbe rispettare i seguenti criteri:

- prima del processo di ottimizzazione i dati sperimentali dovrebbero essere filtrati;
- tutti i dati sperimentali dovrebbero essere considerati nel processo, avendo le stesse opportinità di essere ottimizzati;

- una funzione di costo dovrebbe essere in grado di affrontare problemi multi-obiettivo, i quali dovrebbero avere le stesso opportunità di essere ottimizzati e le cui caratteristiche non dovrebbero influire sulle prestazioni complessive;
- nelle ottimizzazioni con molte curve sperimentali, tutte dovrebbero avere le stesso possibilità di ottimizzazione, anche se il numero di punti sperimentali è diverso per ogni curva.

Una delle funzioni di costo più usate consiste nella somma dei quadrati della differenza di tensione a diversi livelli di deformazione:

$$Sobj(X) = \sum_{k=1}^{n} \frac{(\sigma_i^{\exp} - \sigma_i^{\text{num}}(X))^2}{W_{abs}} \tag{4.1}$$

Dove nell'Eq. 4.1 il numeratore è la differenza tra la tensione ottenuta sperimentalmente e quella ottenuta dalla simulazione numerica, per l'iesimo valore di deformazione, mentre W_{abs} è un fattore di ponderazione da adattare al problema di ottimizzazione in questione.

Analizzando la funzione obiettivo si nota facilmente che se la corrispondenza tra esperimenti fisici e modello numerico fosse perfetta, sarebbe uguale a 0.

Un'altra formulazione della funzione di costo molto diffusa è la seguente:

$$F(A) = \sum_{k=1}^{N} \frac{(P_i^{\text{num}}(A) - P_i^{\exp})^2}{N} \tag{4.2}$$

In Eq. 4.2, F(A) rappresenta la funzione di costo, A il set di parametri del modello costitutivo, N il numero di punti sperimentali e $\mathrm{P(A)^{num}}$ e $\mathrm{P^{exp}}$ sono i valori numerici e sperimentali ad un dato istante.

4.4 Metodi di ottimizzazione

L'ottimizzazione del modello costitutivo incontra difficoltà teoriche e numeriche, a causa dell'esistenza di *minimi locali*, che rendono difficile l'interpretazione e la selezione dei risultati ottenuti. In questi casi l'utente assume un ruolo molto importante nell'analizzare i risultati ottenuti, considerando la definizione fisica e il significato di ogni parametro.

Gli algoritmi per l'ottimizzazione come indicato in [12], si dividono in quattro categorie:

1. *derivative-free*;
2. *a gradiente*;
3. *trust region*;
4. *stocastici*.

4.4.1 Gli algoritmi *derivate-free* e *trust region*

Gli algoritmi *derivate free* sono caratterizzati dal fatto che non richiedono il calcolo del gradiente della funzione da minimizzare e sono usati in per minimizzazioni locali. Esempi di questo tipo sono gli algoritmi *Powell* e *Nelder-Mead*.

Quelli *trust region* approssimano il sottoinsieme della regione di ricerca con una funzione modello, spesso ne viene usata una quadratica. Anche questo tipo di algoritmo è usato in minimizzazioni locali. Fanno parte di questo grupo il *Dogleg* e il *Newton CG trust-region*.

4.4.2 Gli algoritmi *a gradiente*

Questo tipo di algoritmo oltre al valore della funzione utilizza le informazioni sul gradiente (metodi del primo ordine) e sull'*hessiano* (metodi del secondo ordine). Inoltre è noto per dipendere fortemente dal set di parametri di partenza, infatti per ridurre la possibilità di raggiungere massimi o minimi locali, spesso viene ripetuta l'ottimizzazione usando diversi set di input.

Esempi di questo tipo di algoritmi sono:

- il *BFGS*;
- il *L-BFGS-B*;
- il *Conjugate Gradient*;
- il *Truncated Newton*;
- il *Sequential Least Squares*;
- il *Levenberg–Marquardt*.

In particolare il *Levenberg–Marquard* [13] è una variante del classico metodo *Newton–Raphson*, il quale usa derivate prime e seconde.

Tale metodo consiste, dato un punto di partenza, nel costruire un'approssimazione quadratica della funzione obiettivo che corrisponde al valore della derivata prima e seconda in quel punto. Quindi viene minimizzata questa nuova funzione invece di quella obiettivo e il valore minimo ottenuto viene usato come punto di partenza nel passaggio successivo, ripetendo la procedura iterativamente.

Se l'ipotesi dei parametri iniziali non è ben accurata il metodo iterativo può portare a difficoltà di convergenza, quindi per ridurre al minimo questo problema si può applicare il metodo del gradiente per determinare il set di parametri di partenza.

4.4.3 Algoritmi *stocastici*

Gli aloritmi *stocastici* sono principalmente di due tipi, quelli *Basin-hopping* e quelli *evolutivi*, e sono usati in minimizzazioni globali.

La carratteristica principale di questo gruppo di algoritmi è di non essere deterministici e di avere un set di partenza scelto casualmente.

In particolare quelli evolutivi si basano, come già espresso, sul modello dell'evoluzione naturale suggerito da Darwin e sono noti per la loro capacità di ottenere buone soluzioni in problemi complessi di ottimizzazione.

Questi si dividono in tre categorie:

- gli algoritmi genetici;
- le strategie evolutive;
- la prorammazione evolutiva.

Il processo di base consiste in un apprendimento collettivo all'interno di una popolazione di individui, ognuno dei quali rappresenta un punto di ricerca nello spazio di potenziali soluzioni a un dato problema.

Al giorno d'oggi sono spesso utilizzati, con buoni risultati e prestazioni, ad esempio, nella soluzione di problemi strutturali e nella caratterizzazione dei modelli costitutivi, ma anche in campi come la biologia, l'economia, la robotica e le scienze sociali.

La popolarità di questi algoritmi è dovuta principalmente alle seguenti caratteristiche:

- minori probabilità di convergere verso minimi locali;
- il parallelismo implicito che si adatta perfettamente a calcolatori a più processori;
- la robustezza, poichè sono richieste informazioni solo sulla funzione obiettivo e non è necessario il calcolo delle derivate;
- capacità di trovare efficacemente una soluzione in un ampio spazio di ricerca attraverso operazioni probabilistiche;
- la capacità di lavorare con la maggior parte delle funzioni, come discontinue o moltimodali;
- l'utilizzo di poche regole deterministiche, rendendoli applicabili ad una varietà di problemi di ottimizzazione complessi.

L'implementazione di questi algoritmi, quindi, permette di trovare il minimo globale di una funzione nella maggior parte dei casi, ma il loro più grande svantaggio è di richiedere un numero di iterazioni largamente superiore rispetto agli altri tipi di algoritmi.

Questa caratteristica ne limita l'utilizzo, soprattutto in quei casi in cui non si ha a disposizione una grande quantità di tempo per risolvere il problema di ottimizzazione, casi in cui spesso si decide di impiegare gli algoritmi *a gradiente*. Tuttavia l'incremento della velocità di calcolo dei calcolatori, al giorno d'oggi, permette di arrivare ai risultati ottimizzati in un tempo accettabile.

4.4.4 Altri metodi

Sono stati sviluppati anche metodi di approssimazione convessa, come il *CONLIN method* [14], il metodo *MMA* [15] e il metodo *SQP* [16]. Nei problemi di ottimizzazione è anche usata la *programmazione lineare sequenziale*, SLP, [17] ma la sua efficienza è minore rispetto ai metodi precedenti.

Inoltre altri metodi e tecniche, nella loro combinazione, sono stati usati in processi di ottimizzazione complessi, come le *neural networks* [8], basate sull'inteligenza artificiale, le *tecniche di simulated annealing* [18], meta-algoritmi probabilistici usati spesso in problemi difficili in cui ci sono molte possibilità di incappare in minimi locali, *algoritmi immunitari* [19], che si ispirano al sistema immunitario umano.

Infine in [20] viene proposto un nuovo algoritmo per l'identificazione inversa dei parametri del materiale, nel caso del taglio dei metalli.

4.5 Significato fisico dei parametri

In generale gli approcci per la determinazione dei parametri costitutivi eseguono una ricerca orientata su un ampio spazio senza la conoscenza del significato fisico di ogni parametro e di conseguenza senza la conoscenza di intervalli ragionevoli entro cui essi dovrebbero rimanere.

Pertanto, nella determinazione dei parametri costitutivi, è necessario inserire alcuni vincoli nell'universo della ricerca tenendo conto del significato fenomenologico e fisico dei parametri e dei loro valori tipici.

in questo lavori, molti dei parametri del modello costitutivo hanno un significato fenomenologico ben definito, il che costringe ad un'attenta analisi dell'universo di ricerca di ogni parametro. Questo fattore aumenta di molto la difficoltà nell'identificazione e rende necessari dati sperimentali, ottenibili ad esempio da prove di trazione o di taglio a diverse temperature, al fine di utilizzarli come termini di paragone.

Quindi il problema di ottimizzazione nell'identificazione dei parametri segue le seguenti fasi:

1. determinazione del modello costitutivo che meglio rappresenti e prevedi il comportamento plastico del materiale a trazione e deformazione. Alcuni dei modelli disponibili sono il *Barlat91* [21], lo *Yld2000-2D* [22] e il modello di *Chaboche* [23];

2. scelta dell'algoritmo di ottimizzazione tra i vari disponibili, considerando vantaggi e svantaggi di ognuno e il tempo a disposizione;

3. Valutazione della validità fisica dei risultati ottenuti, confrtontandoli con dati sperimentali in possesso.

4.6 Problematica dei dati imprecisi e del modello inacccurato

Negli ultimi anni, sono stati sviluppati vari materiali strutturali per supportare l'obiettivo dei progettisti industriali. In molti campi di utilizzo, i materiali sono spesso utilizzati in condizioni operative severe come carico ciclico, alta temperatura, alta pressione e alto irraggiamento, ad esempio, se vengono utilizzati come recipienti in pressione e come tubi di una centrale nucleare. Per la valutazione affidabile dei comportamenti di deformazione di questi materiali, è indispensabile l'analisi termo-anelastica.

Sono stati proposti e discussi una gran varietà di modelli teorici, per descrivere un'ampia gamma di comportamenti plastici di materiali metallici. Le equazioni costitutive derivate da queste teorie coinvolgono molti parametri, che influenzano in modo significativo i comportamenti delle equazioni. Di conseguenza devono essere determinati parametri appropriati, in modo da poter esprimere i comportamenti accurati dei materiali.

Ogni equazione costitutiva ha il proprio metodo per l'identificazione dei parametri. Negli approcci convenzionali, il modello di interesse è prima approssimato ed i suoi parametri sono identificati sequenzialmente attraverso l'approccio di *curve fitting*. Tuttavia, questo processo è dipendente dal problema, e quindi potrebbe non essere facile se il modello è non lineare. Inoltre, il processo può produrre errori significativi dovuti all'approssimazione del modello, in particolare quando lo spazio dei parametri è molto ampio.

D'altra parte il progresso dei calcolatori ha aumentato la popolarità di un approccio in cui tutti i parametri sono identificati simultaneamente e vengono utilizzati metodi di ottimizzazione come illustrato nel Cap.4, per trovare il set di parametri, regolandoli fino a fornire il miglior accordo tra i dati misurati e quelli forniti dal metodo numerico.

Queste tecniche, tuttavia, possono fallire quando i dati misurati sono imprecisi o le equazioni del modello sono inaccurate, poiché possono complicare la funzione obiettivo.

Due strategie principali per affrontare questo problema possono essere:

- modificare la funzione obiettivo mediante una regolarizzazione o scegliendo fattori di ponderazione diversi;
- utilizzare un metodo di ottimizzazione diverso in modo che l'ottimizzazione non possa fallire.

4.7 Confronto tra algoritmi

In precedenza sono stati mostrati i vari metodi di ottimizzazione e i rispettivi algoritmi per risolvere il problema dell'identificazione dei parametri all'interno di un modello costitutivo.

Sorge, quindi, spontaneo il dubbio di quale fra gli algoritmi elencati si adatti meglio ad alcune circostanze, quale necessita di più parametri di input, e quale di più iterazioni, quindi di più tempo di calcolo.

Sono stati svolti moltissimi studi a riguardo, ad esempio in [24] viene proposto un nuovo algoritmo genetico e viene confrontato con gli altri esistenti, in [25] vengono confrontati agloritmi genetici e a grandiente, in [12] vengono messi alla prova la maggior parte degli algoritmi esisetenti.

L'approccio per confrontare i vari metodi potrebbe essere quello di metterli subito alla prova nella situazione reale, quindi il modello costitutivo che si intende utilizzare, o usare preliminarmente delle funzioni test in cui vengono implementati i diversi algoritmi così da avere un ulteriore metro di paragone.

Gli elementi su cui vengono confrontati gli algoritmi sono spesso il valore della funzione di costo raggiunto, il tempo impiegato nel calcolo, il numero di iterazioni o generazioni e i parametri costitutivi ottenuti al termine della procedura di ottimizzazione.

I risultati a cui si è arrivati non danno un giudizio univoco, poichè non esiste un algoritmo migliore di un altro in ogni caso, ma vanno valutati situazione per situazione in base al tempo a disposizione e ai dati in ingresso in possesso dell'utente.

Capitolo 5

Gli algoritmi combinati

5.1 Introduzione

Non esiste un unico algoritmo abbastanza robusto tale da gestire ogni problema di ottimizzazione e che asssicuri il raggiungimento del minimo globale. Tuttavia, combinando i vari metodi di ottimizzazione, si possono sfruttare i punti di forza di ognuno di essi, così da ridurre le probabilità di convergere ad un minimo locale ed invece incrementare quelle di raggiungere quello globale.

Come detto nel Cap.4 gli algoritmi basati sul *gradiente* sono oggettivi, deterministici e richiedono un basso sforzo di calcolo. Tuttavia, la prestazione di questi algoritmi dipende fortemente dai parametri di partenza e, soprattutto nel caso di funzioni non convesse, hanno alte probabilità di arrivare ad un minimo locale.

D'altro lato gli *algoritmi evolutivi* sono noti per avere una bassa dipendenza dai dati di input e per raggiungere molto spesso il minimo globale. Lo svantaggio è che necessitano di un grande numero di iterazioni e di progressi della funzione di costo, ciò aumenta lo sforzo di calcolo del calcolatore e il tempo di convergenza. Tuttavia nel Cap.3 è stato espresso che, per questo tipo di algoritmi, si parla di *parallelismo implicito*, quindi essi si prestano bene a lavorare su più processori contemporaneamente, fattore che può ridurre il problema dell'eccessivo tempo di calcolo.

Analizzando le caratteristiche di questi due algoritmi, è chiaro che sono dotati di punti di forza e di debolezza speculari, per cui potrebbero avere un grande potenziale, se utilizzati in strategie di ottimizzazione combinate.

Tali strategie sono classificate in:

- *a cascata*;
- *paralelele*;
- *ibride*.

In questi metodi occore definire un criterio per cui si passa da un algoritmo ad un altro, quelli più utilizzati sono:

- un valore minimo del delta della funzione di costo tra un'iterazione e l'altra;

- un numero massimo ammissibile di iterazioni, dopo il quale entra in gioco l'altro algoritmo.

5.2 Strategie

5.2.1 Il metodo a cascata

Il metodo *a cascata* consiste in una procedura in più fasi in cui i vari algoritmi vengono attivati uno dopo l'altro in una sequenza pre-specificata. Ad esempio si può far entrare in funzione prima quello genetico e poi quello a gradiente o viceversa, nel caso in cui si opti per quella a due fasi. Altrimenti aumentando queste ultime, aumentano anche le possibili combinazioni con cui si possono far entrare in funzione i metodi a disposizione.

5.2.2 La strategia parallela

La *strategia parallela* consiste nell'usare tante strategie mono-fase parallelamente, cambiando ogni volta i dati di partenza. Quindi si avranno n diversi punti d'inizio, con ognuno un set di parametri di input generato casualmente, e n set di parametri ottimizzati corrispondenti.

Questi ultimi entrano poi in un *selection box*, in cui vengono scelte le migliori soluzioni, in base al valore della funzione obiettivo raggiunto.

In Fig.5.1 è mostrato un esempio di strategia mono-fase, mentre in Fig.5.2 si nota lo schema della *strategia parallela*, in cui x sono i parametri in ingresso, mentre x^* sono quelli in uscita.

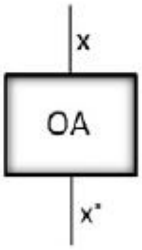


Figura 5.1: Strategia mono-fase.

Il vantaggio apportato da questa tecnica è quello di rendere gli algoritmi impiegati meno dipendenti dal punto di partenza, facendogli inoltre acquisire proprietà probabilistiche.

Gli algoritmi a gradiente subiscono dei netti miglioramenti, mentre quelli evolutivi, essendo già intrinsecamente paralleli e possedendo caratteristiche probabilistiche, ne risentono di meno.

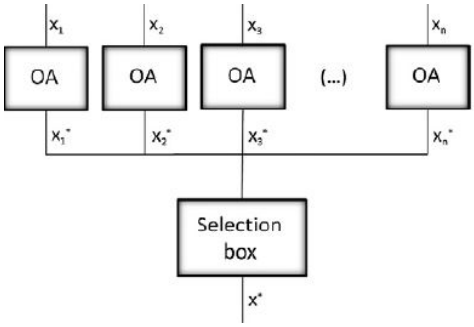


Figura 5.2: Strategia parallela.

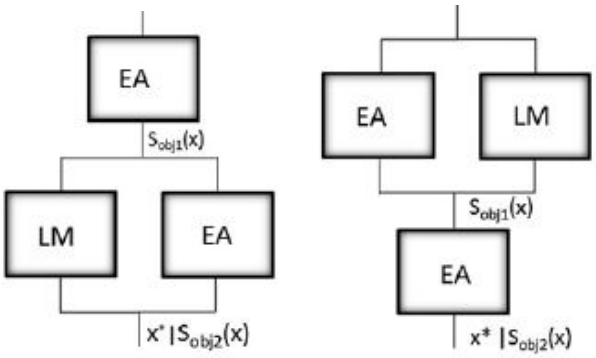
5.2.3 La strategia ibrida

La *strategia ibrida* combina le potenzialità di quella a cascata e quella parallela. Un esempio di utilizzo potrebbe essere quello di usare l'algoritmo genetico per ottenere un punto vicino il minimo, e quindi quello a gradiente, sfruttando le sue abilità nella ricerca locale, per ottenere il minimo.

In questo modo, ancora una volta, combiniamo i vantaggi dei due algoritmi per ottenere un risultato migliore.

Questo schema può subire molte variazioni, ad esempio si può utilizzare il GA, seguito da una strategia parallela tra GA e quello a gradiente, Fig.5.3a , o ancora, partire da un metodo parallelo tra i due e poi utilizzare quello che si desidera come step finale, Fig.5.3b.

I risultati di queste combinazioni sono legati ai punti di partenza, mentre applicando i metodi a cascata parallelamente, come mostrato in Fig.5.4, ci si svincola da questo problema e si ottengono caratteristiche più probabilistiche.



(a) Esempio 1 (b) Esempio 2

Figura 5.3: Esempio di strategie ibride.

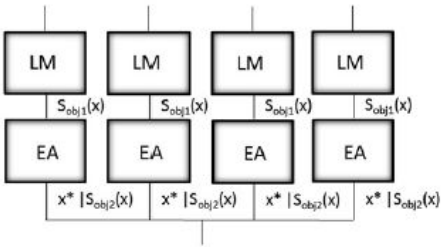


Figura 5.4: Strategia ibrida con cascata parallela.

5.3 Vantaggi ottenuti

Da [26] si nota come l'algoritmo a gradiente, nello specifico il *Levenberg–Marquardt*, è più efficiente di quello genetico quando vengono usati separatemente. Infatti raggiunge un valore minore della funzione obiettivo con un numero nettamente inferiore di iterazioni.

Per quanto riguarda la strategia *a cascata*, in un modello di incrudimneto elasto-plastico non linare, si ottengono modesti miglioramenti in termini di funzione di costo utilizzando, nell'ordine, quello evolutivo e poi quello a gradiente. Mentre nel caso di un modello iperelastico si ottengono miglioramenti in termini dei parametri ottimizzati ottenuti.

Usando la strategia *parallela* si notano buoni miglioramenti soprattutto in quei casi in cui la funzione obiettivo ha molti minimi locali. Infatti, avendo un numero pari a n di punti di partenza, la probabilità di raggiungere il minimo globale incrementa di n, o in ogni caso, quella di raggiungere un minimo locale inferiore, aumenta.

Infine, per quanto riguarda gli *algoritmi ibridi*, si è notato come, usando i risultati dell'algoritmo evolutivo come set di partenza per quello a gradiente, si raggiungono dei parametri finali di alta qualità. Inoltre il tempo di calcolo impiegato è molto basso. Infatti attraverso questa tecnica, il numero di generazioni di quello evolutivo può essere nettamente ridotto e quello a gradiente converge in pochi passaggi, partendo da una situazione già molto vicina a quella ottimale.

Capitolo 6

Il Metodo dei Campi Virtuali

6.1 Introduzione

Nel caso delle lamiere il processo di laminazione induce un'orientazione preferenziale dei grani nella struttura cristallina, generando un'anisotropia [27]. Ciò significa che il materiale reagisce in maniera diversa a tensione e deformazione, in base all'orientazione delle stesse.

Sono stati sviluppati molti modelli di anisotropia plastica come il *Yld200-2D* e il *Yld2004-18p*, i quali sono molto adatti ad applicazioni industriali grazie alla loro capacità di riprodurre la risposta anisotropa del materiale.

Ciononostante per definire i parametri costitutivi sono necessari un gran numero di test sperimentali, sia monoassiali, che biassiali, su provini cruciformi. Questo fa accrescere i costi da conseguire per una caratterizzazione completa e necessita di particolari strumenti di misura, non sempre disponibili nei laboratori industriali.

Per aggirare questi problemi negli ultimi anni sono stati sviluppati dei metodi per definire inversamente anche i modelli di materiali più complessi, grazie anche all'apporto dato dalla diffusione di tecniche di misurazione a pieno campo come la DIC, *Digital Image Correlation*. Infatti, l'uso di campi eterogenei di deformazione permette di raccogliere un gran numero di informazioni riguardo al materiale con un singolo test, riducendo quindi gli sforzi sperimentali.

Un metodo ampiamente diffuso è, ad esempio, la FEMU, *Finite Element Model Updating*, dove vengono utilizzati test simulati, come nella FEM, variando iterativamente i parametri del materiale, fino a minimizzare la distanza tra le variabili misurate e i risultati numerici. Uno dei maggiori limiti di questa tecnica è quello di dover definire precisamente le condizioni al contorno, le quali possono essere identificate facilmente nel caso di una prova monoassiale di trazione, ma nel caso di test più complessi la situazione non è di semplice risoluzione. Un'ulteriore limite è che la FEMU si serve di simulazioni agli Elementi Finiti per ogni set di parametri, quindi il processo è molto lento e complesso.

Un'altro esempio di metodi inversi è il *Metodo dei Campi Virtuali* (VFM) , usato in molti problemi di plasticità non lineare, nel caso di comportamento isotropo o anisotropo. Questo approccio dipende dalla scelta dei campi virtuali, la quale può essere effettuata dall'utente o con una procedura dedicata.

Il vantaggio del VFM rispetto alla FEMU è che le condizioni al contorno sono determinate dai campi virtuali, a patto che questi siano cinematicamente coerenti con il problema.

In [28] viene proposto un metodo inverso per i problemi di grandi deformazioni, basato sul VFM, allo scopo di ridurre gli sforzi sperimentali da sostenere e di aumentare l'efficienza di calcolo dell'intero processo. Le prossime considerazioni e valutazioni saranno relative a tale studio.

6.2 Virtual Fields Method

L'idea di base è quella di disaccoppiare l'identificazione del comportamento all'incrudimento del materiale da quella riguardante il modello di plasticità anisotropa. La prima analizzando la *curva tensione-deformazione*, mentre la seconda sfrutta la tecnica del VFM.

Dal punto di vista teorico, questa tecnica ha come punto di partenza il *Principio dei Lavori Virtuali* (PVW), secondo il quale, trascurando le forze d'inerzia, il lavoro virtuale delle forze esterne (EVW) equivale a quello delle forze interne (IVW), Eq.6.1.

$$\int_{\mathcal{B}_0} T_{1pk} : \delta F^* dV_0 = \int_{\partial\mathcal{B}_0} (T_{1pk}\hat{n_0})\delta u^* dA_0 \tag{6.1}$$

in cui $\mathcal{B}_\prime$ è un corpo nello spazio, nella sua configurazione iniziale, che subisce un processo di deformazione al tempo t, δu^* è un campo di spostamento virtuale arbitrario, ammissibile dal punto di vista cinematico, δF^* il corrispondente *deformation gradient* e infine T_{1pk} è il *Primo tensore della tensione di Piola–Kirchhoff*. Quest'ultimo è definito come segue

$$T_{1pk} = det(F)\sigma F^{-T} \tag{6.2}$$

dove σ è il *tensore della tensione di Cauchy*.

Viene definita la funzione di costo, Eq.6.3, come la differenza fra i due termini dell'equazione, la quale dipende dai parametri costitutivi ϵ, dagli step di carico t te dagli spostamenti virtuali δu^*.

$$\psi(\epsilon, \delta u^*, t) = \int_{\mathcal{B}_0} T_{1pk} : \delta F^* dV_0 - \int_{\partial\mathcal{B}_0} (T_{1pk}\hat{n_0})\delta u^* dA_0 \tag{6.3}$$

Se i parametri costitutivi sono esatti, la funzione di costo ψ deve essere uguale a zero per ogni t e per ogni δu^*.

Nei problemi non lineari i parametri costitutivi sono definiti come quelli tali da minimizzare la funzione di costo.

Spesso, la raccolta dei dati di spostamento e deformazione della maggior parte del campione non è un processo semplice e per il *VFM* vengono utilizzati solo i dati a pieno campo misurati sulla sua superficie. Ciò porta ad ipotizzare una distribuzione costante dei campi meccanici nella direzione dello spessore; quindi,

poiché l'identificazione viene eseguita su provini ricavati da lamiere, si assume la condizione di tensione piana, dal momento che, per definizione, la dimensione dello spessore è nettamente inferiore rispetto alle altre due.

6.2.1 La scelta dei campi virtuali

Il Principio dei Lavori Virtuali è valido per ogni campo virtuale cinematicamente coerente con il problema, tuttavia la loro scelta incide sui risultati dell'identificazione, poiché essi forniscono pesi diversi ai punti materiali contenuti nella funzione di costo. Nella scelta dei campi virtuali si possono seguire due strategie, che si distinguono per il modo in cui questi vengono generati:

- *UDVFs*;
- *SBVFs*.

Il primo approccio, UDVFs, prevede che l'utente definisca i campi utilizzando funzioni polinomiali o armoniche, con le quali ha avuto a che fare nella propria esperienza. L'utilizzo di questo metodo presuppone una profonda conoscenza del problema in questione.

Il secondo approccio, SBVFs, si basa sulla generazione automatica dei campi, in relazione ai criteri disponibili. Tale metodologia riduce l'importanza della conoscenza e dell'esperienza dell'utente e quindi risulta più facilmente fruibile nelle applicazioni industriali.

Inizialmente questo approccio era stato utilizzato in un caso di elasticità anisotropa lineare, in cui i campi virtuali avevano il compito di ridurre gli effetti disturbanti nell'identificazione dei parametri costitutivi. In seguito è stato dimostrato [29] che, estendendo l'applicazione al caso elasto-plastico isotropo, l'implementazione dei campi virtuali è capace di migliorare l'accuratezza dei risultati.

6.2.2 Il modello Yld2000-2D

Il modello utilizzato per descrivere il comportamento anisotropo dei materiali è il Yld2000-2D [22], che utilizza due trasformazioni lineari del tensore deviatorico delle tensioni di Chaucy. La funzione di snervamento ϕ è definita dell'Eq.6.4:

$$\phi = \phi' + \phi'' = 2\bar{\sigma^a} \tag{6.4}$$

dove $\bar{\sigma}$ indica la tensione equivalente e le due funzioni sono:

$$\phi' = (X'_1 - X'_2)^a \tag{6.5}$$

$$\phi'' = (2X''_2 + X''_1)^a + (2X''_1 + X''_2)^a \tag{6.6}$$

ϕ è la somma di due funzioni simmetriche rispetto a X'_i e X''_j. Questi ultimi sono i valori principali delle matrici X' e X":

$$X'_i = \frac{1}{2}(X'_{11} + X'_{22} + \sqrt{(X'_{11} - X'_{22})^2 + 4(X'_{12})^2}) \tag{6.7}$$

$$X''_j = \frac{1}{2}(X''_{11} + X'_{22} + \sqrt{(X''_{11} - X''_{22})^2 + 4(X''_{12})^2}) \tag{6.8}$$

Le componenti di X' e X" sono ottenute dalle seguenti trasformazioni lineari del *tensore della tensione di Cauchy*:

$$\mathbf{X'} = \mathbf{L'}\ \sigma \tag{6.9}$$

$$\mathbf{X"} = \mathbf{L"}\ \sigma \tag{6.10}$$

dove:

$$\begin{bmatrix} L'_{11} \\ L'_{12} \\ L'_{21} \\ L'_{22} \\ L'_{66} \end{bmatrix} = \begin{bmatrix} 2/3 & 0 & 0 \\ -1/3 & 0 & 0 \\ 0 & -1/3 & 0 \\ 0 & 2/3 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_7 \end{bmatrix} \tag{6.11}$$

$$\begin{bmatrix} L''_{11} \\ L''_{12} \\ L''_{21} \\ L''_{22} \\ L''_{66} \end{bmatrix} = \frac{1}{9} \begin{bmatrix} -2 & 2 & 8 & -2 & 0 \\ 1 & -4 & -4 & 4 & 0 \\ 4 & -4 & -4 & 1 & 0 \\ -2 & 8 & 2 & -2 & 0 \\ 0 & 0 & 0 & 0 & 9 \end{bmatrix} \begin{bmatrix} \alpha_3 \\ \alpha_4 \\ \alpha_5 \\ \alpha_6 \\ \alpha_8 \end{bmatrix} \tag{6.12}$$

Nelle Eq.6.9, 6.10, 6.11, 6.12 σ è il *tensore delle tensioni di Cauchy* e $\alpha_1 \dots \alpha_8$ sono gli otto parametri da caratterizzare del materiale.

L'esponente è un valore predefinito, che dipende dalla struttura cristallina del materiale. Vale 6 nel caso di struttura cubica a corpo centrato, mentre assume il valore di 8 con quella cubica a facce centrate. I restanti 8 parametri richiedono dati sperimentali in termini di tensioni e deformazioni lungo lo spessore, ottenibili con stati di carico monoassiali e biassiali.

In Fig.6.1 è mostrata la risposta del materiale ottenuta con il modello *YLD2000-2D* [30]. Questo modello ha una sensibilità sufficiente per utilizzare come input, oltre al *flow stress*, anche i valori dell'*R-value*, che possono essere più piccoli rispetto al caso isotropo, in cui il valore è 1. La Fig.6.1 infatti, mostra la superficie di snervamento ottenuta con $r_0 = 0.20$, $r_{45} = 0.28$, $r_{90} = 0.20$. La linea tratteggiata, invece, mostra la risposta del materiale che si ottiene con la maggior parte delle altre funzioni

di snervamento. Inoltre sono anche riportati i dati sperimentali ottenuti con un campione di Al–2.5wt% Mg.

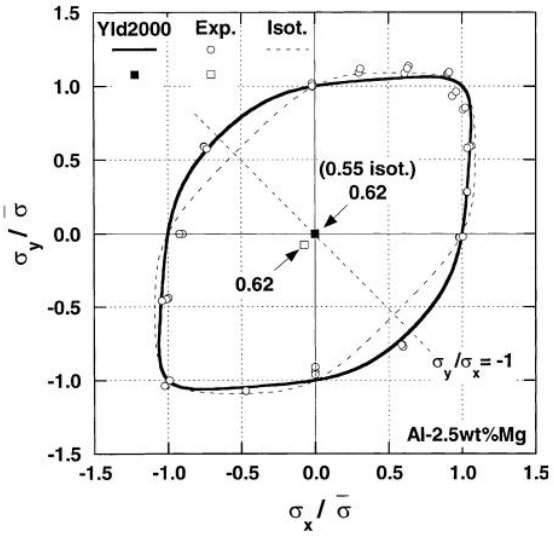


Figura 6.1: Superficie ottenuta con *YLD2000-2D*.

Si nota come, usando questi valori di R, la superficie ottenuta usando il modello *Yld2000-2D* è in ottimo accordo con i dati sperimentali.

6.3 Applicazione del modello inverso

I dati in ingresso sono la geometria del provino, i campi degli spostamenti e le forze, misurate con celle di carico. Il primo passo è quello di definire i parametri di incrudimento, usando il metodo della curva tensione – deformazione. Questo recupera tale curva come funzione a tratti, attraverso una funzione inversa, e ottiene una buona efficienza computazionale. Questa procedura permette di includere informazioni del materiale, che possono essere usate per aumentare la qualità del campo delle deformazioni.

Lo step successivo consiste nell'applicare il *Metodo dei Campi Virtuali* per identificare i coefficienti del modello di anisotropia. Questi vengono cambiati iterativamente dall'algoritmo di ottimizzazione che si è scelto, finché la funzione di costo non arriva al suo valore minimo. La scelta di escludere i parametri di incrudimento dal VFM favorisce la sua convergenza, quindi vengono ridotti significativamente gli sforzi di calcolo. Questa procedura permette di scegliere tra i due tipi di campi virtuali illustrati precedentemente.

I vari step della procedura inversa per l'identificazione sono mostrati in Fig.6.2.

L'uso degli SBVFs permette il loro ricalcolo ad ogni iterazione svolta dall'algoritmo di ottimizzazione, finché lo scarto tra due valori consecutivi della funzione di costo sia sotto una tolleranza scelta. Al fine di migliorare l'efficienza di calcolo dell'intero

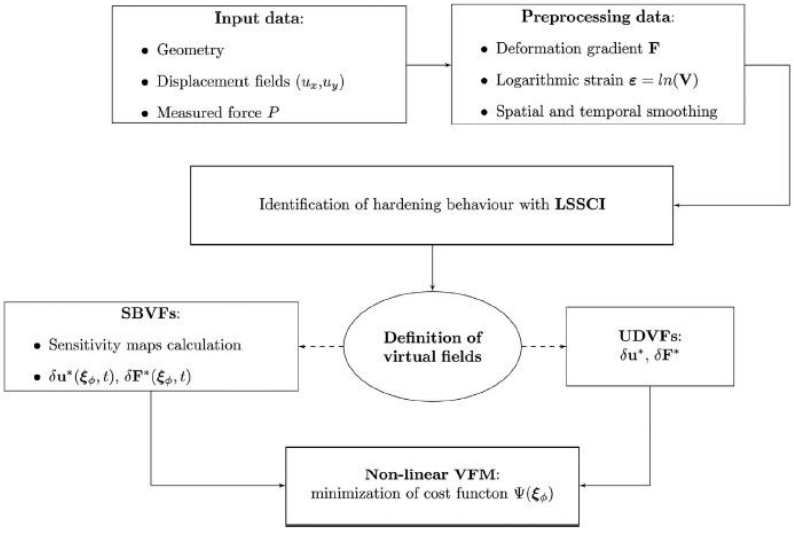


Figura 6.2: Passaggi dell'identificazione.

processo, vengono utilizzati gli SBVFs, senza che questi si aggiornino durante la minimizzazione.

Inoltre i parametri del modello anisotropo, nell'iterazione iniziale, sono posti tutti uguali ad 1, quindi i campi sono ottenuti perturbando il caso isotropo.

Due tra gli algoritmi disponibili per questa ottimizzazione sono il *Squential Quadratic Programming* e il *Levemberg-Marqardt algorithm* . Viene scelto il secondo perché, nonostante la sua accuratezza sia influenzata dalla scelta dei parametri iniziali, raggiunge la convergenza con un numero minore di iterazioni.

La fattibilità di questo metodo viene valutata con dati numerici, nello specifico vengono simulate prove di trazione a diverse angolazioni (0, 45, 90 gradi rispetto a RD) su provini con intagli circolari. Questa caratteristica produce un campo di deformazione eterogeneo sulla superficie del provino, con una diffusa zona plastica, mettendo a disposizione una grande quantità di informazioni.

Inoltre, vengono svolte prove sperimentali per caratterizzare il modello *Yld2000-2D* sull'acciaio BH340, prima seguendo il protocollo standard e poi utilizzando la strategia inversa, così da avere un'ulteriore metro di paragone.

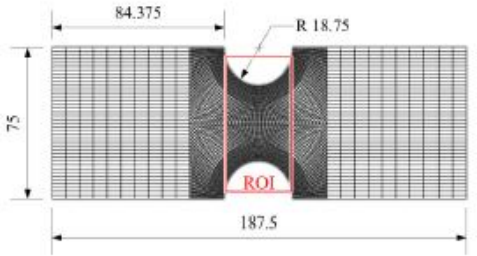


Figura 6.3: Gemoetria del provino.

Il significato fisico dei coefficienti del modello *Yld2000-2D* non è immediato e semplice, quindi viene adottato un metodo efficace per visionare l'accuratezza dei risultati ottenuti. Tale metodo consiste nel valutare la capacità dei parametri di prevedere il *Normalized flow stress*, cioè la tensione in campo plastico normalizzata, e l'*R-value*, che consiste nel coefficiente di anisotropia normale, cioè il rapporto fra la deformazione lungo la direzione della larghezza e quella lungo la direzione dello spessore. La tensione è stata normalizzata usando uno sforzo biassiale, così da includere nel grafico informazioni riguardanti anche questo stato di tensione.

6.4 Risultati

Si evince che il disaccoppiamento tra il comportamento all'incrudimento e la funzione di snervamento porta risultati migliori e riduce il tempo di calcolo. Infatti a parità di vincoli, questo approccio, permette di eliminare dei gradi di libertà legati alle variabili che regolano il comportamento all'incrudimento. Questo fa sì che l'algoritmo di ottimizzazione si concentri su un insieme ridotto di variabili e quindi, di conseguenza, impieghi meno tempo per raggiungere la convergenza.

Inoltre si nota come gli *SBVFs* ottengano risultati migliori rispetto ai *UDVFs* e che le impostazioni per la loro generazione debbano essere definite opportunamente, poichè influenzano i risultati.

L'identificazione raggiunge una buona riproduzione dell'anisotropia con tensione biassiale e quindi può essere applicata con successo nella caratterizzazione dei parametri del modello *Yld2000-2D*.

Infine come si nota in Fig.6.4, in cui sono confrontate le superfici ottenute con i dati sperimentali, i risultati ottenuti con il VFM e quelli ottenuti con la calibrazione standard, il Metodo dei Campi Virtuali si avvina molto alla realtà sperimentale.

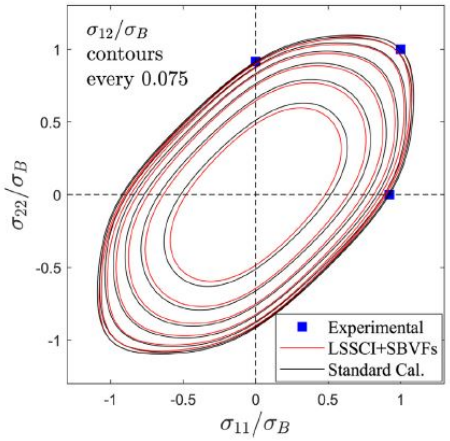


Figura 6.4: Superfici ottenute con VFM, procedura standard e dati sperimentali.

Capitolo 7

Implementazione degli algoritmi genetici nella strategia inversa

7.1 Introduzione

Come già espresso nel Cap.4, per affrontare complessi problemi, come l'identificazione dei parametri costitutivi del materiale, il metodo *FEM* ha bisogno di precisi dati di input, come la geometria, la legge non lineare riguardante il comportamento anisotropo e la legge di attrito.

Una possibile alternativa a questo approccio *diretto*, è quello *inverso*, come la FEMU o il VFM, che consiste nel cercare un set di parametri per i quali realtà sperimentale e modello numerico si avvicinino il più possibile. La distanza tra questi due modelli è, perciò, la funzione di costo da minimizzare.

Tale funzione dovrebbe avere le caratteristiche indicate in precedenza ed alcuni esempi possono essere le Eq.4.1 e Eq.4.2.

L'ottimizzazione da svolgere non è banale e incontra molte difficoltà teoriche e numeriche, a causa dei minimi locali, e si richiede all'utente un'attenta analisi dei risultati ottenuti, di cui va controllata la validità fisica.

Nel Cap.4 sono stati elencati i vari algoritmi di ottimizzazione che possono essere utilizzati in questo problema, alcuni non utilizzano le informazioni sul gradiente, altri ne necessitano e altri ancora sfruttano caratteristiche probabilistiche. Inoltre si è notato come, gli algoritmi che sfruttano il gradiente, hanno una forte dipendenza dal set di parametri iniziali, mentre quelli di base stocastica ne risentono molto meno.

Tuttavia, è anche noto che, quelli a gradiente sono ottimi per ricerche locali e arrivano a convergenza con un minor tempo di calcolo, rispetto a quelli stocastici, che invece performano meglio in ricerche globali ma hanno bisogno di un gran numero di iterazioni e sviluppi della funzione di costo prima di arrivare all'ottimo. Infine questi ultimi hanno meno probabilità di convergere ad un minimo locale, rispetto a quelli che sfruttano le informazioni sulle derivate.

Inoltre nell'identificazione dei parametri possono essere usati diversi modelli, in base alle situazioni. Esempi di modelli disponibili sono il *Barlat91*, il *Yld2000-2D*, quello di *Chaboche* e il modello *Hill48*.

Nell'attività sperimentale di questa tesi è stata svolta l'identificazione dei parametri costitutivi appartenenti ad un modello di plasticità anisotropa, si è optato per il *Yld2000-2D*, e ci si è basati sul *Metodo dei Campi Virtuali*, illustrato nel Cap.6 e in [28].

L'algoritmo di ottimizzazione usato è l'*algoritmo genetico* e l'implementazione è stata svolta attraverso il software *Matlab*.

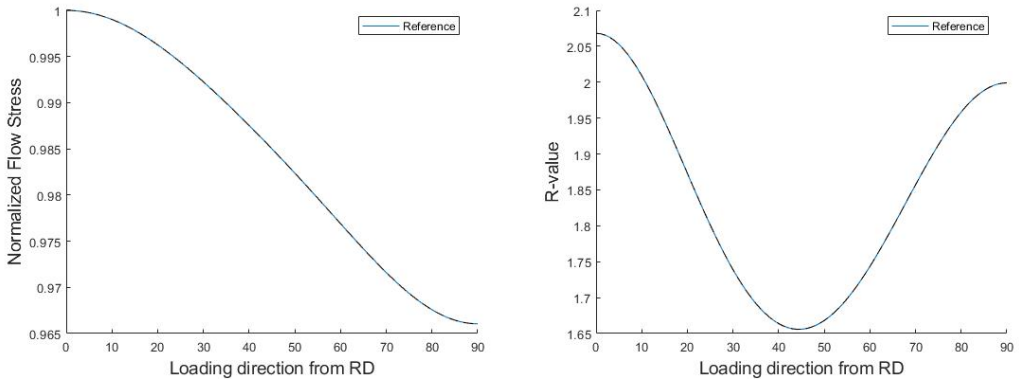
La validità fisica dei risultati ottenuti è stata analizzata attraverso le curve di *Normalized flow stress* e dell'*R-value* corrispondenti, che sono state confrontate con quelle ottenute con dei parametri di riferimento.

I parametri di riferimento sono indicati in Tab.7.1, in cui i primi tre parametri riguardano la legge di incrudimento, gli altri otto quelli del modello costitutivo e l'ultimo è l'esponente fisso pari a 6.

Le curve ottenute con i dati di riferimento sono mostrate in Fig.7.1a e in Fig.7.1b.

Tabella 7.1: Parametri di riferimento per la legge di incrudimento e il modello costitutivo adottato, *Yld2000-2D*.

K/MPa	ϵ_0	N	α_1	α_2	α_3	α_4	α_5	α_6	α_7	α_8	α
980	0.02	0.48	1.11	1.35	1.21	1.11	1.07	0.96	1.21	1.15	6



(a) Curva del *Normalized flow stress* di riferimento. (b) Curva dell'*R-value* di riferimento.

Figura 7.1: Curve di riferimento.

7.2 Ambiente Matlab

Tutte le ottimizzazioni sono state svolte con il software *Matlab*, che mette a disposizione l'algoritmo genetico e molte opzioni riguardanti tutti i vari aspetti di questo complesso metodo.

I dati di input che vengono richiesti dal software in questione sono:

- la funzione obiettivo;
- il numero di variabili;
- le equazioni e le disequazioni lineari di vincolo;
- i valori limite superiori e inferiori che possono assumere i parametri da ottimizzare;
- i vincoli non lineari;
- le opzioni da impostare in base al tipo di ottimizzazione che si intende svolgere.

Inoltre *Matlab* offre l'opportunità di selezionare i dati di output che l'utente ha interesse nel visionare, questi sono:

- la soluzione finale, cioè il vettore che rappresenta il miglior punto trovato dall'algoritmo durante le proprie iterazioni;
- il valore minimo della funzione obiettivo raggiunto;
- l'*exitflag*, cioè un numero intero che spiega la ragione per cui l'algoritmo si è fermato. Quest'ultima potrebbe essere il numero massimo di iterazioni o il tempo massimo raggiunto, o ancora il caso in cui la variazione della funzione di costo tra due iterazioni consecutive sia minore di una certa soglia di tolleranza, decisa a priori dall'utente;
- la popolazione finale, restituita come matrice;
- il valore della funzione di fitness relativa alla popolazione finale.

Infine sono disponibili molte opzioni per ottenere l'ottimizzazione che si vuole, le più importanti sono:

- la probabilità di cross-over, che influisce sul numero di individui che subiranno l'azione dell'operatore genetico;
- il criterio di terminazione;
- il numero di individui nella popolazione iniziale;
- il numero di generazioni prodotte;
- la funzione che regola la selezione degli individui genitori;
- la possibiità di svolgere il calcolo in parallelo, caratteristica intrinseca degli algoritmi genetici.

7.3 Implementazione

Nel valutare l'utlizzo di questo tipo di algoritmo nel problema dell'indentificazione dei parametri del materiale con una strategia inversa, sono state fatte sei ottimizzazioni, variando di volta in volta le opzioni disponibili, così da provare ad esplorare e comprendere il modo di operare del GA. Infatti, essendoci molte caratteristiche da poter variare, in base a quelle scelte in un caso l'algoritmo si può comportare in maniera totalmente diversa rispetto ad altri casi, quindi è bene avere a disposizione una panoramica di varianti per poter dare una valutazione complessiva.

Come detto nel Cap.3 l'algoritmo come primo step crea, in maniera casuale, la prima popolazione di individui/soluzioni. In tutte le prove svolte in questa tesi il numero di individui della popolazione iniziale è tenuto costante, pari a 80. Il motivo di questa scelta, risiede nel fatto che questo la variazione di questo importante parametro avrebbe generato risultati difficilmente confrontabili fra loro.

Le opzioni che sono state fatte variare nelle prove sono:

- il numero delle generazioni;
- il tasso di cross-over;
- il criterio di selezione.

Il numero delle generazioni è stato fatto variare da un minimo di 40 ad un massimo di 80, la probabilità di cross-over tra 0.65 e 0.95, infine per i metodi di selezione da scegliere, elencati nel Cap.3, si è optato per il metodo di default del software, *stochunif*, e quello a *roulette*.

Il metodo *stochunif* imposta una linea in cui ogni genitore ne occupa una porzione proporzionale al proprio valore di fitness e l'algoritmo avanza a passi costanti, selezionando gli individui genitori che creeranno la prossima generazione. Invece il metodo a *roulette* simula la ruota di una roulette, in cui le porzioni di questa sono assegnate agli individui proporzionalmente alla loro fitness, mentre il lancio della pallina è simulato da un numero causuale che seleziona una delle porzioni corrispondente ad un indivuo.

I sei tipi algoritmi genetici usati nelle prove verrano chiamati rispettivamente GA1, GA2, ..., GA6. Le caratteristiche di ognuno sono mostrate nella Tab.7.2.

7.4 Risultati ottenuti

7.4.1 Introduzione ai risultati

I risultati ottenuti sono classificati in:

- parametri ottimizzati;
- valore minimo della funzione di costo;

Tabella 7.2: Caratteristiche degli algoritmi usati nelle 6 prove.

Nome	*Popolazione in.*	*Generazioni*	*Tasso di crossover*	*Selezione*
GA1	80	40	0.8	stochunif
GA2	80	80	0.8	stochunif
GA3	80	40	0.95	stochunif
GA4	80	40	0.65	stochunif
GA5	80	40	0.65	roulette
GA6	80	64	0.5	roulette

- numero di iterazioni e tempo di calcolo impiegato.

Inoltre è stata verificata la validità fisica dei parametri ottenuti alla fine di ogni ottimizzazione attraverso il confronto con quelli di riferimento, mediante le rispettive curve del *Normalized flow stress* e dell'*R-value*.

Sono stati poi costruiti i grafici relativi all'andamento della funzione di costo per ogni prova, in relazione alle iterazioni e quelli che mostrano la minimizzazione di tale funzione.

Infine, per comprendere meglio il modo di operare degli algoritmi in questione, sono stati calcolati i valori della *deviazione standard* relativi alla funzione di costo e agli otto parametri, per i primi 80 individui, quindi la popolazione iniziale, e per tutte le iterazioni.

7.4.2 Risultati

I parametri ottimizzati ottenuti nelle varie prove sono illustrati in Tab.7.3.

Tabella 7.3: Parametri ottimizzati ottenuti con le 6 prove.

	α_1	α_2	α_3	α_4	α_5	α_6	α_7	α_8
GA1	1.3590	1.3522	1.3924	1.2022	1.3211	2.2192	1.3923	0.6679
GA2	0.5324	1.8278	2.5383	1.1378	0.3888	1.8084	1.2496	0.4778
GA3	1.0421	1.3106	1.9656	1.2466	1.1008	1.1120	2.0099	0.7694
GA4	1.3838	1.1337	2.1492	1.2342	1.1093	1.2939	1.2393	0.5989
GA5	1.5575	1.0692	1.8440	1.1191	1.3235	2.5020	1.3164	0.3967
GA6	1.3347	1.2393	2.4677	1.3111	1.0015	0.8756	1.2917	0.5415

In Tab.7.4 sono mostrati il numero di iterazioni e il tempo impiegato per ogni prova, mentre in Tab.7.5 sono illustrati i valori minimi della funzione di costo nei vari casi.

Nella Tab.7.6 e nella Tab.7.7 sono illustrati i valori riguardanti le deviazioni standard della funzione e di ogni parametro in ogni ottimizzazione, relativi rispettivamente alla popolazione iniziale al numero totale delle iterazioni svolte.

La **prima prova** ha raggiunto un valore della funzione di costo pari a 52.5925 dopo 2239 iterazioni, nelle Fig.7.2a e 7.2b sono illustrate rispettivamente le curve del

Tabella 7.4: Numero di iterazioni e tempo nelle 6 prove.

	Numero di iterazioni	*Tempo impiegato*
GA1	2239	3h 6m 23s
GA2	4719	6h 59m 39s
GA3	1600	3h 2m 36s
GA4	3280	6h 41m 46s
GA5	2800	4h 4m 2s
GA6	3340	7h 16m 15s

Tabella 7.5: Valore minimo raggiunto dalla funzione di costo nelle 6 ottimizzazioni.

	Valore minimo della funzione di costo
GA1	52.5925
GA2	49.1543
GA3	57.1374
GA4	30.2127
GA5	44.2665
GA6	30.5692

Normalized flow stress e del*R-value* ottenute con i parametri ottimizzati con GA1, confrontate con quelle ottenute con i dati di riferimento.

Si nota come, per quanto riguarda la tensione, che le due curve sono molto vicine, infatti la distanza massima dal riferimento è di circa l'1%, mentre c'è uno scarto massimo pari a 1.5 relativo al valore del coefficiente di anisotropia R.

Nelle Fig.7.3a e 7.3b sono mostrati gli andamenti della funzione di costo, rispettivamente dal punto di vista temporale e dal punto di vista della sua mimizzazione.

La **seconda prova** è quella che doveva creare il maggior numero di generazioni e di conseguenza è quella che ha svolto il numero più alto di iterazioni, 4719, per arrivare ad una funzione di costo pari a 49.1543. Dal punto di vista della validità fisica, come mostrano le Fig.7.4a e 7.4b, la distanza dal riferimento è circa del 22% per il *Nomrmalized flow stress* e di 1.4 per quanto riguarda l'*R-value*.

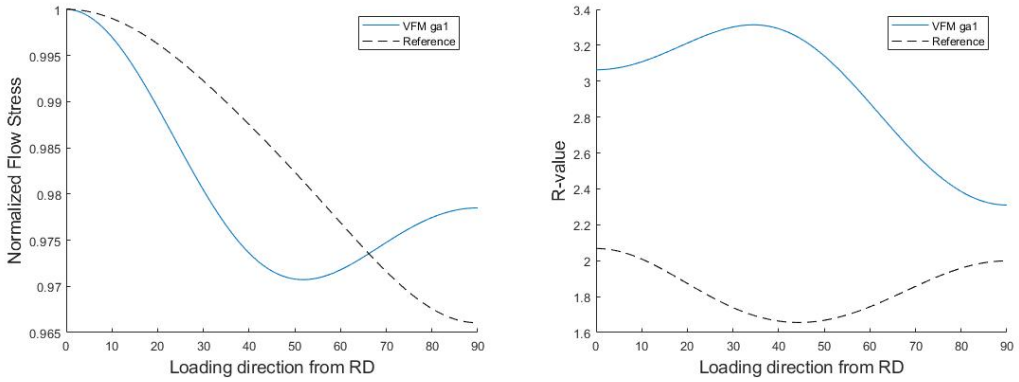
Nelle Fig.7.5a e 7.5b sono illustrati i grafici della funzione di costo rispetto alle

Tabella 7.6: Valori della deviazione standard della funzione di costo e dei parametri nelle sei prove, relativi alla popolazione iniziale.

	Funz. di costo	α_1	α_2	α_3	α_4	α_5	α_6	α_7	α_8
GA1	249.33	0.79	0.78	0.74	0.77	0.77	0.71	0.70	0.78
GA2	145.09	0.81	0.76	0.70	0.55	0.76	0.73	0.63	0.73
GA3	242.3	0.85	0.78	0.79	0.76	0.75	0.74	0.78	0.81
GA4	277.64	0.80	0.82	0.77	0.80	0.76	0.73	0.76	0.81
GA5	280.67	0.72	0.76	0.72	0.80	0.78	0.77	0.73	0.72
GA6	137.54	0.72	0.68	0.58	0.45	0.54	0.79	0.71	0.56

Tabella 7.7: Valori della deviazione standard della funzione di costo e dei parametri nelle sei prove, relativi a tutte le iterazioni.

	Funz. di costo	α_1	α_2	α_3	α_4	α_5	α_6	α_7	α_8
GA1	230.61	0.74	0.71	0.72	0.74	0.73	0.73	0.63	0.80
GA2	125.68	0.48	0.39	0.38	0.23	0.36	0.31	0.32	0.43
GA3	173.79	0.63	0.66	0.71	0.45	0.51	0.59	0.67	0.68
GA4	175.35	0.57	0.50	0.55	0.42	0.42	0.48	0.54	0.62
GA5	201.95	0.43	0.58	0.64	0.43	0.47	0.56	0.53	0.57
GA6	188.40	0.38	0.46	0.43	0.33	0.36	0.49	0.51	0.67



(a) Curva del *Normalized flow stress* GA1.

(b) Curva dell'*R-value* GA1.

Figura 7.2: Confronto tra le curve ottenute con GA1 e quelle di riferimento.

iterazioni, dal punto di vista temporale e decrescente.

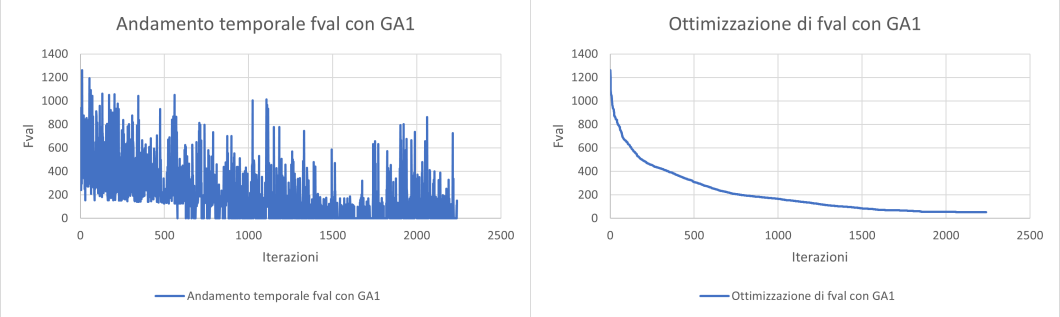
La **terza prova** è quella caratterizzata del più alto tasso di crossover, 0.95, ed ha portato al peggior valore della funzione di costo, 57.1374. Inoltre è GA3 è l'algoritmo che ha impiegato meno tempo, 3h 2m 36s, e meno iterazioni, 1600, per convergere verso quello che sembra, a tutti gli effetti, un minimo locale.

Questa prova è anche quella che ha dato i valori fisici peggiori, infatti c'è una distanza massima dal riferimento del 36% sulla tensione e una differenza molto ampia per quanto riguarda R, Fig.7.6a e 7.6b. Inoltre le due curve ottenute hanno un andamento totalmente incomparabile con quelle di riferimento. Queste evidenze portano a concludere che il risultato ottenuto è inaccettabile, sia dal punto di vista fisico, sia dal punto di vista numerico.

Infine gli andamenti della funzione di costo sono mostrati nelle Fig.7.7a e 7.7b.

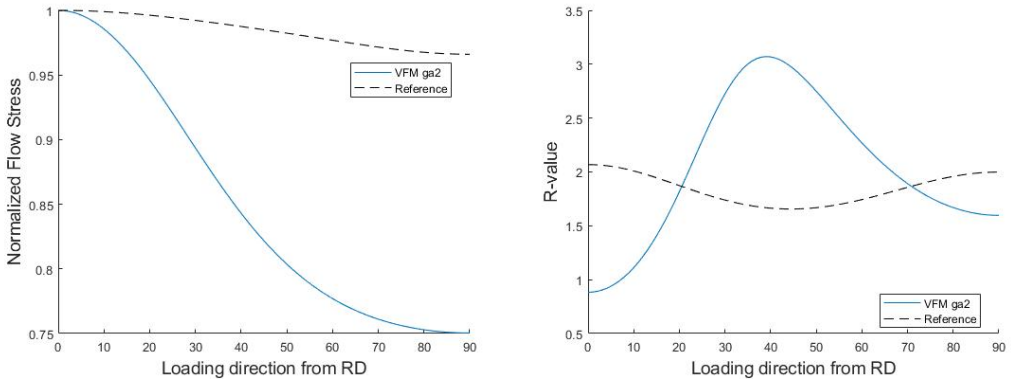
La **quarta prova** è quella che ha portato al minor valore della funzione di costo, 30.2127, dopo 6h, 41m e 46s, con 3280 iterazioni.

La curva del *Normalized flow stress* ha un errore massimo del 11%, Fig.7.8a, e quella dell'*R-value* si distanzia da quella di riferimento di un valore massimo pari a circa 1.7, Fig.7.8b. Inoltre si nota come, in particolare nel secondo caso, le curve abbiano un andamento molto simile rispetto a quelle ottenute con i parametri di



(a) Andamento temporale funzione di costo GA1. (b) Minimizzazione della funzione di costo GA1.

Figura 7.3: Andamenti della funzione di costo rispetto alle iterazioni con GA1.



(a) Curva del *Normalized flow stress* GA2. (b) Curva dell'*R-value* GA2.

Figura 7.4: Confronto tra le curve ottenute con GA2 e quelle di riferimento.

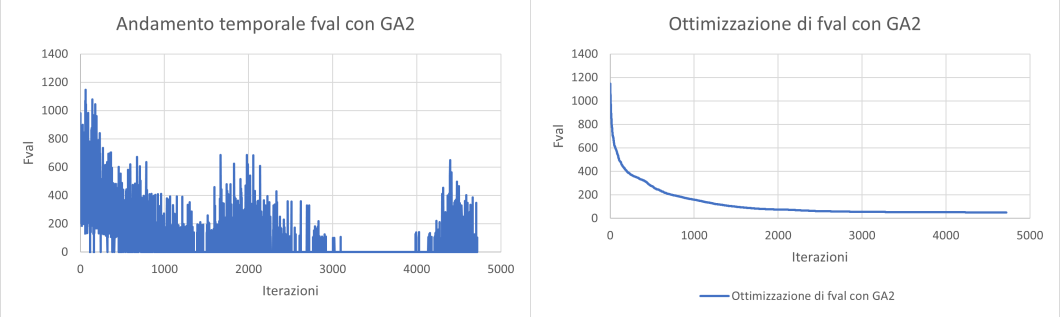
riferimento.

Nelle Fig.7.9a e 7.9b sono illustrati i grafici della funzione di costo rispetto alle iterazioni, dal punto di vista temporale e decrescente.

Con la **quinta prova** si è ottenuto un valore minimo della funzione di costo pari a 44.2665, dopo 2800 iterazioni. Dal punto di vista della validità fisica, la curva della tensione normalizzata, Fig.7.10a, si distanzia da quella di riferimento di un valore massimo di circa il 14%, mentre l'errore massimo di quella relativa al valore R è pari a 2.5, Fig.7.10b.

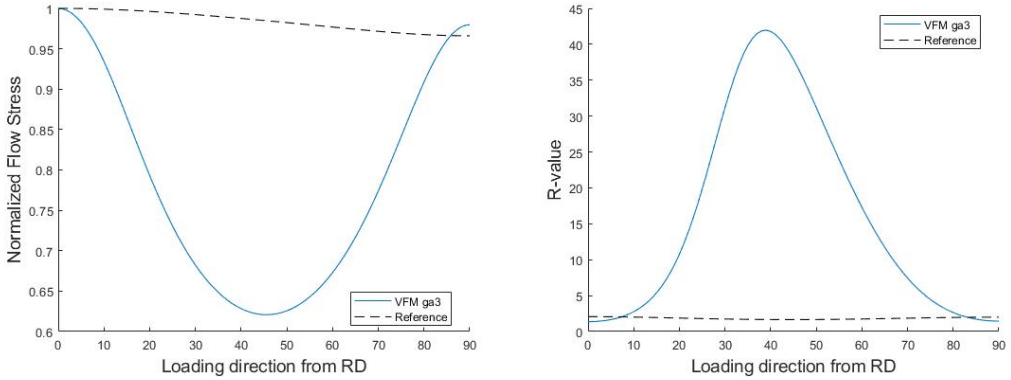
Gli andamenti della funzione di costo ottenuti con GA5 sono mostrati nelle Fig7.11a e 7.11b .

Infine, la **sesta prova** ha ottenuto il secondo miglior valore della funzione di costo, 30.5691, con il maggior tempo di cacolo di tutte le sei prove, 7h, 16m e 15s. La curva del *Normalized flow stress*, Fig.7.12a, ha un errore massimo rispetto a quella di riferimento del 8%, mentre quella dell'*R-value*, Fig.7.12b, si distanzia di un valore massimo pari a 1.5. Inoltre, specialmente quella relativa al coefficiente di anisotropia, le due curve seguono un andamento coerente, rispetto a quello del riferimento.



(a) Andamento temporale funzione di costo GA2. (b) Minimizzazione della funzione di costo GA2.

Figura 7.5: Andamenti della funzione di costo rispetto alle iterazioni con GA2.



(a) Curva del *Normalized flow stress* GA3. (b) Curva dell'*R-value* GA3.

Figura 7.6: Confronto tra le curve ottenute con GA3 e quelle di riferimento.

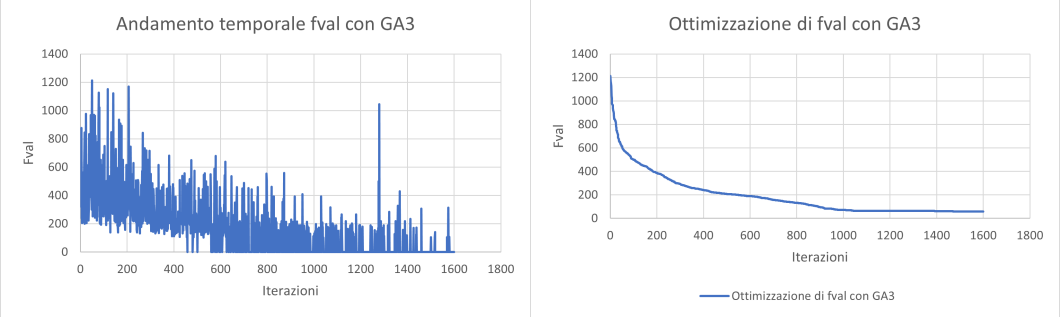
Nelle Fig.7.13a e 7.13b sono illustrati rispettivamente i grafici della funzione di costo, ottenuti con GA6, rispetto alle iterazioni, dal punto di vista temporale e decrescente.

7.5 Discussione dei risultati

I valori più bassi della funzione di costo sono stati ottenuti con il GA4 e il GA5, che rispettivamente hanno minimizzato fino a 30.2127 e 30.5691. Per quanto riguarda il tempo di calcolo, la prova che ne ha impiegato di meno è stata la terza, con 3h 2m 36s.

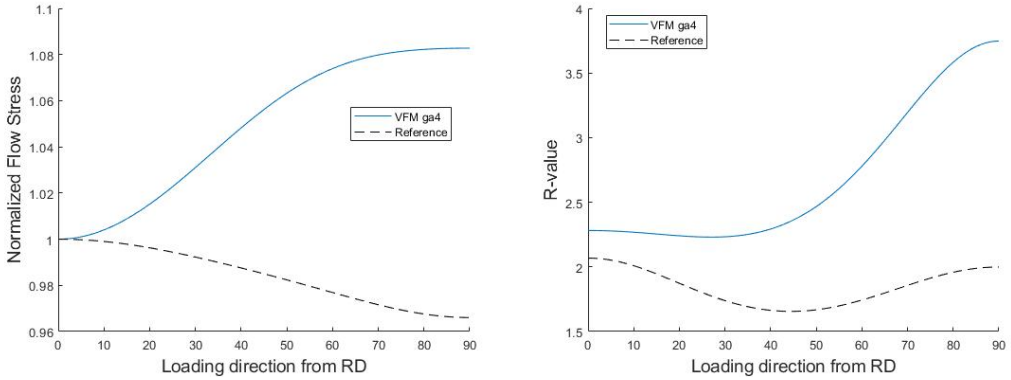
In particolare il GA3 ha anche ottenuto il valore più alto della funzione di costo, 57.1374, ed è la prova che, dal punto di vista della validità fisica, si distanzia maggiormente dalle curve di riferimento.

L'algoritmo che ha svolto il numero maggiore di iterazioni, 4719, è stato il GA2, cio è in linea con il fatto che fosse quello con pù generazioni da creare, 80.



(a) Andamento temporale funzione di costo GA3. (b) Minimizzazione della funzione di costo GA3.

Figura 7.7: Andamenti della funzione di costo rispetto alle iterazioni con GA3.



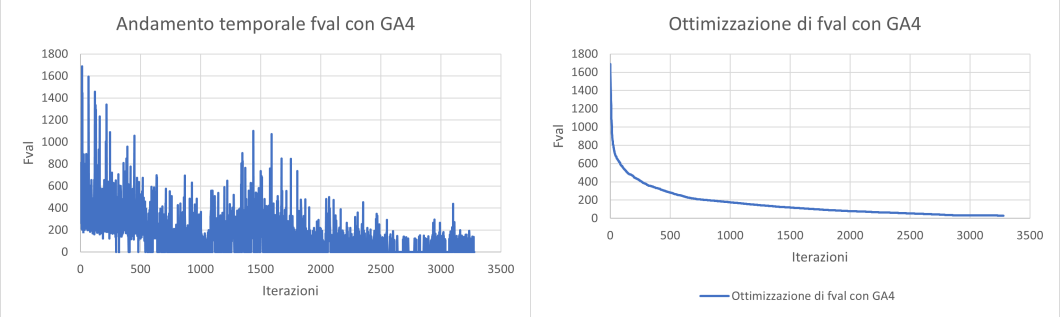
(a) Curva del *Normalized flow stress* GA4. (b) Curva dell'*R-value* GA4.

Figura 7.8: Confronto tra le curve ottenute con GA4 e quelle di riferimento.

La prova che ha impiegato il maggior tempo per arrivare a convergenza è stata la sesta, con 7h 15m 16s, che sorprendentemente non è quella con il numero maggiore di generazioni, ma quella caratterizzata dal tasso di crossover più basso, 0.5. Questo può essere spiegato con il fatto che, non applicando un tasso troppo eccessivo, l'algoritmo è molto più stabile e difficilmente cade in qualche minimo locale.

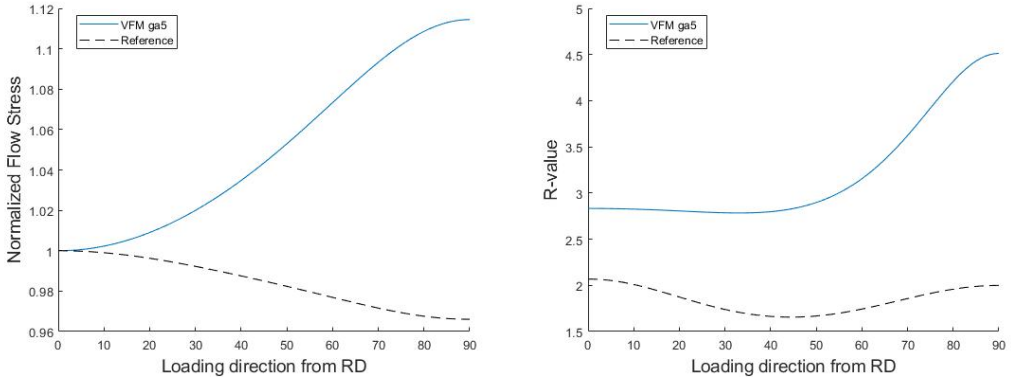
Dal punto di vista della validità fisica dei parametri ottenuti, la prova che meglio si adatta ad entrambe le curve di riferimento è la prima. Il GA1, però, non è l'algoritmo che raggiunge il valore più basso della funzione di costo. Questo caso mostra chiaramente come i metodi di ottimizzazione lavorino esclusivamente sul significato matematico dei parametri, mentre il controllo su quello fisico è compito delll'utente.

Per quanto riguarda le deviazioni standard, dai risultati si nota come, in tutte le prove effettuate e relativamente a tutti i parametri, i valori siano maggiori nel caso del calcolo sulla popolazione iniziale, rispetto a quelli relativi a tutte le iterazioni svolte. Questo risultato è in linea con la logica che c'è dietro questo tipo di algoritmo. Infatti il GA tende a differenziare quanto più la popolazione di partenza, così da



(a) Andamento temporale funzione di costo GA4. (b) Minimizzazione della funzione di costo GA4.

Figura 7.9: Andamenti della funzione di costo rispetto alle iterazioni con GA4.



(a) Curva del *Normalized flow stress* GA5. (b) Curva dell'*R-value* GA5.

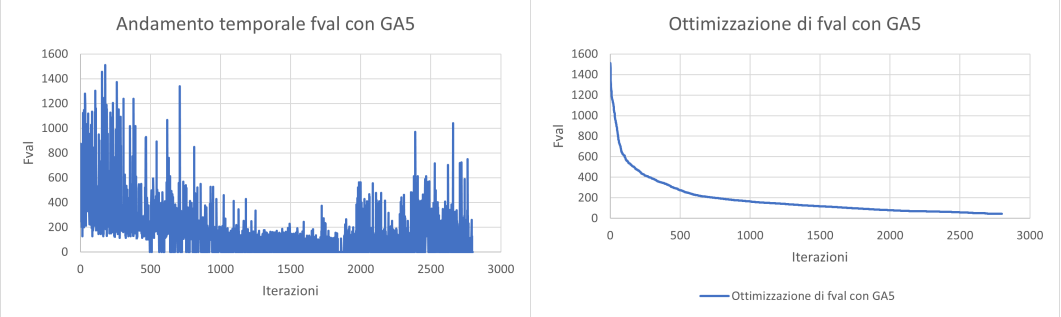
Figura 7.10: Confronto tra le curve ottenute con GA5 e quelle di riferimento.

renderla eterogenea, per esplorare il più possibile lo spazio delle soluzioni e ridurre le probabilità di convergere verso minimi locali.

La prova caratterizzata dai valori più bassi della deviazione standard, in entrambi i casi, è la sesta, in accordo col fatto che fosse quella con il tasso di crossover più basso, quindi una larga fetta degli individui genitori è rimasta intatta nelle generazioni successive. Invece quella caratterizzata dai valori più alti è la prima prova, cioè quella che ha creato il maggior numero di generazioni.

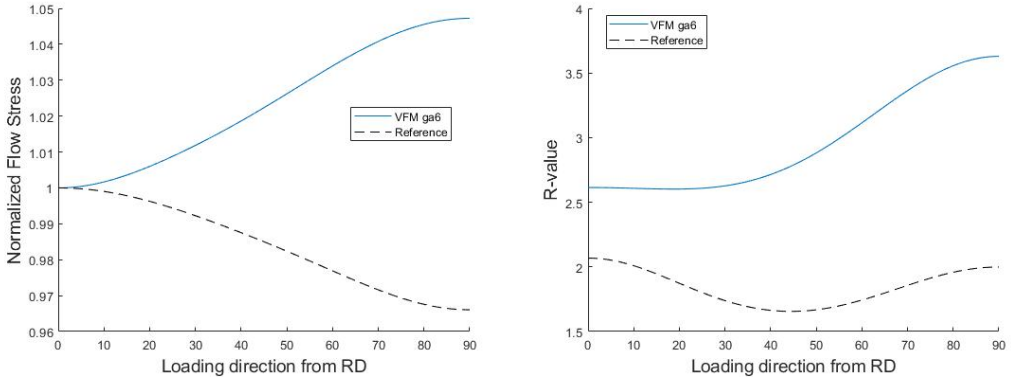
Dai risultati ottenuti si può notare come le prove caratterizzate dai più bassi tassi di crossover, cioè GA4, GA5 e GA6, sono quelle che hanno ottenuti i migliori risultati in termini della funzione di costo, invece la terza ottimizzazione con tale valore pari a 0.95 ha dato risultati decisamente negativi, sia dal punto di vista fisico, che numerico.

Questo dimostra che l'operatore genetico di crossover, pur essendo una componente fondamentale nella struttura dell'algoritmo genetico, perchè permette di rimescolare il patrimonio genetico delle varie soluzioni, se viene usato con probabilità troppo alte ne compromette la stabilità e lo conduce inevitabilmente a terminare il processo in poco tempo e verso minimi locali.



(a) Andamento temporale funzione di costo GA5. (b) Minimizzazione della funzione di costo GA5.

Figura 7.11: Andamenti della funzione di costo rispetto alle iterazioni con GA5.



(a) Curva del *Normalized flow stress* GA6. (b) Curva dell'*R-value* GA6.

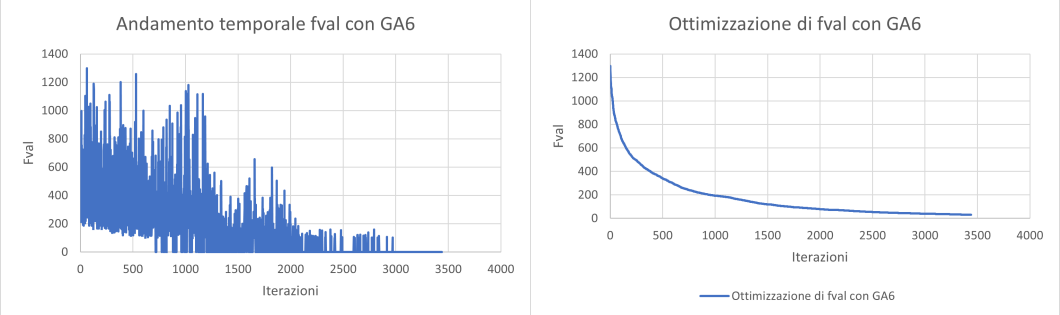
Figura 7.12: Confronto tra le curve ottenute con GA6 e quelle di riferimento.

Per quanto riguarda il metodo di selezione, invece, il confronto tra la quarta e la quinta prova dà una serie di indicazioni interessanti. Infatti il GA4 e il GA5 differiscono solamente per questo parametro e quindi paragonandone i risultati si possono vedere i vantaggi di un metodo rispetto all'altro. In particolare si nota come l'algoritmo che sfrutta il metodo *stochunif* arriva a convergenza in meno iterazioni, meno tempo e dà un risultato migliore dal punto di vista della funzione di costo e della validità fisica, rispetto a quello che utilizza il metodo *roulette*.

7.5.1 Confronto con l'algoritmo *Levenberg-Marquardt*

Infine sono stati confrontati i risultati ottenuti tramite le sei diverse prove con algoritmo genetico, con quelli ottenuti, nello stesso problema, tramite l'utilizzo del *Levenberg-Marquardt*, illustrati in [28]. Quest'ultimo algoritmo è implentato in *Matlab* tramite un altro metodo di ottimizzazione, chiamato *Lsqnonlin*.

I risultati ottenuti tramite *Lsqnonlin* sono mostrati in Tab.7.8, mentre i confronti con le curve di riferimento sono in Fig.7.14a e in Fig.7.14b.



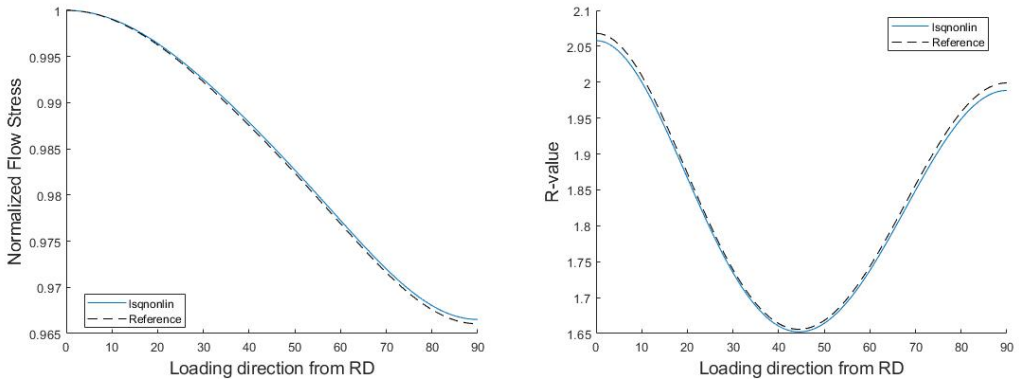
(a) Andamento temporale funzione di costo GA6. (b) Minimizzazione della funzione di costo GA6.

Figura 7.13: Andamenti della funzione di costo rispetto alle iterazioni con GA6.

Tabella 7.8: Risultati ottimizzazione tramite *lsqnonlin*.

	Funz. di costo	*Iterazioni*	*Tempo*
Lsqnonlin	0.0155	720	19m

Si può notare come l'algoritmo *Levenberg-Marquardt* ottenga risultati migliori sia dal punto di vista della funzione di costo, che in quello della validità fisica, per di più con un minor numero di iterazioni e un minor tempo di calcolo.



(a) Curva del *Normalized flow stress* Lsqnonlin. (b) Curva dell'*R-value* Lsqnonlin.

Figura 7.14: Confronto tra le curve ottenute con Lsqnonlin e quelle di riferimento.

Capitolo 8

Conclusioni

Nei capitoli precendenti è stato illustrato il principio di funzionamento degli algoritmi genetici, la loro applicazione nel problema dell'identificazione dei parametri costitutivi del materiale all'interno di un modello, il confronto con il modo di operare di altri algoritmi e dei metodi per combinare i vantaggi di uno con quelli dell'altro.

Il più grande vantaggio degli algoritmi genetici è quello di non dipendere dai parametri di partenza, poichè sono di base stocastica e non deterministica. Inoltre sono ottimi ricercatori di minimi globali e hanno una bassa probablità di convergere verso quelli locali.

Invece è noto come, gli algoritmi che sfruttano le informazioni sul gradiente, hanno una forte dipendenza dai parametri di input e hanno grandi capacità di ricerca del minimo in uno spazio ristretto. Un altro fattore che non è a favore di quelli a gradiente riguarda il loro modo di operare, infatti, in un problema in cui vanno ottimizzati molti parametri, la procedura di calcolo delle derivate può risultare macchinosa e difficoltosa.

Tuttavia, come si può notare di risultati delle prove sperimentali nel Cap.7, gli algoritmi genetici necessitano di un gran numero di iterazioni e di un tempo di calcolo molto più elevato rispetto a quello a gradiente in questione, il *Levenberg-Marquardt*. Inoltre i risultati ottenuti con i GA sono nettamente di qualità peggiore, sia in termini di valore della funzione di costo raggiunto, sia nel confronto sulla validità fisica con i dati di riferimento, rispetto a quelli ottenuti con il *Levenberg-Marquardt*.

Le prove sperimentali sono state svolte variando le opzioni disponibili in *Matlab* e si può notare come, alzare troppo il tasso di crossover, sia contropuducente e renda l'algoritmo instabile a tal punto da farlo convergere ad un minimo locale (GA3). Inoltre la terza prova si rivela essere anche quella che si distanzia maggiormente dal riferimento, sia per tensione in campo plastico, sia per coefficiente di anisotropia.

Dal Cap.7 risulta come il metodo di selezione influenzi la prestazione dell'algoritmo genetico. Infatti, dal confronto tra la quarta e la quinta prova, si evince che il metodo *stochunif* porta a risultati migliori in tutti i campi, in un minor tempo di calcolo e minor numero di iterazioni, rispetto al metodo *a roulette*.

Inoltre dal punto di vista fisico, i risultati che più si avvicinano alle curve di riferimento, sono quelli ottenuti dall'algoritmo con il numero maggiore di generazioni da creare, il che rende questo fattore la base per una buona ottimizzazione.

Dai risultati illustrati si può quindi concludere che l'algoritmo genetico dipende fortemente dalle varie opzioni che vengono impostate dall'utente e che, mettendone anche solo una non corretta ai fini dell'ottimizzazione, come ad esempio il tasso di crossover pari a 0.95 nel GA3, viene inevitabilmente compromessa la prestazione dell'algoritmo e si ottengono risultati non accettabili, dal punto di vista fisico e numerico. Invece l'algoritmo a gradiente, avendo un numero inferiore di varianti in base alle opzioni inserite, risulta molto più affidabile.

Infine si può affermare che, la metodologia più interessante consiste in quella di usare entrambi gli algoritmi, combinati in uno *ibrido*, in modo da sfruttare i vantaggi che caratterizzano uno e l'altro e minimizzare gli svantaggi di entrambi, come illustrato nel Cap.5.

Bibliografia

[1] A G. B. Tettamanzi. Algoritmi evolutivi: concetti e applicazioni. *Mondo digitale 1*, 2005.

[2] G E P Box. Evolutionary operation: A method for increasing industrial productivity. *Journal of the Royal Statistical Societ.*, 1957.

[3] H. J. Bremermann. The evolution of intelligence. the nervous system as a modelof its environment. technical report 1, contract no. 477(17). *Department of Mathematics, University of Washington, Seattle*, 1958.

[4] Walsh M.J. Fogel Lawrence J., Owens A. J. Artificial intelligence through simulated evolution. *John Wiley Sons, New York*, 1966.

[5] Holland John H. Adaptation in natural and artificialsystems. *University of Michigan Press,Ann Arbor*, 1975.

[6] Rechenberg I. Evolutionsstrategie: Optimierungtechnischer systeme nach prinzipiender biologischen evolution. *Frommann-Holzboog,Stoccarda*, 1973.

[7] J R Koza. Genetic programming: on the programming of computers by meansof natural selection,. *The MIT Press, Cambridge, Massachussets*, 1993.

[8] Ramos U. Leal, R. Algoritmos geneticos na optimizacao de compositos laminados. *Proceeding of the V Congreso de Metodos Numericos engenieria*, 2002.

[9] Savalle S. Agice Chaboche Jl, Nouailhas D. Logiciel pour l'identificationinteractive graphique des lois de comportement. *La Rech Aéros*, pages 3:59–76, 1991.

[10] Schnur; Zabaras. An inverse method for determining elastic materialproperties and a material interface. *Int J Numer Methods Eng*, 1992.

[11] ;A. Andrade-Campos R. de Carvalho ;R.A.F. Valente. Optimization strategies for non-linear material parameters identificationin metal forming problems. *Computers and Structures 89*, page 2, 2011.

[12] Ł. Kowalewski ; M. Gajewski. Assessment of optimization methods used to determine plasticityparameters based on dic and back calculation methods. *Cross Mark*, page 6, 2019.

[13] D.W. Marquardt. An algorithm for least-squares estimation of non-linear parameters. *J. Soc. Indust.Appl. Math 11*, pages 431–441, 1963.

[14] C. Fleury. First and second order convex approximation strategies in structural optimization. *Struct.Optim*, pages 1,3–10, 1989.

[15] K. Svanberg. The method of moving asymptotes. a new method for structural optimization. *Int. J. Numer.Meth. Engng*, pages 24, 359–373, 1987.

[16] K. Schittkowski. Nlpql: a fortran subroutine solving constrained nonlinear programmingproblems. *Ann. Oper. Res*, pages 485–500, 1985.

[17] Pappalettere C. Lamberti, L. Comparison of the numerical efficiency of different sequential linearprogramming based algorithms for structural optimisation problems. *Comput. Struct*, 2000.

[18] Gelatt C.D. Vecchi M.P. Kirkpatrick, S. Optimization by simulated annealing. *Science 4598 (220)*, pages 671–680, 1983.

[19] Timmis J. Castro, L. Artificial immune systems: A new computational approach. *Springer-Verlag, NewYork, Inc.*, 2002.

[20] Bergs T. ;Hardt M. ; Schraknepper D. Inverse material model parameter identification for metal cutting simulations by optimization strategies inverse material model parameter identification for metal cutting simulations by optimization strategies. *MM Science Journal*, 2019.

[21] F. Barlat;D.J. Lege; J.C. Brem. *International Journal of Plasticity 7*, pages 693–712, 1991.

[22] Barlat F; Brem JC.; Yonn JW.; Chung K.; Dick RE; Lege DJ. Plane stress yield function for alluminum alloy sheets-part1: theory. *Int J Plast*, pages 19: 1297–319, 2003.

[23] J. L. Chaboche ;G. Rousselier. On the plastic and viscoplastic equations. *J. Press. »es. echnol. ASME, 105*, pages 153–164, 1983.

[24] Furukawa T.; YAagawa G. Inelastic constitutive parameter identification using a evolutionary algorimhm with continuous individuals. *International journal for numerical methods in engineering, VOL. 40*, 1997.

[25] R. de Carvalho ;R.A.F. Valente; A. Andrade-Campos. Optimization strategies for non-linear material parameters identification in metal forming problems. *Computers and Structures 89*, 2010.

[26] B.M. Chaparro; S. Thuillier; L. F. Menezes; P.Y. Manach; J.V. Fernandes. Material parameters identification: Gradient-based, genetic and hybridoptimization algorithms. *Computational Materials Science 44*, 2008.

[27] Wu PD ;MacEwen SR ; Lloyd DJ ; Tugcu P ;Neale KW . On prestraining and the evolution of material anisotropy in sheet metals. *Int J Plast*, pages 21,723–39, 2005.

[28] A.Lattanzi ;F.Barlat; F Pierron; A Marek ; M Rossi. Inverse identification strategies for the characterization of transformation-based anisotropic plasticity models with the non-linear vfm. *International Journal of Mechanical Sciences 173*, 2020.

[29] Rossi M Pierron F . Marek A, Davis FM. Extension of the sensitivity-based virtual fields to large deformation anisotropic plasticity. *Int J Mater Form*, 2018.

[30] Yoon JW Chung K Dick RE Lege DJ Barlat F, Brem JC. Plane stress yield function for aluminum alloy sheets –part 1: theory. *Int J Plast*, 2003.