ALMA MATER STUDIORUM - UNIVERSITÀ DI BOLOGNA

SCUOLA DI INGEGNERIA E ARCHITETTURA
-Sede di Forlì-

CORSO DI LAUREA
IN INGEGNERIA AEROSPAZIALE
Classe: 10

ELABORATO FINALE DI LAUREA

In
COSTRUZIONI AERONAUTICHE

L'UTILIZZO DI ALGORITMI GENETICI NEL PROGETTO PRELIMINARE DI UN VELIVOLO

CANDIDATO

Christian Shikh Farshi

RELATORE:

Prof.ssa Ing. Sara Bagassi

CORRELATORE

Prof.ssa Ing. Francesca Lucchi

Anno Accademico 2012/13

Sessione I

Indice

Introduzione

Il progetto di un velivolo risulta essere un processo multidisciplinare molto complesso. Per poter determinare una configurazione di variabili che permetta di soddisfare i requisiti, che si desiderano ottenere dal velivolo, sono necessarie una serie di stime che richiedono altrettanti cicli di analisi delle caratteristiche, prima di poter ottenere una configurazione completa o accettabile. Il processo di progetto richiede, così, un gran numero di iterazioni per poter trovare la migliore configurazione.

Per sviluppare un progetto aeronautico risultano coinvolte un gran numero di discipline, ognuna limitata al proprio *set* di equazioni, di obiettivi da raggiungere e di vincoli imposti. Risulta così molto complesso riuscire a prevedere le conseguenze che dei singoli cambiamenti possano causare nell'intero progetto.

Metodi avanzati di analisi e codici per il dimensionamento sono stati sviluppati per definire un progetto aeronautico. Tutto ciò è stato ottenuto attraverso l'implementazione di codici esistenti per il dimensionamento con metodologie di ottimizzazione numerica.

In questo lavoro di tesi verranno descritti gli strumenti di ottimizzazione noti come algoritmi genetici e verrà presentato come questi possano essere inquadrati all'interno della fase preliminare del progetto di un velivolo.

Si è deciso di improntare il lavoro su queste metodologie di ricerca, che si sono sviluppate nel corso degli ultimi quaranta anni e che rappresentano ad oggi, seppur già molto studiate, un campo di indagine ancora aperto e di sicuro interesse soprattutto in ambito aeronautico.

Si inizierà con una trattazione, contenuta nel primo capitolo, in cui si ripercorrerà l'evoluzione storica del progetto in ambito aeronautico, dalle prime prove di volo fino ai tempi moderni ed alla suddivisione del progetto aeronautico stesso in varie fasi.

La trattazione continuerà con l'introduzione degli algoritmi genetici e delle varie metodologie di ottimizzazione, che vengono utilizzate fino a terminare il secondo capitolo con un confronto tra le varie tecniche e gli algoritmi genetici stessi.

All'interno del terzo capitolo verrà descritto più nel dettaglio come opera un algoritmo genetico mettendone in luce la struttura e dando una descrizione degli operatori di base di cui è composto. Il capitolo si concluderà con un esempio che permetterà di vedere nel dettaglio come operano gli algoritmi genetici.

Si procederà così alla descrizione della teoria matematica, che sta alla base del funzionamento degli algoritmi genetici, nota con il nome di Teoria degli Schemi o Teorema fondamentale degli Algoritmi Genetici.

Verrà conclusa la parte relativa a queste metodologie di ottimizzazione con la descrizione di alcuni problemi di interesse pratico e si introdurranno alcune modifiche alla struttura di base, che permettono di utilizzare gli algoritmi genetici per affrontare problemi molto diversi tra loro.

In conclusione verranno presentati e discussi esempi di applicazione di tali metodologie in ambito prettamente aeronautico il cui risultato rappresenta un possibile utilizzo degli algoritmi genetici proprio nella fase preliminare del progetto di un velivolo.

1. Il progetto preliminare di un velivolo

All'interno di questo capitolo si andrà a ripercorrere l'evoluzione storica dell'avamprogetto aeronautico, presentando i vari strumenti che, successivamente, si sono sviluppati ed utilizzati [1]. Si discuterà poi come, nel progetto preliminare, si possano considerare anche le scelte di tipo architettonico, mostrando come il problema sia stato recentemente affrontato con l'ausilio dei SW CAD-3D parametrici. Infine si presenterà come, in questo modo, l'avamprogetto sia diventato molto più dettagliato di quanto fosse in precedenza, portando così a prevedere, per analisi preliminari o di più alto livello di sistema, anche forme semplificate di '*pre-design*'. Verrà inoltre inquadrato in questa ottica l'utilizzo delle metodologie di ottimizzazione a cui appartengono gli algoritmi genetici.

1.1 L'avamprogetto e le sue caratteristiche

In campo aerospaziale la definizione di 'avamprogetto' o 'progetto preliminare' racchiude tutte quelle attività che vengono svolte nella fase iniziale di un progetto. Ovviamente i concetti di base possono essere generalizzati alla progettazione di un qualsiasi prodotto, di media-alta complessità, per i vari settori dell'Ingegneria.

Il primo passo della progettazione consiste nell'elaborare un'idea di base delle caratteristiche che deve avere il prodotto che si ha intenzione di realizzare. In generale si può riassumere questa prima fase nel modo seguente:

1) **definizione di massima della possibile architettura del prodotto**, ovvero scegliere, senza scendere nel dettaglio, quali forme di elementi adottare e mutue posizioni di tali elementi. Ognuna delle alternative, aggregate in più combinazioni, inizialmente si presenta potenzialmente in grado di essere adottata. Ad esempio per realizzare il 'sistema propulsivo' si tratta di scegliere quale tipo di propulsore utilizzare, in che numero ed in quale posizione;
2) **definizione indicativa**, ovvero con un margine di errore compreso tra il 10% ed 15%, delle caratteristiche quantificabili di maggior rilevanza del prodotto che si vuole realizzare come dimensioni, pesi e prestazioni che si vogliono ottenere.

Ad esempio, per un velivolo da trasporto passeggeri di classe ‘media’, le scelte dell’architettura in fase di avamprogetto potrebbero fornire dei dati del tipo:

- lunghezza = 50m;
- diametro massimo della fusoliera = 5.5m;
- allungamento alare = 7.5;
- carico alare = 700 kg/m^2;
- spinta massima = 2 x 26.000 kg = 52.000 kg
- peso massimo al decollo = 180 ton;
- peso a vuoto operativo = 105 ton;
- peso del combustibile = 155 ton;
- velocità massima = 860 km/h;
- lunghezza decollo = 1700 m.

A questo livello di approssimazione un progetto non è certamente pronto per essere utilizzato in produzione, fase a cui si può giungere solo dopo una serie di estesi affinamenti e approfondimenti che richiedono l’utilizzo di ingenti risorse umane, temporali e ovviamente materiali per procedere a prove sperimentali e prototipazione che richiedono un onere economico molto più elevato rispetto all’avamprogetto stesso.

Un simile impiego di risorse è industrialmente proponibile solo a valle di una conseguita convinzione che il futuro prodotto sia fattibile e competitivo all’interno del settore di mercato che andrà ad occupare, o comunque accettato da un committente, come ad esempio avviene in ambito militare.

Proprio per queste ragioni in molti casi i termini ‘studio di fattibilità’ o ‘fase di fattibilità’ sono sinonimi, o quasi, di ‘avamprogetto’.

Alla luce di queste considerazioni si può osservare come l’avamprogetto, il cui risultato potrebbe essere inteso come ‘progetto non costruibile ma valutabile e confrontabile’, spicca per la sua rilevanza soprattutto in considerazione del ruolo chiave che tale fase occupa per decisioni critiche quali iniziare o meno un’onerosa attività di sviluppo.

L’avamprogetto, costituisce inoltre, il punto di partenza di un’eventuale attività di sviluppo, per cui scelte caratterizzanti il prodotto futuro, definite appunto in

avamprogetto, quali architettura di base e definizioni quantitative delle caratteristiche basilari, resteranno le linee guida per tutto il progetto, a meno di eventuali riprogettazioni.

In sostanza si può affermare che già dalla fase preliminare di progetto, il futuro prodotto viene in gran parte caratterizzato in termini di prestazioni, costi, etc. (figura 1).

Le maggiori difficoltà che si incontrano in questa fase, oltre alle criticità già messe in luce, sono essenzialmente quelle di costruire modelli matematici affidabili in un contesto, come già detto, poco definito e, soprattutto in ambito aerospaziale, di elevata complessità.

Per una visione più teorica delle problematiche relative all'avamprogetto dei velivoli si rimanda al lavoro di E.Antona [2].

Figura 1: Fasi di sviluppo del progetto [1].

1.2 Stato dell'arte

Per quanto esposto nel paragrafo precedente risulta interessante considerare l'evoluzione che l'attività di avamprogetto ha registrato nel corso degli anni, con particolare riferimento alle nuove forme ed esigenze che si sono manifestate e agli strumenti che si sono potuti utilizzare in campo aerospaziale negli ultimi anni.

1.2.1 Primi anni dello sviluppo aeronautico

Nei primi anni del Novecento che videro la realizzazione dei primi voli la fase di avamprogetto non era ancora sviluppata come descritto nel paragrafo precedente.

I progettisti, per lasciarsi guidare nel progetto di un velivolo, si affidavano molto più a prove sperimentali, alla costruzione di prototipi e a prove più o meno empiriche. La caratteristica base della progettazione aeronautica dei primi anni era quella di essere condotta con un impiego di risorse umane che era rappresentato dal singolo progettista e da un ridotto numero di collaboratori, generalmente disegnatori. A differenza di quanto avviene oggi l'impiego di 'ore/uomo' risultava circa costante nel tempo togliendo così il significato di avere una separazione tra fasi successive del progetto.

Agli inizi dello sviluppo dell'aeronautica il vero e proprio punto decisionale era rappresentato dalla scelta di procedere o meno alla produzione di serie sulla base dei risultati offerti dai vari prototipi.

1.2.2 Periodo tra le due guerre

Dalla metà degli anni '20, dopo un periodo di profonda crisi a seguito della fine del primo conflitto mondiale, la tecnica aeronautica progredì molto velocemente; agirono fortemente l'impresa di Lindberg, le competizioni come la coppa Schneider, le trasvolate di vere e proprie flotte di velivoli, ma soprattutto la carica innovativa portata da grandi progettisti e grandi scienziati. Il risultato fu quello di ottenere velivoli da trasporto sufficientemente sicuri ed efficienti verso la fine degli anni '30, tanto da permettere i primi regolari servizi commerciali. In ambito militare, accanto a macchine di concezione obsoleta si preannunciava l'avvento dei primi bombardieri strategici e dei primi caccia monoplani che risulteranno i protagonisti del conflitto che era imminente.

I principali risultati ottenuti in questo periodo di forte evoluzione possono essere così riassunti:

- utilizzo dell'ala monoplana a sbalzo;
- l'adozione di profili alari biconvessi NACA e successivamente 'laminari';
- comprensione delle problematiche aerodinamiche tra cui numero di Reynolds, primi modelli vorticosi delle superfici portati e primi metodi di stima delle distribuzioni di portanza;
- carrelli retrattili;

- introduzione di dispositivi di ipersostentazione sempre più efficienti;
- sviluppo di nuovi concetti strutturali poi culminati nelle strutture a semi-guscio;
- introduzione dei materiali metallici in sostituzione al legno;
- sviluppo della tecnologia dei rivetti a testa ribassata;
- progresso in ambito propulsivo comprensivo delle eliche a passo variabile;
- inizio dello sviluppo dei primi sistemi di bordo e delle componenti impiantistiche.

Di pari passo con questi sviluppi tecnici si iniziava a tener conto delle prime Normative che guidavano il progettista nelle sue scelte.

Tuttavia lo schema organizzativo restava quello dei primi anni di sviluppo con un progettista che prendeva scelte globali e pochi collaboratori che in generale erano dei disegnatori.

1.2.3 Nascita delle varie fasi di progetto.

Durante il II conflitto mondiale la crescente complessità degli aeromobili e la necessità di non dilatare i tempi di sviluppo portarono ad un aumento dei progettisti impiegati, fermo restando l'autorità di progettista 'capo'. Inoltre l'approfondimento delle discipline specialistiche portò logicamente il responsabile del progetto ad avvalersi di un numero sempre più elevato di collaboratori specializzati nei vari settori, che lavoravano in parallelo nello sviluppo di uno stesso progetto.

Ciò rese praticamente irrinunciabile avere poco dopo l'avvio di un progetto, una configurazione di base su cui iniziare le attività degli specialisti.

Contemporaneamente l'ottimizzazione della produttività in ottica dello sforzo bellico richiesto dal conflitto mondiale cominciò a richiedere di non impegnare inutilmente risorse di progettazione e attività di realizzazione di prototipi su progetti non funzionali alle esigenze richieste, inoltre iniziò a nascere, per interesse dei clienti, la volontà di stimolare la competitività del sistema industriale presentando richieste ad un maggior numero di aziende.

Nacque così la necessità di sviluppare studi di avamprogetto su cui si poteva poi effettuare la scelta del solo progetto destinato ad essere sviluppato su cui concentrare risorse e finanziamenti.

Si giunse così ad una suddivisione delle varie attività di sviluppo con spunti di caratterizzazione delle varie fasi che con il tempo divennero uno standard da seguire (figura 2).

FASI DEL PROGETTO	PRELIMINARE CONCEPTUAL	DI MASSIMA PRELIMINARY	DETTAGLIO DETAILED
Risultati della fase	Progetti non costruttivi ma confrontabili. Elementi definiti a livello "scatole nere".	Verifiche prestazionali. Progettazione sotto-sistemi Emissione specifiche per forniture esterne.	▪ Disegni costruttivi. ▪ Studi di fabbricazione: cicli lavoro, progetto attrezzature produzione. ▪ Impostazione contratti.
Disegni	Tre viste, "spaccati"	Disegni di sezioni, gruppi, sotto-gruppi, particolari	Disegni costruttivi
Accuratezza della valutazione dei pesi	92%	97% → 98%	99%
Tempo occorrente	Da 1 a 4 settimane	Da 6 a 15 mesi	Da 1 a 5 anni (comprensivo Sperimentazione in volo (prolungam. di Fase 1))
Tipo grandezze definite	Superficie alare	numero e spaziature longheroni e centine alari	Per ogni giunzione numero, diametro e passo rivetti

Figura 2: Fasi di progetto [1].

1.2.4 L'avamprogetto moderno

La nascita dell'elaboratore elettronico portò significativi cambiamenti nello sviluppo della fase di avamprogetto.

Negli anni '70 e '80 l'avvento dei primi SW C.A.D. (*Computer Aided Design*), tendenzialmente bidimensionali, permise di iniziare a visualizzare il velivolo che si stava realizzando con un disegno in due o tre viste. Tuttavia non era ancora possibile, in quanto software non ancora parametrici, di riuscire ad integrare i complessi algoritmi che necessita la fase di avamprogetto.

Negli anni '80 non mancarono tentativi di una scelta ottimale e preliminare del layout, cui poi applicare un algoritmo di avamprogetto per dimensionarlo, se possibile, tenendo conto del layout scelto.

Da circa un decennio, però, grazie all'evoluzione delle tecnologie utilizzate, l'attività di avamprogetto è radicalmente mutata. Nuovi e sempre più potenti

software CAD tridimensionali e parametrici hanno iniziato ad inserirsi sempre di più nelle varie fasi di progetto.

Limitandoci alla fase di avamprogetto per così dire 'classico', l'attività di studio dell'architettura del velivolo era relegata ad un ruolo modesto e complementare lasciando la definizione della configurazione del velivolo alle successive attività specialistiche di aerodinamica e strutture oppure ad un'attività di disegno a posteriori.

Come già visto in precedenza, un'attività di progettazione, ossia di definizione e sviluppo di un nuovo prodotto, è sempre basata su successive fasi di scelte qualitative e di modellizzazioni per riuscire ad ottenere dei valori matematici; in sostanza si tratta di una successione, più o meno lunga di cicli tipo quelli della figura 3.

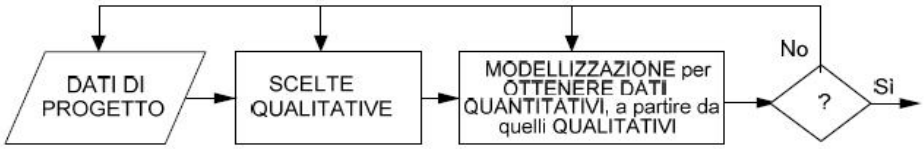

Figura 3: Ciclo di progetto.

Da sempre lo strumento più utilizzato per effettuare scelte qualitative è il disegno. Tuttavia ve ne sono altri, sviluppati più recentemente, come ad esempio i prototipi, ottenuti automaticamente attraverso le tecniche di *Rapid Prototyping*.

La modellizzazione più comoda per ottenere dati quantitativi a partire da quelli qualitativi, risulta essere da sempre quella matematica integrabile o sostituibile con modelli fisici di similitudine.

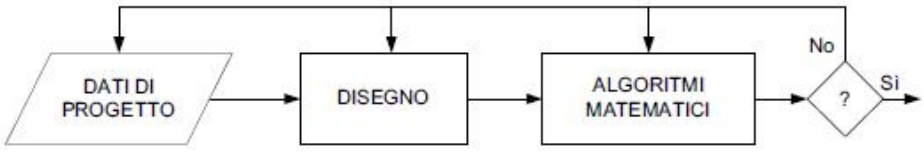

Figura 4: Strumenti dell'avamprogetto.

Tuttavia in attività molto preliminari o per definire progetti molto semplici il ciclo mostrato in figura 4 può ridursi a quello della figura 5.

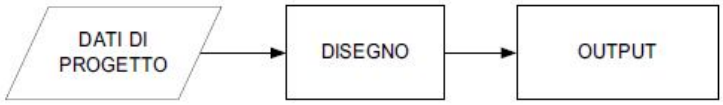

Figura 5: Ciclo di progetto di studi preliminari o nel caso di prodotti molto semplici.

Quanto detto mette in luce come l'avamprogetto classico lasciava alle fasi successive l'onere di elaborare i possibili layout. L'avvento e l'evoluzione del computer portarono una forte innovazione nell'integrazione del disegno nelle fasi iniziali di progetto.

Oggi il problema della valutazione della configurazione architettonica in avamprogetto è risolvibile integrando un SW CAD-3D parametrico con i tradizionali algoritmi di calcolo. I vantaggi portati da questi nuovi strumenti possono essere elencati come segue:

- creazione di un modello solido 3D più rappresentativo di un disegno bidimensionale;
- possibilità di ottenere proiezioni ortogonali e prospettiche del modello che può essere fatto ruotare attorno ai tre assi cartesiani e che può essere scalato e che viene rappresentato con effetti di profondità con un'ottima simulazione visiva del modello solido;
- possibilità di misurare quote sul modello stesso, valutare volumi del modello solido creato e messa in tavola di sezioni;
- capacità di trasmettere il modello virtuale a macchine utensili per la realizzazione diretta di un modello reale oppure a macchine di prototipazione rapida per lo sviluppo di prototipi in scala;
- possibilità di rendere il modello parametrico, consentendo la variazione di opportune grandezze e di mutare le dimensioni del modello solido in seguito all'aggiornamento dei dati.

1.3 Metodologie per l'avamprogetto

Nella fase di avamprogetto si cerca di ottenere una prima valutazione delle caratteristiche del velivolo in funzione dei requisiti richiesti.

Tenendo presente che l'aeroplano è la macchina sulle cui caratteristiche globali, più che su ogni altra, il peso ha un'influenza estremamente condizionante ne consegue la necessità di una sua stima sin dalle prime fasi di sviluppo del progetto.

Risulta interessante notare come fin dagli albori dell'avamprogetto moderno (nel 1939 negli U.S.A. fu fondata la SAWE - *Society of Allied Weight Engineers*) due furono le vie considerate: quella statistica e quella analitica.

Raccogliendo i dati di velivoli esistenti la SAWE, nonché vari altri studiosi in altri paesi, cercarono di correlare questi pesi in funzione dei parametri tecnici gestiti dai metodi di avamprogetto. Nella stessa ottica vennero ricavati su base statistica anche due set di WERs (*Weight Estimation Relationships*) che negli studi di metodologie computerizzate di avamprogetto sono stati e sono tuttora utilizzati: il primo dovuto a R.N.Staton [3] per velivoli da combattimento ed il secondo dovuto a M.N.Beltrano e M.Morris [4] valido per velivoli da trasporto in grado di stimare persino il costo della realizzazione dei vari sottosistemi attraverso procedure di CERs (*Cost Estimation Relationships*).

La via analitica utilizzò approcci come quello con cui G.Gabrielli [5, 6] ricavò espressioni del peso teorico di ala e fusoliera.

Avendo così acquisita la capacità di stima dei pesi per avere un quadro completo si deve aggiungere la disponibilità di formule empiriche di stima di caratteristiche aerodinamiche (dedotte per via statistica in 'Handbooks' quali i 'Data-sheets' inglesi o i 'DATCOM' statunitensi) con l'utilizzo di relazioni di meccanica del volo, in genere di tipo prestazionale, per la valutazione di massima delle prestazioni di un velivolo.

Resta così da discutere come integrare relazioni di questo tipo al fine di riuscire a sintetizzarle in fase di avamprogetto.

Esistono numerose metodologie di avamprogetto, come ad esempio quella rappresentata dall'analisi di configurazione adottata da G.Gabrielli [7] che considera un numero di equazioni pari a quello delle incognite.

Molte altre procedure di relazione tra stima dei pesi iniziale e design preliminare sono state sviluppate come ad esempio quelle proposte da J.Roskam [8] o da E.Torenbeek [9].

Un'ulteriore metodologia di avamprogetto che è stata ed è tuttora molto utilizzata è rappresentata dalle analisi parametriche [2] che utilizzano metodologie quali area di compatibilità e del 'tappeto' di soluzioni.

Appare evidente che le analisi parametriche possono essere condotte graficamente, il che ne spiega il successo anche quando l'uso del computer non era ancora diffuso. Tuttavia con l'avvento del computer si sono affermati, accanto alle analisi parametriche, i metodi di ottimizzazione.

Si rimanda al prossimo capitolo la descrizione delle metodologie di ottimizzazione per così dire più classiche mentre si vuole ora introdurre l'argomento chiave di questa tesi.

Infatti un'innovativa metodologia di ottimizzazione che ha trovato applicazione non solo in ambito aeronautico è rappresentata dai cosiddetti algoritmi genetici.

2. Gli Algoritmi Genetici

In questo capitolo verranno introdotte quelle metodologie di ricerca ed ottimizzazione che prendono il nome di algoritmi genetici: si andrà ora ad analizzare cosa siano, la loro origine e le differenze sostanziali con le altre procedure di ricerca. Verrà inoltre illustrato come un semplice algoritmo genetico possa presentare un grado di robustezza superiore rispetto alle tecniche classiche di ottimizzazione.

2.1 Introduzione

Gli algoritmi genetici sono una famiglia di tecniche di ottimizzazione che si ispirano alla teoria della selezione naturale di Charles Darwin che regola l'evoluzione biologica.

I sistemi biologici esibiscono un alto grado di robustezza ed efficienza nel risolvere una vasta serie di problemi essenziali per la loro sopravvivenza, ma, al contrario dei sistemi artificiali creati dall'uomo, essi non sono il frutto di un disegno progettuale, bensì sono il risultato di un processo evolutivo basato sulla riproduzione degli individui migliori, sulla ricombinazione genetica dei loro cromosomi e su alcune mutazioni casuali.

Anche se l'esatto funzionamento dei meccanismi evolutivi è ancora oggetto di discussione, vi è un generale accordo su alcuni principi di base che contraddistinguono lo sviluppo filogenetico degli organismi viventi:

- l'evoluzione naturale opera sui cromosomi degli individui invece che sugli individui stessi, ovvero su di una codifica genetica (il 'genotipo') delle caratteristiche fisiche dell'organismo (il 'fenotipo');
- il processo di selezione naturale favorisce la riproduzione generazionale dei cromosomi che hanno dato luogo agli organismi più efficienti dal punto di vista adattativo: esso è quindi il momento in cui il fenotipo esercita un'influenza sul genotipo, anche se questa influenza è solo indiretta poiché non prevede la trascrizione diretta nel genotipo delle caratteristiche acquisite dal fenotipo nel corso della sua vita;

- i meccanismi biologici della riproduzione costituiscono il cuore del processo evolutivo: la combinazione dei codici genetici dei due genitori e l'introduzione di piccole mutazioni casuali dà luogo alla creazione di nuove strutture genetiche la cui bontà adattiva (e di conseguenza la probabilità di riproduzione) assume un significato solo in relazione alle capacità di sopravvivenza degli individui corrispondenti;
- l'evoluzione naturale opera su popolazioni di individui attraverso un processo ciclico e generazionale che non possiede memorie storiche, ma si basa esclusivamente su contingenze ambientali definite a livello dell'interazione fra ogni singolo individuo e il proprio ambiente ecologico.

Benché le modalità operative siano basate essenzialmente su dei processi casuali senza alcuna finalità definita a priori, i sistemi biologici possiedono dunque molte caratteristiche di robustezza, di auto-organizzazione, di adattamento e di efficienza che sono altamente desiderabili se catturate ed incorporate nei sistemi artificiali creati dall'uomo. Tuttavia i metodi tradizionali di costruzione dei sistemi artificiali difficilmente riescono ad ottenere risultati eguagliabili alle prestazioni degli esseri viventi anche nei compiti più semplici, come ad esempio la navigazione, la ricerca di un rifugio, la fuga da un predatore o il riconoscimento di oggetti. Uno dei motivi di questo fallimento risiede probabilmente nel fatto che i metodi tradizionali si basano su procedimenti analitici (isolamento del problema, definizione teorica, identificazione delle variabili e derivazione formale di una soluzione specifica) che sono molto diversi dai principi di sviluppo dei sistemi biologici.

Gli algoritmi genetici invece si basano su procedimenti molto simili a quelli impiegati dall'evoluzione naturale e possiedono due precise finalità: il tentativo di comprendere meglio i meccanismi di sviluppo filogenetico dei sistemi viventi attraverso un processo simulativo di sintesi e il desiderio di sfruttare questi procedimenti per la creazione di sistemi artificiali che esibiscano le stesse caratteristiche di robustezza, efficienza e flessibilità degli organismi viventi nell'eseguire compiti difficilmente risolvibili con i metodi tradizionali.

L'avvento degli elaboratori elettronici ha rappresentato una svolta rivoluzionaria nella storia della scienza e della tecnologia, che sta accrescendo enormemente le

nostre capacità di predire e controllare la natura in forme che erano a malapena immaginabili soltanto mezzo secolo fa.

L'obiettivo di creare l'intelligenza artificiale e la vita artificiale può essere, in effetti, fatto risalire alle origini dell'era informatica: già i primi informatici inseguivano il sogno di instillare nei programmi l'intelligenza, la capacità di duplicarsi e di imparare, nonché quella di controllare l'ambiente circostante.

Il tentativo di costruire modelli per il cervello ha dato origine al settore delle reti neurali, quello di imitare l'apprendimento umano al settore dell'apprendimento automatico e, infine, la simulazione dell'evoluzione biologica ha dato vita al campo della computazione evolutiva, di cui gli algoritmi genetici sono l'esempio più importante.

Gli algoritmi genetici sono applicabili ad un'ampia varietà di problemi di ottimizzazione non indicati per gli algoritmi classici, compresi quelli in cui la funzione obiettivo è discontinua, non derivabile, stocastica, o fortemente non lineare.

2.2 Terminologia

Poiché gli algoritmi genetici nascono nell'ambito della simulazione biologica la terminologia utilizzata nella letteratura risulta un mix tra le scienze naturali e quelle artificiali [10].

Il concetto di *stringa* dei sistemi artificiali rappresenta l'analogo di *cromosoma* delle scienze biologiche.

Nei sistemi naturali uno o più cromosomi si uniscono per generare il corredo genetico totale di un organismo che prende il nome di *genotipo*. Nei sistemi artificiali il pacchetto di stringhe che contiene le informazioni da elaborare prende il nome di *struttura* (nei casi più semplici la struttura è rappresentata da una sola stringa, in tal caso i due concetti coincidono).

L'organismo che si forma dall'interazione del corredo genetico con il suo ambiente si chiamo *fenotipo*. La sua controparte artificiale derivante dalla decodifica della struttura prende il nome di *set di parametri*, *soluzione alternativa* o *punto*. Nella terminologia naturale i cromosomi sono composti da *geni* che possono presentare

determinate caratteristiche che prendono il nome di *alleli*. Inoltre nella genetica la posizione di un gene (*locus*) viene identificata separatamente dalla sua funzione.
Nella genetica artificiale possiamo affermare che le stringhe sono composte da *caratteristiche* che possono assumere diversi *valori*. Tali caratteristiche possono essere collocate in *posizioni* diverse all'interno della stringa.
Riassumendo la corrispondenza tra la terminologia naturale ed artificiale può essere schematizzata brevemente come riportato nella tabella 1.

Terminologia naturale	**Terminologia Artificiale**
Cromosoma	Stringa
Gene	Caratteristica
Allele	Valore della caratteristica
Locus	Posizione nella stringa
Genotipo	Struttura
Fenotipo	*Set* di parametri, soluzione alternativa, soluzione decodificata

Tabella 1: Confronto tra la terminologia naturale e quella artificiale.

2.3 Tecniche ed algoritmi di ottimizzazione

Nell'ambito dello sviluppo di un progetto uno dei problemi che ci si ritrova maggiormente ad affrontare consiste nel riuscire a coordinare una serie di aspetti che, pur in contrasto tra loro, devono riuscire a coesistere al meglio. Il progettista sarà chiamato ad attuare una serie di compromessi tra le varie possibilità che gli si presentano per riuscire ad ottimizzare la caratteristica scelta sulla base dei vincoli imposti dalle restanti variabili del problema considerato. Sarà bene allora, prima di esaminare i meccanismi e le capacità degli algoritmi genetici, cercare di capire un po' meglio che cosa si intende per *ottimizzazione* e quali tecniche vengano utilizzate per raggiungerla.
La definizione più convenzionale si può trovare in Beightler, Philips & Wilde (1979) [11]:

"*Man's longing for perfection finds expression in the theory of optimization. It studies how to describe and attain what is Best, once knows how to measure and alter what is Good or Bad…optimization theory encompasses the quantitative study of optima and methods for finding them*."

Il processo di ottimizzazione cerca prestazioni migliori attraverso la ricerca di qualche punto, o punti, di ottimo. Va sottolineato come questa definizione contenga due concetti distinti: (1) si cerca un miglioramento attraverso (2) qualche punto di ottimo. C'è una chiara distinzione tra il processo di miglioramento e l'ottimo in generale.

Spesso quando si giudicano le procedure di ottimizzazione ci si concentra solamente sulla convergenza (con il metodo scelto si raggiungerà il valore massimo?) tralasciando completamente le prestazioni temporanee.

Questa enfasi trae origine dal concetto di ottimizzazione presente nel calcolo matematico. Non rappresenta tuttavia un'enfasi naturale. Analizzando la gran parte dei problemi naturali ci si accorge di come la convergenza verso la soluzione migliore non sia l'obbiettivo principale piuttosto si cerca di migliorare e di superare le esperienze precedenti. Appare così chiaro come l'obiettivo principale dell'ottimizzazione sia il miglioramento.

"*Can we get to some good, "satisfing" level of performance quickly*?" (Simon, 1969) [12].

In questa ottica si inserisce la *ricerca operativa* (nota anche come *teoria delle decisioni*) ovvero quella branca della matematica applicata in cui problemi decisionali complessi vengono analizzati e risolti mediante modelli matematici e metodi quantitativi avanzati (ottimizzazione, simulazione, ecc.). L'obiettivo è quello di fornire un supporto alla presa di decisioni e, per giungere a questo scopo, la ricerca operativa fornisce strumenti matematici di supporto alle attività decisionali in cui occorre gestire e coordinare attività e risorse limitate o vincolanti al fine di massimizzare o minimizzare una funzione obiettivo.

La ricerca operativa si occupa di formalizzare un problema in un modello matematico e calcolarne una soluzione ottima, quando possibile, o approssimata. Permette di fatto di operare le scelte migliori per raggiungere un determinato

obiettivo rispettando vincoli che sono imposti dall'esterno e non sono sotto il controllo di chi deve compiere le decisioni.

Una volta scelto l'obbiettivo dell'attività si procederà ad elaborare un processo di ottimizzazione che sia caratterizzato dalle seguenti fasi:

1. riconoscere i fattori che caratterizzano l'attività da studiare;
2. scoprire le relazioni che vincolano o condizionano quei particolari fattori;
3. convertire queste relazioni in termini matematici, stabilendo un complesso di equazioni o disequazioni (*modello matematico delle politiche possibili*) le cui soluzioni definiscano tutte le possibili casistiche che si vogliono prendere in considerazione nello svolgimento di quella particolare attività;
4. indicare nel modello matematico di ottimizzazione così costruito, un algoritmo risolutivo eseguibile dai mezzi di elaborazione matematica.

Ovviamente il modello matematico utilizzato, per quanto preciso, sarà una schematizzazione che approssima la realtà, soprattutto in campo aeronautico in cui le variabili in gioco possono essere migliaia.

Risulta infatti opportuno effettuare delle riduzioni del numero di fattori in gioco e cercare di evidenziare quanto più possibile solo quelle variabili che possono essere quantificabili in modo preciso e che sono ritenute necessarie all'indagine effettuata.

Nell'ambito delle problematiche di ottimizzazione possiamo distinguere due macroclassi:

1. *problemi monobiettivo* in cui si cerca l'ottimo di una funzione scalare dipendente da una o più variabili;
2. *problemi multi obiettivo* in cui si cerca simultaneamente l'ottimo di più funzioni obiettivo che possono essere anche in contrasto tra loro, o si cerca per una sola funzione il criterio più giusto per ottimizzare tutte le variabili che anche non implicitamente ne vanno a condizionarne il valore finale.

Cerchiamo allora di analizzare in cosa consiste un problema monobiettivo tralasciando l'analisi multi obiettivo che richiede una trattazione certamente più rigorosa e che non rappresenta lo scopo principale di questa tesi.

In un problema monobiettivo, si cerca di identificare il valore ottimo delle x_i *(i=1,2,3,...,n)* variabili, che rendono ottima la funzione obiettivo $f(x_i)$, che può essere ad una variabile o a più variabili.

L'ottimo del problema può essere rappresentato dal *massimo* o dal *minimo* della funzione obiettivo *Max[f(x)]* o *Min [g(x)]*.

Ricordando che i due problemi sono equivalenti nel senso che si può passare dall'uno all'altro semplicemente cambiando il segno della funzione:

$$\begin{cases} Max\,[f(x)] = Min[-f(x)] \\ Min\,[g(x)] = Max[-g(x)] \end{cases}$$

Risolvere questo tipo di problematiche risulta essere notevolmente più semplice nel caso di funzione continua e derivabile in corrispondenza di un punto stazionario. In tal caso, e se la funzione è ad una sola variabile, basta infatti calcolare i punti in cui la derivata prima si annulla per ottenere il minimo.

In casi specifici come per esempio quelli riportati in figura 6 tale procedimento non risulta immediato.

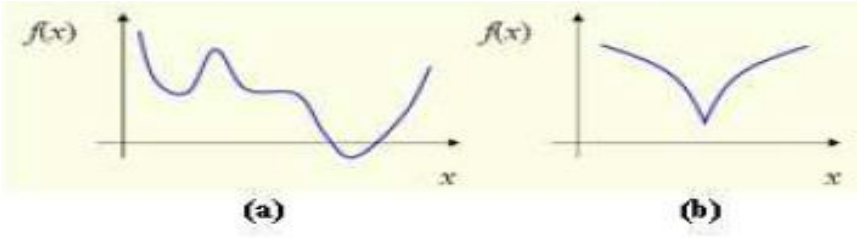

Figura 6: Esempio di funzioni multimodali e monomodali.

La funzione obiettivo può infatti essere monomodale o multimodale. Nel primo caso preso un intervallo $[x_a;x_b]$ esiste un solo punto di minimo (o massimo) x_{min} appartenente a tale intervallo (figura 3 b). Nel caso di funzione multimodale (figura 3 a), invece, all'interno di un intervallo $[x_a;x_b]$ si possono avere diversi tipi di estremi: locale, che massimizza la funzione solo all'interno di un intervallo limitato o globale, che massimizza la funzione all'interno di tutto il suo dominio.

Esistono diversi algoritmi di ottimizzazione che permettono di massimizzare una funzione obiettivo. Questi si possono suddividere in due classi: deterministici e stocastici.

Negli *algoritmi deterministici* si determinano l'estremo locale della funzione facendo uso delle condizioni di ottimo, ovvero azzeramento del gradiente e matrice Hessiana. La qualità del risultato dipende dal punto di partenza scelto.

Gli *algoritmi stocastici* prendono ispirazione dalle regole che governano alcuni fenomeni naturali e sono indipendenti dal tipo di problema scelto; inoltre appartengono alla classe degli algoritmi di ordine zero, ovvero non richiedono il calcolo delle derivate per la ricerca del massimo o del minimo della funzione.

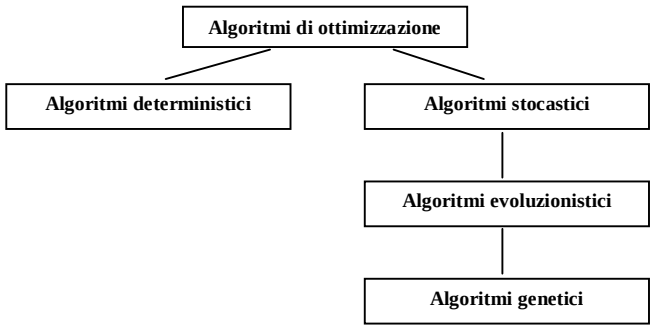

Figura 7: Classificazione degli algoritmi di ottimizzazione.

2.4 Introduzione ai processi di ottimizzazione mediante algoritmi genetici

Come visto nei paragrafi precedenti gli algoritmi genetici sono tecniche di ottimizzazione che pongono le basi nella meccanica della selezione naturale e nella genetica naturale.

Essi combinano il concetto di sopravvivenza del più adatto con uno scambio di informazioni strutturato ed in parte randomizzato.

Attraverso l'utilizzo di una serie di operazioni attuate su una popolazione di stringhe, in ogni generazione vengono introdotte nuove creature artificiali con l'utilizzo di "pezzi" degli individui più adatti della generazione precedente. Vengono inoltre introdotte anche delle parti differenti per rendere il campo di ricerca il più ampio possibile.

Si utilizzano così le informazioni accumulate dalle generazioni passate per riuscire a determinare nuovi punti di ricerca che permettano di ottenere prestazioni di volta in volta migliori.

Appare subito chiara la stretta analogia tra questi processi artificiali ed i principi della selezione naturale.

Questo concetto è stato introdotto, come già esposto, da Charles Darwin nel 1859 nel trattato *L' origine delle specie* [13], e rappresenta il meccanismo con cui avviene l'evoluzione delle specie e secondo cui, nell'ambito della diversità genetica delle popolazioni, si ha un progressivo (e cumulativo) aumento della frequenza degli individui con caratteristiche ottimali (fitness) per l'ambiente di vita.

Lo sviluppo del progetto viene quindi eseguito usando delle funzioni che ne modellino i requisiti desiderati stabilendo un complesso matematico di equazioni o disequazioni (*modello matematico delle politiche possibili)* le cui soluzioni definiscono tutte le politiche che si vogliono prendere in considerazione durante lo svolgimento della particolare attività scelta.

Si cerca così di creare un algoritmo che permetta di ottimizzare una o più funzioni obiettivo (o, usando un termine più vicino alle scienze biologiche, funzione di fitness o di idoneità).

La funzione di fitness è quella che permette di associare ad ogni soluzione uno o più parametri legati al modo in cui quest'ultima risolve il problema considerato.

Generalmente è associata alle prestazioni computazionali e quindi alle prestazioni temporali della soluzione. Essa rappresenta quindi l’oggetto vero proprio dell’indagine. Si cercherà quindi di ottimizzare tale funzione attraverso l’utilizzo di queste metodologie.

Il primo a sviluppare questo tipo di ottimizzazione è stato John Holland uno scienziato statunitense professore di psicologia, elettrotecnica ed informatica presso l'Università del Michigan di Ann Arbor.

La sua ricerca aveva un duplice scopo: da una parte cercò di spiegare i processi adattivi dei sistemi naturali e dall’altra cercò di elaborare un software che permettesse di mantenere gli stessi meccanismi dei sistemi naturali.

Il tema centrale della ricerca sugli algoritmi genetici, ed in parte la loro forza, è stato da sempre quello della loro *robustezza*, ovvero l’equilibrio tra l’efficienza richiesta e l’efficacia necessaria per potersi adattare a campi d’indagine differenti.

Rendendo più robusti i sistemi artificiali si posso, inoltre, ridurre o del tutto eliminare costose riprogettazioni.

Come numerosi studi dimostrano a partire dalla monografia di Holland del 1975 intitolata *Adaptation in Natural and Artificial Systems* [14], gli algoritmi genetici rappresentano sia teoricamente che empiricamente delle metodologie di ricerca di provata robustezza nel campo di quelli spazi di ricerca che prevedono l'analisi di sistemi complessi.

Questi algoritmi, inoltre, risultano non onerosi dal punto di vista computazionale e non sono limitati da ipotesi restrittive nello spazio di ricerca (come ad esempio continuità o esistenza delle derivate) delle metodologie per così dire più tradizionali.

Proprio per queste ragioni gli algoritmi genetici stanno trovando sempre più applicazioni in campi differenti dalle scienze naturali, all'economia fino all'ingegneria, dimostrando tutta la loro robustezza.

2.5 Robustezza delle varie metodologie

La letteratura corrente identifica tre tipologie differenti di metodologie di ricerca: *calculus-based*, *enumerative*, e *random* [10].

Le metodologie *calculus-based* sono state largamente studiate. Vengono suddivise in due classi principali: *dirette* ed *indirette*.

I *metodi indiretti* cercano gli estremi locali risolvendo un set di equazioni, di norma non lineari, risultante dall'aver imposto un valore del gradiente della funzione obiettivo pari a zero.

Questa rappresenta di fatto una generalizzazione multidimensionale della nozione elementare di calcolo degli estremi.

Cercando un possibile picco di partenza si restringe la ricerca a quei punti che presentano una pendenza nulla in ogni direzione.

I *metodi diretti*, d'altra parte, cercano un ottimo locale e si muovono in una direzione relativa al valore locale del gradiente. Questa indagine prende il nome di *hill-climbing*: si cerca un valore di ottimo locale e si risale la funzione nella direzione più ripida possibile.

Seppur entrambe queste metodologie siano state a lungo studiate e modificate alcune semplici considerazioni ne mostrano la mancanza di robustezza. Innanzitutto entrambi i metodi si basano su un'indagine locale: il valore di ottimo ricercato dipende dal valore iniziale che si è preso in considerazione, così facendo se la funzione possiede più picchi si potrebbero perdere i picchi più importanti e rischiare di analizzare solo un evento meno importante.

In secondo lungo, le metodologie *calculus-based* dipendono dall'esistenza delle derivate che, seppur se ne possa accettare un'approssimazione numerica, rimane comunque un limite molto stringente.

È chiaro che metodi che dipendono da richieste restrittive come continuità della funzione ed esistenza delle derivate rimangono accessibili solo ad un dominio limitato di problematiche.

Per queste due ragioni si può affermare che i metodi *calcus-based* sono insufficientemente robusti all'interno di determinati domini d'esistenza.

Per quanto riguarda gli *schemi enumerativi* l'idea di base è abbastanza semplice: si cerca di valutare la funzione obiettivo in ogni punto dello spazio di ricerca, uno alla volta.

Per questo motivo il loro grande difetto risiede nella mancanza di efficienza. Seppur essi possano essere reputati degli strumenti di indagine validi in campi di ricerca ristretti, molte problematiche reali richiedono degli spazi di ricerca troppo grandi per un'indagine condotta punto per punto che possa condurre a qualche risultato utile.

Proprio per queste ragioni gli algoritmi di ricerca *random* stanno avendo sempre più popolarità rispetto alle altre metodologie.

Tuttavia anche queste metodologie, che cercano e salvano il valore migliore, devono essere scartate per la loro bassa efficienza. Infatti, con una ricerca *random* non si possono aspettare dei risultati migliori di quelli ottenibili con gli schemi enumerativi.

Sarà bene allora separare le metodologie di ricerca *random* dalle tecniche di ricerca randomizzate.

In questo sottogruppo si possono infatti distinguere tre classi:

1. ricerca puramente *random*;
2. ricerca randomizzata: utilizza scelte *random* come strumento di ricerca diretta (gli algoritmi genetici ne sono un esempio);
3. ricerca ibrida: utilizza ricerche *random* congiuntamente alle altre metodologie. Per esempio cercando un ottimo globale partendo da dei valori casuali per poi utilizzare metodologie *calculus-based* per trovare i valori di massimo locale.

Gli algoritmi genetici sono un esempio di procedure di ricerca che utilizzano delle scelte *random* come strumento di guida all'interno di uno spazio di parametri codificati.

Si può così affermare che i metodi di ricerca convenzionali non sono robusti. Questo, ovviamente, non implica che siano inutilizzabili. Gli schemi menzionati e le innumerevoli combinazioni che ne derivano sono stati utilizzati con successo in molte applicazioni.

Si vedrà in seguito come gli algoritmi genetici possano essere dei validi strumenti per colmare questa mancanza di robustezza delle metodologie per così dire più tradizionali.

2.6 Confronto tra gli algoritmi genetici e altre tecniche di ottimizzazione

Le quattro principali differenze tra gli algoritmi genetici e le altre procedure di ottimizzazione [10] sono:

1) gli algoritmi genetici lavorano codificando i parametri del problema e non con i parametri stessi;
2) ricercano una popolazione di punti e non un punto solo;
3) lavorano direttamente sulla funzione (funzione obiettivo) e non sulle derivate o su altre informazioni ausiliare;
4) utilizzano un metodo probabilistico e non deterministico.

La prima differenza sostanziale tra gli algoritmi genetici e le procedure di indagine tradizionali consiste, per l'appunto, nella codifica dei parametri del problema in stringhe di lunghezza finita composte utilizzando un alfabeto di caratteri finito.

Per esempio pensando di voler massimizzare una funzione in cui compaia una sola variabile *x*, i metodi tradizionali andrebbero a lavorare direttamente sul parametro *x* cercando di incrementarlo finché la funzione obiettivo non raggiunga il valore più alto.

Negli algoritmi genetici il primo passo del processo di ottimizzazione consiste nella codifica della variabile x come una stringa di lunghezza finita.

Per esempio prendiamo una funzione $f = f(s)$ dove *s* rappresenta una configurazione particolare di cinque parametri. Immaginando questi parametri come degli *switch on-off*, con le altre procedure di ottimizzazione si lavorerebbe direttamente sulla posizione degli interruttori spostandoli da una posizione all'altra usando le regole di transizione del particolare metodo utilizzato. Con gli algoritmi genetici si codificano le posizioni degli *switch*, ad esempio usando un codice binario dove 1 corrisponde alla posizione di acceso e lo 0 corrisponde alla posizione di spento, con una stringa dalla lunghezza finita. Ogni stringa viene generata in modo del tutto casuale scegliendo ogni volta tra il valore 1 o 0. Così la stringa 1110 corrisponderà ai primi tre interruttori accesi e l'ultimo spento.

Si genera così una popolazione di stringhe su cui iniziare ad effettuare l'indagine.

Successivamente si vedrà come gli algoritmi genetici esplorino le similitudini tra le codifiche dei parametri per ottenere ulteriori informazioni sul problema considerato. Proprio per questo motivo gli algoritmi genetici hanno il vantaggio di essere molto meno soggetti alle limitazioni imposte agli altri metodi (continuità, esistenza delle derivate ed altro ancora).

Un altro vantaggio è rappresentato dall'utilizzare una popolazione di stringhe e non, come negli altri metodi, andando ad effettuare una ricerca punto dopo punto utilizzando delle regole di transizione.

La ricerca punto-punto risulta molto pericolosa per l'alta probabilità di ottenere dei falsi picchi in campi di ricerca multimodali (*many peaked*) come mostrato nella figura 8.

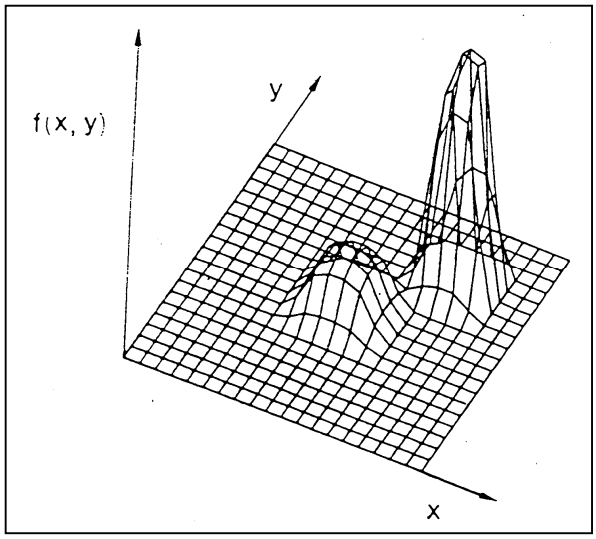

Figura 8: Funzione multimodale (*many peaked*).

Lavorando, invece, con una popolazione di stringhe, gli algoritmi genetici analizzano in parallelo i diversi possibili picchi riducendo di gran lunga la possibilità di ottenere dei falsi positivi.

Inoltre le altre procedure di ricerca per effettuare il processo di ottimizzazione richiedono la conoscenza di molte altre informazioni aggiuntive per poter essere opportunamente utilizzate. Ad esempio, il metodo del gradiente richiede l'utilizzo delle derivate (calcolate analiticamente o numericamente) mentre altre procedure di ricerca locale come le tecniche *greedy* richiedono la conoscenza della maggior parte, se non di tutti, i parametri tabulati.

D'altra parte gli algoritmi genetici non hanno bisogno della conoscenza di queste informazioni ausiliare bensì affrontano la ricerca semplicemente analizzando il valore finale che una particolare stringa fa assumere alla funzione obiettivo che si è scelto di analizzare.

Questa caratteristica rende gli algoritmi genetici molto più applicabili degli altri metodi di ricerca.

Infine, mentre le altre metodologie utilizzano un approccio deterministico gli algoritmi genetici si muovono nello spazio di ricerca utilizzando delle regole di transizione probabilistiche.

A prima vista può sembrare una ricerca alla cieca, tuttavia gli algoritmi genetici utilizzano il metodo probabilistico come strumento per guidare la ricerca ottenendo un'alta possibilità di miglioramento.

Messe insieme, queste quattro differenze, uso diretto della codifica dei dati, ricerca di una popolazione, indipendenza dalle informazioni ausiliarie e l'utilizzo di operatori randomizzati, contribuiscono a rendere gli algoritmi di ricerca molto più robusti rispetto alle altre tecniche.

3. Struttura base di un algoritmo genetico

All'interno del capitolo verrà presentata la struttura e i principi di base di un algoritmo genetico e verrà presentato un primo esempio numerico.

3.1 Il funzionamento generale

La struttura base di un algoritmo genetico prevede 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 con caratteristiche che risultano sempre migliori al passare dei cicli.

Il primo passo di questo algoritmo è creare una popolazione casuale dalla quale partire (figura 9) mentre le fasi successive si ripetono ad ogni generazione e sono associate al principio della selezione naturale o alla genetica.

In particolare le prime fasi si occupano di selezionare gli individui migliori della popolazione, mentre le altre di generare nuovi individui.

Quest'ultime operazioni sono possibili unendo o modificando le caratteristiche che identificano un individuo.

Dal punto di vista della genetica, i nuovi cromosomi si ottengono ricombinando il loro patrimonio genetico ovvero andando a modificare i geni con gli operatori di mutazione e di *crossover*.

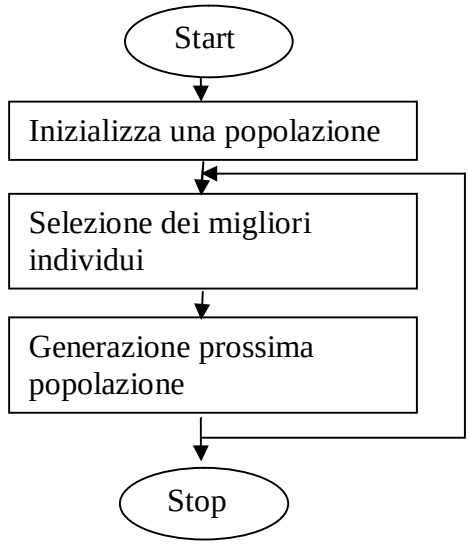

Figura 9: Struttura generale di un algoritmo genetico.

Per ogni combinazione di geni è possibile calcolare un valore di *fitness* che indica la capacità con cui il cromosoma o la soluzione sia in grado di risolvere il problema. Nella selezione naturale questo valore misura l'adattamento dell'individuo nell'ambiente. Per cui una *fitness* migliore è legata ad una maggiore probabilità di sopravvivenza, mentre all'interno dell'algoritmo genetico ad una maggiore probabilità di selezione.

3.2 Principi di base

La struttura base di un algoritmo genetico risulta essere piuttosto semplice, richiedendo principalmente delle operazioni di copia e di scambio parziale di stringhe.

Tuttavia la spiegazione teorica di come un processo così semplice, almeno ad una prima analisi, possa funzionare, risulta essere un argomento molto più sottile.

Proprio la semplicità con cui possono essere eseguite le operazioni necessarie alla risoluzione dell'algoritmo e la robustezza dei risultati sono le due caratteristiche principali del potere di questi algoritmi.

Il primo passo da effettuare nell'utilizzo di queste tecniche è quello di generare una popolazione iniziale di individui posta sotto forma di stringhe.

Scelta la funzione che si vuole ottimizzare, ovvero la funzione obiettivo, si analizzano le possibili variabili del campo di ricerca e si procede a generare un possibile scenario di partenza per la successiva analisi.

Si sceglie allora la lunghezza che dovrà caratterizzare le varie stringhe (l) ed un alfabeto finito di caratteri (m), come può essere quello binario.

Si procede allora, in modo del tutto casuale, ad assegnare ad ogni individuo delle caratteristiche all'interno dell'alfabeto di codifica scelto.

Verrà così generata una popolazione iniziale di individui dalla grandezza più o meno ampia in base al problema considerato.

Gli algoritmi genetici operano su questa popolazione iniziale di cromosomi artificiali, che vengono fatti riprodurre selettivamente in base alle prestazioni dei fenotipi e che danno origine rispetto al problema da risolvere. Durante il processo riproduttivo le repliche dei cromosomi degli individui migliori vengono accoppiate

casualmente e parte del materiale genetico viene scambiato, mentre alcune piccole mutazioni casuali alterano localmente la struttura del codice. Le nuove strutture genetiche vanno quindi a rimpiazzare quelle dei loro genitori dando luogo ad una nuova generazione di individui. Il processo continua fino a quando nasce un individuo che rappresenta una soluzione accettabile per il problema in esame.

Gli algoritmi genetici si basano quindi su tre operatori principali:

1. riproduzione selettiva;
2. *crossover* o ricombinazione genetica;
3. mutazione casuale dei cromosomi.

3.2.1 Codifica

Per ‘codifica genetica’ ci si riferisce al tipo di rappresentazione che viene utilizzata per identificare le soluzioni del problema nei cromosomi artificiali. Un cromosoma artificiale è rappresentato da una sequenza di simboli che costituiscono le caratteristiche dell’individuo: per questo motivo viene comunemente definito con il nome di ‘stringa genetica’.

Un tipo di codice utilizzato molto frequentemente è quello binario: in questo caso il cromosoma di ciascun individuo della popolazione è una stringa di lunghezza finita di simboli binari (0,1).

Vi sono molti altri tipi di codifica possibile: un cromosoma può essere anche composto da una serie di numeri reali oppure da una serie finita di simboli appartenente ad un qualsiasi alfabeto arbitrario. Il funzionamento dell’algoritmo genetico non è compromesso dal tipo di rappresentazione in quanto gli operatori genetici si limitano a selezionare le stringhe corrispondenti ai fenotipi migliori e a ricombinarne i vari pezzi a prescindere dal materiale su cui essi lavorano.

La scelta del tipo di codifica è però importante per il tipo di problema che si vuole risolvere: non esiste una codifica che vada bene per tutti i problemi, né esistono regole generali che permettano di fare delle scelte ottimali.

Il problema della rappresentazione genetica e delle regole di decodificazione da genotipo a fenotipo è quindi molto importante per poter sfruttare al meglio le potenzialità di ricerca dell’algoritmo genetico.

3.2.2 Funzione obiettivo

La funzione obiettivo (o funzione di valutazione) serve per giudicare le prestazioni di ciascun fenotipo rispetto al problema che vogliamo risolvere: essa fornisce un valore numerico per ciascun individuo proporzionale alla bontà della soluzione offerta.

Intuitivamente si può pensare alla funzione obiettivo come ad una forma di profitto, di utilità, o ad una caratteristica generale che si è deciso di massimizzare.

Questa particolare funzione svolge un ruolo analogo a quello dell'ambiente fisico per gli organismi biologici, in quanto misura le prestazioni dell'individuo: per questo motivo viene comunemente definita 'funzione di *fitness*'.

Una caratteristica interessante degli algoritmi genetici è che la funzione di *fitness* può assumere vari gradi di complessità a seconda delle conoscenze disponibili: l'unico requisito è che essa permetta un ordinamento delle stringhe genetiche in base alle loro prestazioni sul problema da risolvere.

3.2.3 Riproduzione selettiva

Una volta definiti il tipo di rappresentazione genetica e la funzione di *fitness*, il primo passo consiste nella creazione di una popolazione iniziale di stringhe genetiche.

Solitamente la popolazione iniziale è composta da stringhe casuali. Ciascuna stringa di questa generazione iniziale viene a turno decodificata e valutata in base alla funzione obiettivo.

Una volta calcolata la *fitness* di ogni individuo viene scelta una sottopopolazione per generare nuove soluzioni. Esistono varie tecniche per selezionare l'insieme delle soluzioni per la riproduzione chiamato *mating pool*. La più immediata e semplice consiste nel ordinare gli n individui della popolazione in base al valore della funzione di *fitness* e scegliere le prime k soluzioni (con $k < n$).

Consideriamo $P(t)$ la popolazione alla generazione t-esima formata da n individui ordinati secondo il valore dello loro *fitness* in modo crescente.

$P(t)=\{x_1, x_2,\ldots, x_n\}$

Se l'algoritmo genetico deve risolvere un problema di minimo è sufficiente selezionare i primi *k* elementi (figura 10 a), altrimenti gli ultimi *k* elementi se è un problema di massimo (figura 10 a).

Un'altra tecnica molto usata prende il nome di *proportional selection* che prevede di selezionare gli individui in base ad una probabilità p_i proporzionale alla loro *fitness*.

Il processo di riproduzione selettiva consiste così nella creazione probabilistica di un numero di copie di ciascuna stringa proporzionale al valore di *fitness* ottenuto dal fenotipo corrispondente.

Ricopiare ciascuna stringa in proporzione al proprio valore obiettivo significa che le stringhe che hanno riportato un valore di fitness maggiore avranno una probabilità maggiore di produrre uno o più figli: l'operatore di riproduzione selettiva svolge dunque un ruolo simile alla legge di sopravvivenza del più "adatto" in natura.

Vi sono diversi modi di realizzare al computer la riproduzione selettiva probabilistica: il metodo più diffuso fa ricorso all'utilizzo della *ruota della fortuna truccata* (figura 10). Consideriamo la situazione comune in cui manteniamo una popolazione di dimensione costante ad ogni generazione e in cui rimpiazziamo completamente tutti i membri della popolazione ad ogni ricambio generazionale. La ruota della fortuna avrà dunque tante caselle quanti sono gli individui della popolazione ma la dimensione della casella sarà proporzionale ai valori di *fitness* di ciascun individuo.

La riproduzione selettiva consiste nel far 'girare' la ruota tante volte quanti sono gli individui da generare e nel creare ogni volta una copia della stringa corrispondente alla casella in cui si trova alla fine l'indice della ruota.

In particolare come primo passo viene calcolata la *fitness* totale *F* della popolazione *P(t)*.

$$F_{\max} = \sum f(x_i) \qquad F_{\min} = \sum \frac{1}{f(x_i)}$$

In questa espressione $f(x_i)$ rappresenta il valore della *fitness* dell'individuo x_i e le sommatorie sono intese da 1 ad n ovvero su tutta la popolazione *P(t)*. Inoltre il calcolo della *fitness* totale dipende dal se stiamo considerando problemi di minimo

o di massimo. Grazie a questo valore è possibile ricavare la probabilità con cui le soluzioni saranno selezionate.

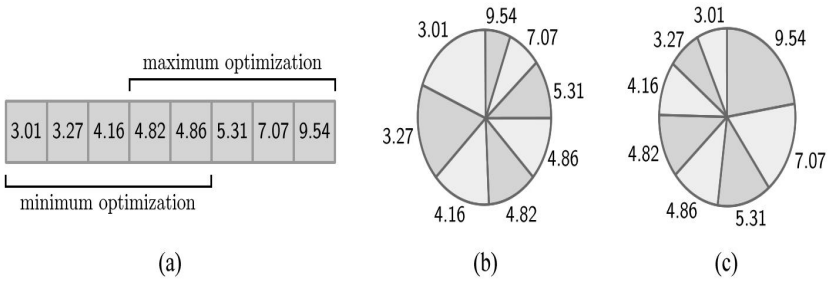

Figura 10: Selezione lineare e *proportional selection*.

Per ottenere queste probabilità si ricorre alle seguenti formule, utili nel caso di problemi di minimo e di massimo.

$$p_i = \frac{1}{f(x_i) \cdot F_{\min}} \qquad p_i = \frac{f(x_i)}{F_{\max}}$$

Il risultato di queste due espressioni può essere rappresentato con una ruota suddivisa in spicchi, la cui grandezza è proporzionale alla probabilità associata alle soluzioni (figura 10b e c). La ruota viene fatta girare per *k* volte, *n* se vogliamo una generazione con lo stesso numero di individui della popolazione iniziale, e vengono estratte le soluzioni corrispondenti che andranno a fa parte del *mating pool*.

Il valore atteso del numero di figli per ciascuna stringa è dato da ($n \cdot p_i$), ma il numero effettivo di figli si ottiene facendo ricorso ad un generatore di numeri casuale.

Una volta selezionata per la riproduzione la stringa viene copiata interamente andando a comporre una nuova popolazione di tentativo su cui agiranno gli altri operatori.

Si inizia così ad intraprendere un percorso di selezione degli individui con le caratteristiche migliori, ovvero con il valore più alto di *fitness*, che andranno a creare la generazione successiva.

3.2.4 *Crossover*

Dopo la riproduzione si procede al *crossover* attraverso due semplici operazioni.

Per prima cosa si raggruppano in modo casuale le stringhe a gruppi di due. Ciascuna coppia verrà incrociata con una probabilità p_c scelta a priori.

Si procede successivamente a scegliere casualmente una posizione k all'interno delle stringhe, con k che può variare tra 1 e la lunghezza di una stringa l meno uno [1 , l-1].

Si vanno allora a creare due nuove stringhe scambiando i caratteri dalla posizione k+1 ad l compresi. Per esempio se consideriamo due stringhe A_1 ed A_2 di lunghezza l = 10 codificate con un alfabeto binario, che compongono la popolazione iniziale dove:

A_1 = 1 0 1 0 | 0 0 1 1 1 0

A_2 = 0 0 1 1 | 0 1 0 0 1 0

Ipotizziamo che scegliendo in modo casuale un numero da uno a quattro otteniamo k=4 (come indicato dal simbolo |). Le nuove stringhe che risulteranno dall'incrocio delle due iniziali e che andranno a comporre la nuova generazione risulteranno essere:

A_1´ = 1 0 1 0 0 1 0 0 1 0

A_2´ = 0 0 1 1 0 0 1 1 1 0

In cui si è applicato il crossover come visibile in figura 11.

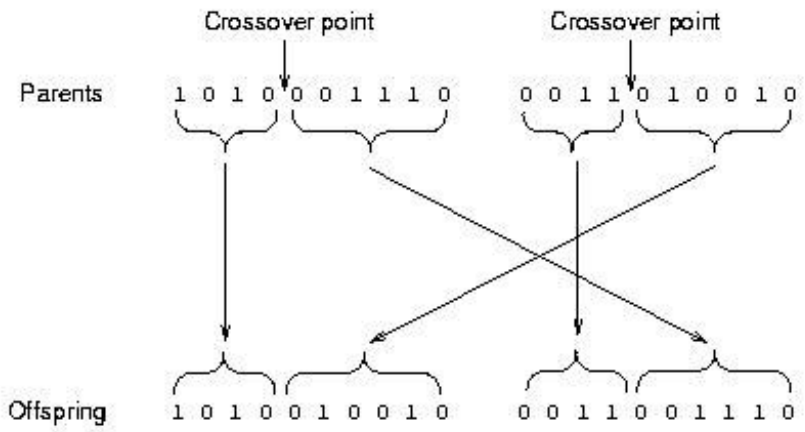

Figura 11: *Crossover* binario ad un punto.

Il *crossover* binario si chiama *ad un punto* quando il cromosoma figlio di lunghezza n è composto dai primi k elementi del primo padre e dagli ultimi n-k elementi del secondo (figura 11). Nel *crossover a due punti* si usa la stessa tecnica anche se i genitori vengono divisi in tre parti. Infine il *crossover* a n-punti si ottiene

costruendo un cromosoma dove l'i-esimo elemento è preso con la stessa probabilità dal primo o dal secondo padre.

Differente è l'approccio della codifica reale, in quanto non è riconducibile al *crossover* della biologia molecolare. Per cui l'operatore viene costruito con altre tecniche, sempre basate sul concetto di unire le caratteristiche di due individui per crearne un terzo.

L'operatore di *crossover* viene applicato a due cromosomi a codifica reale la cui struttura è un vettore di reali di lunghezza *n*. Questo tipo di codifica rientra nei modelli di rappresentazione *one gene-one variable*, dove esiste una corrispondenza uno a uno tra i geni del cromosoma e le variabili dello spazio delle soluzioni. Consideriamo due individui C_1 e C_2 con *n* componenti:

$C_1=(c_1^1, c_2^1,\ldots, c_n^1)$ $C_2=(c_1^2, c_2^2,\ldots, c_n^2)$

Il *crossover* più semplice è chiamato *flat crossover* dove l' i-esima componente del cromosoma figlio viene scelta casualmente nell'intervallo *[c1,c2]*, determinato dalle i-esime componenti dei vettori C_1 e C_2.

I meccanismi della riproduzione e del *crossover* risultano essere sorprendentemente facili da attuare, richiedendo niente di più che una generazione di numeri casuali, copia delle stringhe e qualche cambio parziale.

Nondimeno, la combinazione tra riproduzione e lo strutturato, seppur randomizzato, scambio di informazioni donano all'algoritmo genetico gran parte della sua forza.

Si può vedere come in un processo del genere si cerchi di costruire nuove soluzioni a partire dalle soluzioni parziali che nelle prove precedenti hanno fornito i risultati migliori.

Seppur la riproduzione ed il *crossover* rendano l'algoritmo molto robusto è necessario introdurre un ultimo operatore ovvero la mutazione.

3.2.5 Mutazione

In un semplice algoritmo genetico la mutazione rappresenta l'occasionale (cioè con probabilità p_m molto basse) alterazione casuale di un valore all'interno di una stringa.

Vi sono diversi tipi di mutazione a seconda del tipo di rappresentazione genetica impiegata: ad esempio, nel caso del codice binario, gli elementi selezionati per essere mutati modificano il loro stato da zero a uno o viceversa (figura 12); nel caso di un codice composto da numeri reali, invece, gli elementi selezionati assumono un nuovo valore estratto in modo casuale da un certo intervallo.

La teoria tradizionale ritiene che il *crossover* sia più importante della mutazione per quanto riguarda la rapidità nell'esplorare lo spazio di ricerca; la mutazione però apporta un po' di 'casualità' al processo in modo che tutti i punti dello spazio di ricerca abbiano possibilità di essere esplorati.

I meccanismi della riproduzione e del *crossover* risultano, infatti, essere sorprendentemente facili da attuare, richiedendo niente di più che una generazione di numeri casuali, copia delle stringhe e qualche cambio parziale.

Questo processo risulta essere fondamentale anche perché attraverso la riproduzione ed il crossover si rischia di perdere qualche informazione importante (per esempio nel caso di codifica binaria un 1 o uno 0 in qualche posizione particolare).

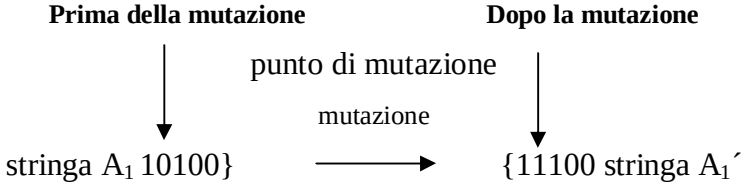

Figura 12: Mutazione.

Questo operatore può essere così visto come un'ulteriore assicurazione contro la prematura perdita di informazioni.

Per ottenere dei buoni risultati dall'algoritmo si usano di norma delle percentuali molto basse nell'ordine di un valore mutato ogni mille bit.

Il basso tasso di mutazione ci porta così a concludere che la mutazione stessa rappresenta un meccanismo secondario ma necessario all'interno del processo di ottimizzazione attuato dagli algoritmi genetici.

3.3 Convergenza

La parte dell'algoritmo ispirata al principio della selezione naturale si occupa di classificare gli individui e selezionare le soluzioni migliori all'interno della popolazione. Con queste operazioni viene sollecitata la riproduzione tra gli individui migliori e viene garantita la convergenza dell'algoritmo verso una soluzione ottima che può essere locale o globale.

Per ottenere una convergenza globale è importante che la popolazione sia composta da individui con un patrimonio genetico vario. Questo si traduce in un insieme eterogeneo di soluzioni con una visione globale dello spazio di ricerca, capace di individuare la completa topologia del problema.

Così grazie agli operatori dell'algoritmo è possibile ottenere un'esplorazione del campo di ricerca capace di generare da una generazione di partenza individui con caratteristiche via via migliori.

La nuova popolazione di stringhe genetiche rimpiazza parzialmente o completamente le vecchie stringhe. Il processo di decodifica, valutazione, riproduzione selettiva, incrocio e mutazione si ripete ciclicamente per parecchie generazioni fino a quando viene ottenuta una stringa che codifica una soluzione soddisfacente.

Se l'algoritmo genetico è correttamente implementato, la popolazione evolverà nel susseguirsi delle generazioni, in modo che la *fitness* del miglior individuo e la media in ogni generazione crescano verso l'ottimo globale (figura13). La convergenza è la progressione verso la crescente uniformità. Si dice che un gene converge quando il 95% della popolazione condivide lo stesso valore. La popolazione converge quando tutti i geni convergono.

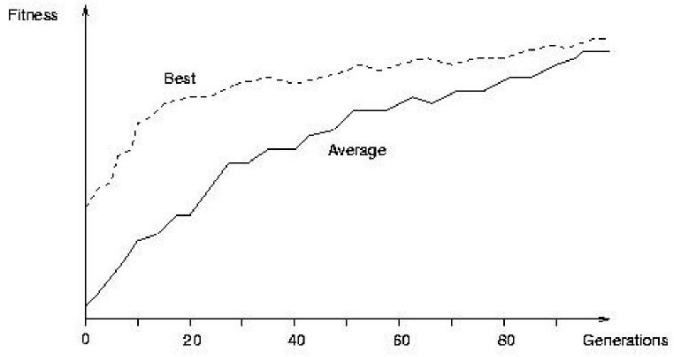

Figura 13: Convergenza di un algoritmo genetico.

3.4 Un primo esempio di applicazione

In questo paragrafo verrà presentato un semplice esempio in cui sarà possibile osservare il funzionamento di un algoritmo genetico.

L'esempio proposto è tratto dal libro '*Genetic Algorithms in Search, Optimization and Machine Learning*' del 1989 scritto da D.E.Goldberg [10] e permetterà di osservare come opera un semplice algoritmo genetico.

Il problema consiste nel trovare i valori della variabile *x* che massimizzano la funzione potenza $\Phi(x) = x^2$, dove *x* può variare tra 0 e 31.

Il primo passo consiste nella scelta della rappresentazione genetica: si utilizzeranno un codice binario e stringhe di lunghezza 5 che permettono di codificare numeri (interi) da 0 ('00000') a 31 ('11111').

Viene creata dunque una piccola popolazione composta solamente da quattro stringhe che vengono generate casualmente. Ciascuna stringa viene decodificata e il suo valore di *fitness* viene calcolato applicando la funzione Φ al numero decimale ottenuto (tabella 2).

Alla fine del processo di valutazione ogni stringa riceve una probabilità di riproduzione proporzionale alla propria *fitness*. Moltiplicando questa probabilità per il numero *n* di individui nella generazione si può ottenere una prima stima del numero di figli che verranno generati da ciascuna stringa nella generazione successiva.

Osserviamo due importanti indici evolutivi: il valor medio ed il valore massimo della *fitness*.

Num. Stringa	**Popolaz. iniziale**	**Decod. *X***	**Valutazione $F(x)=x^2$**	**Prob. di riprod. $p_i=f(x_i)/\sum f(x_i)$**	**Num.figli atteso $N = p_i \cdot n$**	**Num. Estraz.**
1	01101	13	169	0.14	0.58	1
2	11000	24	576	0.49	1.97	2
3	01000	8	64	0.06	0.22	0
4	10011	19	361	0.31	1.23	1
somma			1170	1.00	4.00	4
media			293	0.25	1.00	1
massimo			576	0.49	1.97	2

Tabella 2: Dati popolazione iniziale.

Siccome una popolazione è sempre composta da individui che ottengono prestazioni diverse, il valore medio è sempre più basso del valore massimo.

Le nuove stringhe vengono poi accoppiate e per ciascuna di esse (qui la probabilità di *crossover* p_c = 1.0) viene scelto un punto di incrocio casuale attorno al quale esse si scambiano parte del loro materiale.

Infine ciascun elemento viene mutato con probabilità p_m= 0.001: in questo caso nessun elemento subisce modifiche. La nuova popolazione di cromosomi così ottenuta viene nuovamente decodificata e valutata: sia il valore medio che il valore massimo di *fitness* sono superiori al valore della generazione iniziale (tabella 3).

Accoppiam. con crossover (!)	**Incrocio p_c=1.0 (tutti)**	**Mutazioni p_m=0.001 (nessuna)**	**Nuova popolaz.**	**Decod. *X***	**Valutaz. $\Phi(x) = x^2$**
0110!1	01100	01100	01100	12	144
1100!0	11001	11001	11001	25	625
11!000	11011	11011	11011	27	729
10!011	10000	10000	10000	16	256
somma					1754
media					439
massimo					729

Tabella 3: Prima generazione.

Se si osserva il risultato dell'operatore di crossover, si nota che lo scambio casuale non produce necessariamente stringhe ottimali: in questo caso per entrambe le coppie scelte viene generata una stringa con prestazioni uguali o addirittura peggiori della media della generazione precedente (*fitness* 246 e 144). Vengono tuttavia generate anche stringhe migliori con *fitness* 625 e 729.

Il processo di riproduzione selettiva fa sì che mediamente solo le stringhe migliori contribuiscano alla creazione della nuova popolazione. Come è possibile vedere già dopo una sola iterazione l'algoritmo ha creato delle stringhe con valor medio e massimo sempre più vicini alla condizione di ottimo. Inizia così il processo evolutivo degli operatori dell'algoritmo verso una convergenza della soluzione.

4. La Teoria degli Schemi

Si procede ora a delineare nel dettaglio uno degli aspetti più importanti e fondamentali della teoria degli algoritmi genetici. In particolare verrà esposto quello che prende il nome di *Teorema degli Schemi* o *Teorema fondamentale degli algoritmi genetici*.

4.1 Modelli di somiglianza

All'interno della ricerca effettuata dagli algoritmi genetici ci si è finora soffermati sull'importanza di generare delle stringhe che, generazione dopo generazione, aumentino il valore della funzione obiettivo.

Si vuole ora porre il problema sotto un'altra prospettiva: l'indagine effettuata dagli algoritmi genetici, infatti, può essere guidata dall'utilizzo di modelli di somiglianza tra stringhe con un alto valore di *fitness*.

Uno schema o modello di somiglianza può descrivere un sottogruppo di stringhe con delle caratteristiche simili in determinate posizioni all'interno delle stringhe stesse [10].

Per questa discussione si consideri ancora una volta come alfabeto di codifica un semplice codice binario {0,1}.

Per riuscire ad identificare più facilmente uno schema aggiungiamo a questo alfabeto il simbolo * che identificherà la possibilità di avere uno qualsiasi tra i valori utilizzati nella codifica (un 1 o uno 0 nel caso preso in considerazione).

Con l'aggiunta di questo simbolo si possono ora creare degli schemi o modelli di somiglianza (sotto forma di stringhe) utilizzando un alfabeto più esteso composto da {0,1,*}.

Per esempio considerando delle stringhe di lunghezza 5 come quelle del paragrafo precedente, lo schema *111* rappresenterà il gruppo di stringhe composto da {01110,11110,01111,11111}.

Inizia ad apparire più chiaro come gli schemi possano aiutare ad identificare delle similitudini all'interno del set di stringhe che vengono elaborate generazione dopo generazione.

Considerando un alfabeto di k elementi che compongono stringhe di lunghezza l le possibili stringhe che possono esse generate saranno dell'ordine di k^l. Con l'aggiunta del simbolo * si potrà allargare la ricerca a $(k+1)^l$ possibili schemi.

A prima vista sembra che non si sia fatto altro che rendere la ricerca più difficile andando a considerare un numero di dati maggiore di quello di partenza.

Occorre allora soffermarsi a considerare come le singole stringhe, contengano semplicemente delle parti di informazioni. Se prese singolarmente, infatti, sarà possibile solo valutare come la particolare configurazione di variabili che fornisce quella particolare stringa vada a far ottenere un determinato valore alla funzione obiettivo. Andando ad allargare lo spazio di ricerca attraverso l'utilizzo di schemi si possono ottenere molte più informazioni utili alla ricerca. Si dimostra che gli algoritmi genetici vagliano ad ogni generazione l'equivalente di almeno n^3 schemi diversi, benché vengano valutate solamente n stringhe.

Cercando analogie tra la popolazione di stringhe sarà infatti possibile non solo trovare delle similitudini e quindi analizzare le caratteristiche che rendono una particolare configurazione di variabili come adatta a risolvere il problema considerato, ma addirittura a prevedere con buona approssimazione i risultati ottenibili dalle successive iterazioni.

All'interno di questo capitolo si andrà ad analizzare più nel dettaglio come gli schemi vengano elaborati dagli operatori dell'algoritmo e verrà presentato quello che prende il nome di *Teorema Fondamentale degli Algoritmi Genetici* o *Teorema degli Schemi*.

Per ora ci si limita ad osservare come andando a cercare delle similitudini all'interno della popolazione si possano ottenere molte più informazioni rispetto al considerare solo il valore di *fitness* della singola stringa.

Questo processo di elaborazione di modelli di somiglianza appare così importante (e apparentemente limitato ai soli algoritmi genetici) che prende il nome di *parallelismo implicito*.

4.2 Gli schemi come strumenti di miglioramento delle performance

All'interno dell'ultimo paragrafo è stato introdotto il concetto di modello di somiglianza o schema: uno schema è una stringa composta da un alfabeto finito di caratteri.

All'interno di una popolazione di n stringhe ci possono essere numerose somiglianze (dell'ordine di n^3) da poter esplorare: infatti andando ad analizzare gli individui generati dopo ogni generazione all'interno dell'algoritmo, è possibile riconoscere parti di stringhe simili a cui corrispondono caratteristiche simili.

Esplorare queste somiglianze rende il campo di indagine più vasto, visto l'uso di un alfabeto più ampio, e permette di ricavare informazioni più complete rispetto al considerare le stringhe singolarmente. Infatti se si considerano le stringhe di un'intera popolazione e il valore della loro *fitness* cercando delle somiglianze si può arricchire il campo di ricerca con molte più informazioni utili.

Basta pensare che così facendo si possono creare delle stringhe con parti degli individui che meglio soddisfano la funzione obbiettivo studiata ottenendo performance migliori.

L'algoritmo genetico di fatto già opera un incrocio tra gli individui attraverso l'operatore di *crossover* che ne mescola le proprietà. Appare allora chiaro come al suo interno l'algoritmo elaborando una popolazione di stringhe attraverso la riproduzione, il *crossover* e la mutazione generazione dopo generazione processi implicitamente anche gli schemi trovati.

E' quindi necessario andare a delineare come gli operatori dell'algoritmo intervengano sugli schemi osservati e cercare di quantificare quanti ve ne possano essere all'interno di una popolazione.

Prima però si introducono alcune proprietà fondamentali che caratterizzano gli schemi ovvero l'ordine e la lunghezza definita.

4.3 Ordine e lunghezza di uno schema

Si considerino stringhe costruite utilizzando un alfabeto binario V = {0,1}. Si identificano con le lettere maiuscole le stringhe e con le lettere minuscole le singole caratteristiche. Ad esempio la stringa A=10100 è generalizzata come A = a_1 a_2 a_3 a_4 a_5 dove ognuno degli a_j rappresenta una singola caratteristica (o gene) il cui valore (o allele) può essere 1 o 0.

Si indica invece uno schema con la lettera H = h_1 h_2 h_3 h_4 h_5 ed è composto da un alfabeto allargato V + = {0,1,*}. Per l'algoritmo si considera una popolazione di stringhe A_j con j = 1,2,…,*n* contenute in una popolazione A(*t*) al tempo (o generazione) *t*.

Per avere un'idea di quanti schemi vi possano essere si considera che usando un alfabeto di cardinalità *k* e stringhe di lunghezza *l* esistono $(k+1)^l$ schemi possibili. Inoltre in una popolazione di *n* stringhe si possono identificare $(n \cdot k^l)$ possibili schemi considerando che ogni stringa è rappresentativa di k^l schemi.

Viste le dimensioni delle informazioni elaborate occorre definire dei parametri che permettano di distinguere i vari schemi. Per metterne in luce le differenze ci si avvale di due strumenti: ordine e lunghezza definita. L'ordine di uno schema *H*, definito $o(H)$, rappresenta semplicemente il numero di posizioni fissate (in un alfabeto binario, il numero di 1 e di 0 usati per rappresentarlo) presenti in uno schema.

La lunghezza di uno schema *H*, definita $\delta(H)$, è la distanza tra l'ultima e la prima posizione identificata all'interno di una stringa.

Ad esempio prendendo lo schema *H* = *1**0 avrà una lunghezza $\delta(H)$ = 5-2 = 3 poiché l'ultima cifra identificata si trova in posizione 5 e la prima in posizione 2 ed un ordine $o(H)$ = 2 poiché sono solo due le posizioni fissate.

4.4 L'effetto dell'algoritmo sugli schemi

Ora che sono state definite le proprietà di uno schema, ovvero ordine e lunghezza, si va ad analizzare l'effetto che gli operatori di riproduzione, crossover e mutazione hanno su questi modelli di somiglianza.

4.4.1 Effetto della riproduzione

Per prima cosa determiniamo gli effetti della riproduzione su un insieme di schemi presenti in una popolazione di *n* individui.

Si considerino nell'istante *t*, *m* esempi dello schema *H* contenuti nella popolazione A(*t*) ovvero *m* = *m*(*H*,*t*).

Durante la riproduzione, le probabilità che una stringa venga duplicata, sono proporzionali al suo valore adattativo, più precisamente una stringa A_i avrà probabilità p_i di essere selezionata pari a:

$$p_i = \frac{f(x_i)}{\sum f(x_i)}$$

dove $f(x_i)$ indica il valore adattativo dello i-esimo individuo e la sommatoria si riferisce a tutti gli individui della popolazione ovvero con i=1,2,…,*n*.

Analogamente, lo schema *H* avrà probabilità di comparire in una popolazione pari a:

$$p_i = \frac{f(H)}{\sum f(x_i)}$$

Essendo *f(H)* il valore adattativo medio delle stringhe rappresentanti lo schema *H* nell'istante *t*. Pertanto il numero *m*(*H*,*t*+1) di rappresentanti dello schema *H* nella popolazione A(*t*+1), ovvero all'istante *t*+1 o meglio nella popolazione ottenuta dopo la riproduzione, è dato dall'equazione:

$$m(H,t+1) = m(H,t) \cdot n \cdot \frac{f(H)}{\sum f(x_i)}$$

Si osservi che:

$$m(H,t+1) = m(H,t) \cdot n \cdot \frac{f(H)}{\sum f(x_i)} = m(H,t) \cdot f(H) \cdot \frac{n}{\sum f(x_i)} = m(H,t) \cdot \frac{f(H)}{\sum f(x_i)/n}$$

e si noti che $\sum f(x_i)/n$ altro non è che il valore adattativo medio dell'intera popolazione, ponendo $\sum f(x_i)/n = f$ si ottiene quindi:

$$m(H,t+1) = m(H,t) \cdot \frac{f(H)}{f}$$

La formula appena ottenuta lascia intendere che uno schema si propaga in misura proporzionale al rapporto tra il valore adattativo medio dello schema stesso ed il

valore adattativo medio dell'intera popolazione. Così schemi con valore adattativo superiore alla media della popolazione, compariranno, per le generazioni a venire, in un numero crescente di stringhe, mentre gli schemi con valore adattativo inferiore alla media sono destinati ad estinguersi.

La crescita esponenziale, all'interno della popolazione, degli elementi responsabili di un buon risultato, conduce la popolazione stessa a convergere fulmineamente verso un unico risultato, limitando fortemente le capacità esplorative dell'algoritmo.

Per questa ragione l'algoritmo genetico integra la riproduzione selettiva con la ricombinazione degli elementi migliori; tale ricombinazione è, come sappiamo, il risultato del *crossover*.

4.4.2 Effetto del *crossover*

Si dà il caso che la scissione delle stringhe operata dal *crossover* incorra nel pericolo di distruggere le combinazioni più promettenti contenute negli schemi; si è interessati pertanto ad effettuare una stima delle alterazioni apportate dal crossover alla propagazione degli schemi.

Osservando che uno schema è tanto più esposto all'eventualità di una separazione dei suoi elementi quanto più questi sono distanti, infatti è molto più facile che il punto di *crossover* cada tra due valori in uno schema del tipo 1****1 piuttosto che in uno schema del tipo *11***, si può affermare che uno schema ha una certa probabilità p_d di venire distrutto e una probabilità p_s di sopravvivere pari a :

$$p_d = \frac{\delta(H)}{l-1}$$

$$p_s = 1 - p_d = 1 - \frac{\delta(H)}{l-1}$$

dove con $\delta(H)$ si è indicata la lunghezza dello schema H ovvero la distanza tra l'ultima e la prima posizione identificata.

In particolare se si utilizza una scelta casuale nell'effettuare il *crossover*, con probabilità p_c, la possibilità di sopravvivenza di uno schema diventa:

$$p_s \geq 1 - p_c \cdot \frac{\delta(H)}{l-1}$$

Che si riduce all'espressione precedente quando p_c=1.

4.4.3 Effetto della mutazione

Ulteriore fattore che interviene a condizionare la propagazione degli schemi nella popolazione, è la mutazione; ricordando che essa consiste nell'alterazione casuale, ricorrente con probabilità p_m, del valore di un singolo gene.

La sopravvivenza di uno schema H presuppone la conservazione dei valori associati ad ogni singola posizione; pertanto dato che un singolo allele rimane immutato con probabilità $(1-p_m)$, e dato che ciascuna mutazione è statisticamente indipendente, uno schema sopravvive fino a che ciascuno degli $o(H)$ valori definiti dello schema sopravvive.

Moltiplicando la probabilità $(1-p_m)$ (probabilità di conservazione relativa a ciascun valore in ciascuna posizione) per se stessa $o(H)$ volte, otteniamo la probabilità $p_{s.m.}$ con la quale un intero schema sopravvive alla mutazione pari a:

$$p_{s.m.} = (1-p_m)^{o(H)}$$

Per valori molto piccoli di $p_m (<<1)$, l'espressione precedente può essere approssimata riducendosi a

$$p_{s.m.} = 1-o(H) \cdot p_m$$

L'introduzione della mutazione porta così a dover effettuare alcune considerazioni anche sull'ordine degli schemi.

Mentre per la riproduzione ed il *crossover* gli schemi con più probabilità di sopravvivenza e di crescita risultano legati al valore della *fitness* media e alla lunghezza dello schema $\delta(H)$, la mutazione ha portato a considerare più idonei schemi con basso ordine $o(H)$ ovvero con meno posizioni fissate.

4.5 Il Teorema degli Schemi

Finora si sono considerati i singoli effetti che gli operatori dell'algoritmo apportano ad un particolare schema, ma la riproduzione, il *crossover* e la mutazione sono delle tappe del processo evolutivo messo in atto dall'algoritmo genetico.

Si considerano ora gli effetti combinati dei vari operatori sui modelli di somiglianza. Prendendo la riproduzione ed il *crossover* e considerandoli come indipendenti si può ottenere come l'effetto combinato di questi due operatori vada a

fornire una prima stima del numero di stringhe che un particolare schema *H* rappresenterà nella generazione successiva ottenendo:

$$m(H,t+1) \geq m(H,t) \cdot \frac{f(H)}{f} \cdot \left[1 - p_c \cdot \frac{\delta(H)}{l-1}\right]$$

Paragonando questa espressione a quella fornita nel paragrafo precedente per la sola riproduzione appare chiaro come l'effetto combinato di questo operatore con il *crossover* possa essere valutato andando semplicemente a moltiplicare il numero atteso di schemi della sola riproduzione con la possibilità di sopravvivenza p_s che fornisce il *crossover*.

Lo schema *H* allora crescerà o si estinguerà in base a due fattori: se il valore dello schema è al di sopra o al di sotto della media della popolazione e se ha una lunghezza definita lunga o corta. Chiaramente schemi con valori al di sopra della media generazionale e caratterizzati da una lunghezza relativamente corta riceveranno nella generazione seguente una crescita esponenziale.

Introducendo l'ultimo operatore della mutazione possiamo affermare che uno schema *H* riceverà un numero di copie nella generazione seguente sotto riproduzione, *crossover* e mutazione fornito dalla seguente equazione:

$$m(H,t+1) \geq m(H,t) \cdot \frac{f(H)}{f} \cdot \left[1 - p_c \cdot \frac{\delta(H)}{l-1} - o(H) \cdot p_m\right]$$

Come illustrato nel paragrafo precedente l'aggiunta della mutazione porta a considerare anche l'ordine dello schema.

In conclusione possiamo affermare che:

'*Gli schemi piccoli, di basso ordine e con fitness superiore alla media subiscono una crescita esponenziale nelle generazioni successive di un algoritmo genetico*.'

Questa conclusione è così importante che prende il nome di *Teorema degli Schemi* o *Teorema fondamentale degli Algoritmi Genetici*.

Il teorema così espresso nella forma dell'equazione precedente è un limite inferiore, poiché contempla esclusivamente gli effetti distruttivi di incrocio e mutazione. Tuttavia, l'incrocio è considerato una delle principali fonti della potenza di un algoritmo genetico, per la sua capacità di ricombinare istanze di buoni schemi per formarne di nuovi di ordine superiore altrettanto buoni o in alcuni casi migliori.

La supposizione che questo sia il processo mediante il quale gli algoritmi genetici operano è nota con il nome di 'Ipotesi dei blocchi costitutivi' o '*Building Block Hypothesis*'.

4.6 Ipotesi dei blocchi costitutivi

'*Un algoritmo genetico ottiene delle prestazioni vicine all'ottimo attraverso la contrapposizione di schemi piccoli, di basso ordine e con alte prestazioni, chiamati blocchi costitutivi*.'

Sebbene siano state condotte alcune ricerche per provare queste ipotesi [10, 15], per alcune applicazioni non banali si fa ancora affidamento principalmente su risultati empirici.

Durante gli ultimi anni sono state sviluppate molte applicazioni degli algoritmi genetici che supportavano le ipotesi dei blocchi costitutivi in problemi riguardanti campi molto diversi. Tuttavia queste ipotesi mostrano come il problema della codifica per un algoritmo genetico sia cruciale per la sua riuscita finale, e che tale codifica dovrebbe soddisfare l'idea dei blocchi costitutivi relativamente corti.

L'elaborazione degli schemi di lunghezza significativa ridotta e ad alta *fitness* riduce notevolmente la complessità della ricerca di una soluzione: invece di cercare di costruire stringhe che possiedano un'alta *fitness* provando tutte le possibili combinazioni delle variabili disponibili, gli algoritmi genetici procedono gradualmente combinando le migliori soluzioni parziali degli individui precedenti.

Gli schemi rappresentano quindi i mattoni elementari usati dall'evoluzione per la costruzione di complesse strutture. Affinché la procedura di isolamento e combinazione degli schemi risulti in un graduale miglioramento delle soluzioni offerte è importante che la rappresentazione genetica sia appropriata al problema da risolvere, perché rappresentazioni inadeguate aumentano la complessità e la lunghezza significativa degli schemi necessari a codificare una soluzione accettabile.

Vi sono due regole di base per scegliere una rappresentazione adeguata: la rappresentazione dovrebbe permettere la presenza di schemi rilevanti e corti, e dovrebbe basarsi su di un alfabeto ristretto che permetta un'espressione naturale del

problema. Inoltre, uno schema di codifica ben riuscita è uno schema che incoraggia la formazione di blocchi costitutivi, assicurandosi che i geni correlati siano vicini all'interno del cromosoma e che ci sia poca interazione tra i geni.

Per 'interazione tra geni', o epistassi, si intende che il contributo di un gene alla *fitness* dipende dal valore degli altri geni nel cromosoma.

Se queste regole vengono rispettate, l'algoritmo genetico sarà efficiente come predetto dal teorema degli schemi.

Sfortunatamente le due condizioni sono difficili da incontrare e spesso i geni possono essere correlati in modo che non sia possibile metterli vicino in una stringa monodimensionale.

In molti casi, l'esatta natura del legame tra i geni può non essere nota al programmatore, così anche se sono relazioni semplici, può essere impossibile arrangiare la codifica per mostrarle esplicitamente.

Da qui nascono due domande: se non è possibile rispettare le due regole precedenti, può un algoritmo genetico essere modificato in modo da migliorare il suo funzionamento? Se si, in che modo? Queste domande sono entrambe argomento di ricerca.

Il Teorema degli Schemi e l'Ipotesi dei Blocchi Costitutivi si occupano principalmente del ruolo della selezione e dell'incrocio negli algoritmi genetici. Rimane così da chiarire il ruolo della mutazione. Secondo Holland [14] la mutazione è ciò che impedisce la perdita di diversità in una popolazione data.

Per esempio, senza la mutazione, potrebbe capitare che ogni stringa di una certa popolazione con codifica binaria abbia un '1' nella prima posizione. Di conseguenza non ci sarebbe modo di ottenere una stringa che inizi con uno '0'. La mutazione fornisce una 'polizza di assicurazione' nei confronti di un tale irrigidimento.

4.7 Elaborazione degli schemi: un esempio

Per comprendere meglio l'importanza e la validità della teoria degli schemi si va ora a ripercorrere l'esempio proposto nel paragrafo 4.3 osservando questa volta come gli algoritmi genetici processino gli schemi presenti nella popolazione.

Si considerino tre schemi H_1=1****, H_2=*10**, e H_3=1***0, caratterizzati da lunghezze diverse e da ordini diversi.

Lo schema H_1 rappresentativo delle stringhe 2 e 4 dovrebbe fornire dopo la riproduzione un numero di copie pari a 3 corrispondenti alle stringhe 2, 3, 4. Dalla teoria degli schemi ci si aspettano un numero di copie pari a $m(H,t+1) = m(H,t) \cdot f(H)/f$. Calcolando la media dello schema f(H_1) si ottiene $f(H_1)$= (576+361)/2 = 468.2. Dividendo questo risultato per la media generazionale f=293 e moltiplicando il risultato per il numero di schemi H_1, pari a $m(H_1,t)$=2 presenti al tempo t si può ottenere il numero di schemi presenti al tempo t+1 come segue:

$$m(H_1,t+1) = m(H_1,t) \cdot f(H_1)/f = 2 \cdot 468.5/293 = 3.20$$

Confrontando il risultato con l'effettivo numero di copie ottenute dopo la riproduzione rappresentative dello schema considerato, ovvero 3, si ottiene già un primo riscontro nei risultati. Considerando inoltre che lo schema H_1 è caratterizzato da una lunghezza $\delta(H_1)$=0 si può notare come il *crossover* non possa intervenire nella distruzione di questo particolare schema inoltre vista la percentuale di mutazione utilizzata p_m=0.001 ci si aspetta di avere $m \cdot p_m$= 3·0.001=0.003 o meglio nessun cambio di bit e quindi neanche la mutazione interverrà nelle tre copie dello schema.

Si può così osservare come lo schema H_1 abbia ottenuto una crescita esponenziale nel numero di copie tra una generazione e la successiva come previsto dalla teoria degli schemi. Ovviamente lo schema considerato sembrerebbe essere un caso particolare legato al fatto che esso sia rappresentato da un solo bit.

Si considerino allora lo schema H_2=*10** e lo schema H_3=1***0 e si applichi il Teorema degli Schemi trascurando gli effetti della mutazione che, vista la bassa probabilità utilizzata e il basso ordine degli schemi, in entrambi i casi non fornisce alcun cambiamento di bit, si ottiene:

$$m(H_2,t+1) = m(H_2,t) \cdot \frac{f(H_2)}{f} \cdot \left[1 - p_c \cdot \frac{\delta(H_2)}{l-1}\right] = 2 \cdot \frac{(576+64)}{2 \cdot 293} \cdot \left[1 - \frac{1}{4}\right] = 1.64$$

$$m(H_3,t+1) = m(H_3,t) \cdot \frac{f(H_3)}{f} \cdot \left[1 - p_c \cdot \frac{\delta(H_3)}{l-1}\right] = 1 \cdot \frac{576}{293} \cdot \left[1 - \frac{4}{4}\right] = 0$$

In conclusione lo schema H_2 che in origine aveva due stringhe rappresentative (2,3) alla fine del processo di riproduzione dovrebbe avere:

$$m(H_2, t+1) = m(H_2, t) \cdot f(H_2) / f = 2 \cdot 320 / 293 = 2.18$$

Ovvero 2.18 copie dove 320 è la media delle stringhe rappresentative e 293 è la media generazionale. Lo schema H_3 che partiva con una sola stringa rappresentativa dovrebbe fornire dopo la riproduzione un numero di copie pari a:

$$m(H_3, t+1) = m(H_3, t) \cdot f(H_3) / f = 1 \cdot 576 / 293 = 1.97$$

Considerando però gli l'effetti del *crossover* in queste due particolari circostanze osserviamo che mentre lo schema H_2 caratterizzato da una lunghezza definita di $\delta(H_2)$=3-2=1 rischia di essere distrutto solamente una volta su quattro (l-1=5-1=4) lo schema H_3 caratterizzato da una lunghezza di $\delta(H_3)$=5-1= 4 verrà sicuramente distrutto dal *crossover*.

In conclusione, utilizzando la teoria degli schemi, si è previsto un numero di copie dello schema H_2 pari a due e che lo schema H_3 non fornisca alcuna copia alla fine del processo. Questi risultati sono in completo accordo con quelli ottenuti nel paragrafo 4.4 come visibile dalla tabella 4. Inoltre Il calcolo ha confermato la teoria degli schemi: ovvero che gli schemi corti e di basso ordine hanno subito una crescita o una diminuzione esponenziale nel numero di copie in accordo con il loro valore medio di *fitness*.

Num. Stringa	**Popolaz. iniziale**	**Decod. X**	**Valutazione $F(x)=x^2$**	**Prob. di riprod. $p_i = f_i/\sum f_i$**	**Num.figli atteso $N = p_i \cdot n$**	**Num. Estraz.**
1	01101	13	169	0.14	0.58	1
2	11000	24	576	0.49	1.97	2
3	01000	8	64	0.06	0.22	0
4	10011	19	361	0.31	1.23	1
Somma			1170	1.00	4.00	4
Media			293	0.25	1.00	1
massimo			576	0.49	1.97	2
Elaborazione degli schemi						
Prima della riproduzione						
					Stringhe rappresentative	**Media fitness f(H)**
H_1	1****				2,4	469
H_2	*10**				2,3	320
H_3	1***0				2	576

Accoppiam. con crossover (!)	**Incrocio p_c=1.0 (tutti)**	**Mutazioni p_m=0.001 (nessuna)**	**Nuova popolaz.**	**Decod. x**	**Valutaz. $\Phi(x)=x^2$**
0110!1	01100	01100	01100	12	144
1100!0	11001	11001	11001	25	625
11!000	11011	11011	11011	27	729
10!011	10000	10000	10000	16	256
Somma					1754
Media					439
Massimo					729
Elaborazione degli schemi					
Dopo la riproduzione			**Dopo tutti gli operatori**		
Valore atteso	**Numero effettivo**	**Stringhe rappresent.**	**Valore atteso**	**Numero effettivo**	**Stringhe rappresent.**
3.20	3	2,3,4	3.20	3	2,3,4
2.18	2	2,3	1.64	2	2,3
1.97	2	2,3	0.0	1	4

Tabella 4: Valori prima e dopo la prima generazione.

5. Alcuni aspetti pratici

Nei capitoli precedenti si sono descritti i meccanismi di base di un algoritmo genetico necessari per produrre dei buoni risultati nella maggior parte dei casi. Tuttavia all'interno di questo di capitolo si evidenziano quegli aspetti pratici che, nel corso degli ultimi anni, hanno prodotto molteplici variazioni sui diversi componenti dell'algoritmo.

Si analizzano, inoltre, le principali variazioni legate alle differenti componenti degli algoritmi genetici.

5.1 Mantenere la diversità

Come già messo in evidenza gli algoritmi genetici operano su elementi casuali utilizzando dei processi di transizione probabilistici. Si crea così la necessità di ripetere ogni esperimento più volte iniziando da popolazioni iniziali casuali diverse per assicurarsi la stabilità della soluzione ottenuta. Tanto più piccola è la dimensione della popolazione considerata quanto maggiore sarà il rischio di non riuscire a replicare i dati oppure di non riuscire ad ottenere una soluzione soddisfacente.

Il corretto funzionamento degli algoritmi genetici si basa, infatti, sulla diversità dei cromosomi della popolazione. La mancanza di diversità corrisponde ad una stagnazione non desiderabile né durante il processo evolutivo, né quando la *fitness* media e quella massima si sono stabilizzate, si potrebbero, infatti, verificare delle situazioni per cui continuare il processo evolutivo potrebbe portare a nuove e migliori soluzioni.

Inizialmente la diversità è assicurata dalla scelta casuale delle stringhe che compongono la prima popolazione di tentativo. Nel corso dell'algoritmo il processo di riproduzione selettiva tende a ridurre la diversità facendo convergere tutti i cromosomi, fenomeno che prende il nome di 'deriva genetica'.

L'operatore di *crossover* tende a contrastare la riduzione di della diversità creando nuove strutture dalla combinazione di parti delle stringhe esistenti ma non risulta

sufficiente in tutti i casi. Per contrastare la prematura perdita di un allele in una determinata posizione si introduce l'ultimo operatore ovvero la mutazione.
Tuttavia poiché questo operatore agisce ciecamente si cerca di utilizzarlo con una probabilità molto bassa.
Proprio per queste ragioni si è iniziato a diffondere un nuovo metodo per mantenere la diversità all'interno dell'algoritmo. Questo consiste nell'introdurre ad ogni generazione nuove stringhe casuali all'interno della popolazione sostituendole a quelle che hanno ottenuto dei valori più scadenti.

5.2 La codifica

Un cromosoma è una sequenza di simboli che identifica una possibile soluzione al problema affrontato. Generalmente l'alfabeto scelto per la codifica è il codice binario. Secondo Goldberg la rappresentazione binaria è quella che permette di elaborare il numero maggiore di schemi ed in grado di fornire il più alto grado di parallelismo implicito.
Tuttavia questa non rappresenta l'unica possibilità di codifica. In molte ricerche sono stati infatti utilizzati alfabeti con cardinalità più alte.
Studi empirici sugli alfabeti con alte cardinalità hanno usato cromosomi dove ciascun simbolo rappresenta un intero o un *floating point*. Poiché i dati del problema sono spesso numerici, rappresentare i geni direttamente come numeri, anziché come cifre binarie, può risultare vantaggioso.
Di conseguenza anche gli operatori di *crossover* e di mutazione si possono definire in modo molto più naturale.
Per quel che riguarda il *crossover* alcuni esempi maggiormente utilizzati in queste casistiche risultano essere:

- la media;
- la media geometrica ovvero la radice quadrata del prodotto dei valori dei genitori;
- l'*extension* in cui si prende la differenza tra i due valori e la si aggiunge al più alto o la si sottrae al più basso;

Per la mutazione gli esempi più importanti sono:

- il rimpiazza mento casuale di un valore;
- il '*creep*' in cui si aggiunge o si sottrae un valore casuale;
- il '*geometric creep*' ovvero si moltiplica il gene per un valore prossimo all'unità.

Per questi ultimi due operatori il numero generato casualmente può avere diverse distribuzioni: uniforme all'interno di un intervallo dato, esponenziale, binomiale, gaussiana o altro.

5.2.1 Vari tipi di codifica

Per poter affrontare un problema e per riuscire ad analizzare al meglio le soluzioni fornite dall'algoritmo si cerca di codificare i dati nella maniera più naturale possibile. Ovviamente la scelta della codifica dipenderà dalla natura intrinseca del problema considerato e da essa dipenderanno anche gli altri operatori.

Già nei paragrafi precedenti si è illustrata la codifica secondo valori binari molto utilizzata per la sua semplicità e per il grande numero di schemi che è in grado di fornire.

Un'altra tipologia di codifica è quella per permutazione. Con questa codifica tutti i cromosomi sono rappresentati da una stringa di numeri derivante dalla permutazione dei valori di una stringa originale (figura 14). Essa viene utilizzata in problemi di ordinamento come il problema del commesso viaggiatore (nota).

Stringa Originale	(1,2,3,4,5,6,7,8,9)
Cromosoma A	(1,5,3,2,6,4,7,9,8)
Cromosoma B	(8,5,6,7,2,3,1,4,9)

Figura 14: Codifica per permutazione.

Un ulteriore alternativa all'utilizzo dei numeri binari viene fornita dalla cosiddetta codifica per valore. Essa può essere utilizzata in problemi dove sono presenti valori complicati da codificare. In generale risulta correlata a problemi che prevedono dei valori nei cromosomi quali: numeri reali, caratteri o oggetti complicati (figura 15).

Cromosoma A	(1.2324; 5.3243; 0.4556; 2.3293; 2.4545)
Cromosoma B	ABDJEIFJDHDIERJFDLDFLFEGT
Cromosoma C	*(back), (back), (right), (forward), (left)*

Figura 15: Codifica per valore.

Tale tipologia di codifica viene utilizzata per problemi speciali e nel momento in cui viene adottata prevede necessariamente la costruzione di modelli di *crossover* e mutazione specifici.

5.2.2 *Mapping* dei valori ridondanti

Per problemi che presentano degli intervalli di valori ammissibili c'è il rischio di incorrere in alcune problematiche nella codifica e nelle soluzioni che l'algoritmo fornisce. Ad esempio se si utilizza una codifica binaria, come avviene nella maggior parte delle casistiche, ed il numero dei valori ammessi del problema non è una potenza di due si rischia che alcuni dei valori forniti siano ridondanti ovvero che non corrispondano a nessun valore ammesso. Si prenda, ad esempio, in considerazione un problema caratterizzato da un gruppo di dieci oggetti. Utilizzando una codifica binaria si dovrà necessariamente far riscorso a quattro bit per la codifica dei geni, considerando ad esempio i cromosomi dallo 0000 fino al 1001 che rappresenta il numero 9 tradotto in linguaggio binario. Tuttavia durante il processo di evoluzione gli operatori di *crossover* e di mutazione agiscono in modo casuale e non si può garantire che non si manifestino dati ridondanti come ad esempio, nel caso considerato, la presenza di un 1111 rappresentativo del numero decimale 15 che non sarebbe ammesso come soluzione del problema.

Anche se per i problemi più comuni legati all'utilizzo di algoritmi genetici, caratterizzati da funzioni continue, questo problema non si manifesta e si possono comunque presentare alcune soluzioni:

- scartare il cromosoma non valido e generarne uno nuovo;
- assegnare al cromosoma una penalità che ne abbassi il valore della *fitness*;
- mappare un codice invalido verso uno valido, cioè assegnare ad un cromosoma non valido il valore di quello valido che gli sta più vicino.

Ovviamente nelle prime due soluzioni proposte si rischia di scartare cromosomi con geni interessanti mentre esistono diverse metodologie per sviluppare la terza:

- rimappamento fisso in cui un valore ridondante viene sostituito con un valore fisso con lo svantaggio di attribuire più probabilità di sopravvivenza al particolare cromosoma scelto per la sostituzione;

- rimappamento casuale dove un valore ridondante viene sostituito con uno valido ma scelto casualmente;
- rimappamento probabilistico che è un ibrido delle due tecniche in cui tutti i valori dei geni e non solo quelli ridondanti vengono rimappati verso uno dei valori validi in maniera casuale.

5.3 Funzione *Fitness* e Selezione

Si tratteranno insieme questi due aspetti della strategia evolutiva degli algoritmi genetici in quanto risultano strettamente correlati. Infatti la riproduzione selettiva adottata dagli algoritmi genetici assegna ad ogni stringa un numero di copie nella generazione successiva proporzionale al valore della *fitness* ottenuta.

5.3.1 Funzione *Fitness*

La funzione di *fitness* rappresenta uno degli aspetti fondamentali della teoria degli algoritmi genetici. Entro margini abbastanza grandi, infatti, i meccanismi di base di questo tipo di algoritmi risultano così robusti da non essere considerati dei punti critici. Quello che risulta critico è invece la funzione obiettivo e lo schema di codifica utilizzato. Si cercherà così di creare una funzione di *fitness* che, almeno idealmente, sia il più regolare e piatta possibile in modo che i cromosomi con *fitness* ragionevole siano il più vicino possibile alla soluzione ottimale.

Proprio per far funzionare bene gli algoritmi genetici si cercherà così di creare una funzione di *fitness* che non abbia ne troppi massimi locali ne che siano troppo isolati. Comunque nel costruire la funzione di valutazione si deve cercare di riflettere il più possibile il valore reale che avrà il cromosoma. Per alcuni problemi la costruzione della funzione può essere un passo ovvio, ma il valore reale di un cromosoma non sempre è una quantità utile per guidare la ricerca.

In molti problemi di ottimizzazione ci sono molti vincoli e molti punti nello spazio rappresentano cromosomi non validi e perciò con valore reale nullo. Un esempio può essere la stesura di un orario scolastico in cui si devono assegnare un certo numero di lezioni avendo un numero finito di aule e di docenti. Poiché un algoritmo genetico sia efficace in questo caso si deve creare una funzione dove l'obiettivo di

un cromosoma valido sarà quello di portare la ricerca il più vicino possibile alla soluzione ottimale.

In certi casi sarà dunque necessario considerare dei sotto-obiettivi significativi che ricompensino la ricerca della soluzione ottimale.

Un altro approccio è quello di considerare delle penalità ai cromosomi che forniscono delle soluzioni inadeguate al problema considerato.

Secondo alcuni infatti è più utile considerare quanti vincoli siano violati piuttosto che quanti siano soddisfatti. Una buona funzione di penalità può essere costruita a partire dal costo di completamento stimato, ovvero il costo necessario per far diventare valido un cromosoma che non lo è.

Si può anche ricorrere a delle funzioni approssimate laddove la funzione di *fitness* sia troppo complessa o lenta da valutare. Caso questo dei problemi di carattere aeronautico seppur in una fase di avamprogetto.

L'utilizzo di funzioni semplificate porterà comunque alla convergenza della soluzione vista la robustezza degli algoritmi genetici.

5.3.2 *Fitness* e Selezione

Quando la funzione di *fitness* è discontinua si può presentare il caso in cui alcuni individui ottengono un valore della funzione di valutazione molto elevato mentre tutti gli altri membri della popolazione ottengono delle valutazioni molto basse.

In questo caso la generazione successiva tenderà ad essere dominata da copie genetiche del miglior individuo le quali non avranno possibilità di combinarsi con stringhe diverse ovvero il processo evolutivo tenderà ad uno stato di convergenza prematura verso un valore di minimo locale.

Una volta che la popolazione converge, la capacità dell'algoritmo di continuare la ricerca per trovare una soluzione migliore è praticamente annullata. Il crossover tra individui molto simili apporterà solo ben pochi miglioramenti, in questi casi solo la mutazione permette di esplorare nuove zone nello spazio di ricerca.

La teoria degli schemi afferma che si dovrebbero allocare prove riproduttive in proporzione alla loro *fitness* relativa, ma si può verificare in alcuni casi una convergenza prematura a causa del fatto che la popolazione non è infinita. Per far

lavorare bene gli algoritmi genetici su di una popolazione finita si deve modificare la maniera con cui vengono scelti gli individui per la riproduzione.

L'idea di base è che si deve controllare il numero di opportunità che ogni individuo riceve in modo che non sia ne troppo basso ne troppo alto. L'effetto desiderato è quello di restringere il *range* della funzione obiettivo in modo da impedire che un qualsiasi individuo '*superfit*' possa improvvisamente prevalere.

In altri casi potrebbe accadere la situazione opposta ovvero tutte le stringhe riportano valori di *fitness* molto simili, per cui quasi tutti gli individui ottengono un numero simile di copie. In questo caso il processo evolutivo tenderà a stagnare diventando simile ad un processo di ricerca casuale.

In questi due casi limite l'individuo migliore potrebbe addirittura fallire nel riprodursi e qualsiasi soluzione, anche raggiunta in modo più o meno casuale, sarà persa nel ricambio generazionale. Esistono alcuni metodi per far fronte a queste problematiche [16, 17].

5.3.3 Tecniche di selezione

La selezione degli individui utilizzati per la riproduzione ha lo scopo di allocare opportunità riproduttive a ciascun individuo. In principio, gli individui vengono copiati dalla popolazione in una vasca di accoppiamento (*mating pool*) seguendo uno schema probabilistico. Sotto un severo schema di riproduzione, la dimensione della vasca è uguale a quella della popolazione. Terminata questa prima fase tutti gli individui presenti nella vasca vengono estratti a coppie e subiscono l'operatore della riproduzione.

Il comportamento di un algoritmo genetico dipende in larga misura da come gli individui vengono scelti per andare nella *mating pool*. Il metodo più utilizzato, e già descritto nei capitoli precedenti, è quello della 'ruota della fortuna truccata'. Esistono numerosi altri sistemi per selezionare in maniera probabilistica i 'genitori': i più rappresentativi sono i metodi ottenuti tramite una rimappatura della *fitness*.

Si parla di *fitness* rimappata esplicitamente se si effettua una qualche operazione di normalizzazione sui valori grezzi al fine di migliorare l'evoluzione dell'algoritmo. I principali metodi di rimappamento esplicito sono:

- *fitness scaling*: i valori decimali di *fitness* vengono rimappati a valori interi rapportandone il valore massimo con quello medio e stabilendo a priori un massimo numero di prove riproduttive per ogni individuo (di solito due);
- *fitness windowing* [18]: simile al precedente ma viene considerata la fitness minima nelle normalizzazioni dei valori. Consiste nel trovare il valore minimo della popolazione e assegnare a ciascuna stringa un nuovo valore di fitness uguale alla differenza tra quello della stringa ed il valore minimo;
- *fitness ranking*: gli individui vengono ordinati a valori di *fitness* crescenti, dal peggiore (numero 1) al migliore (numero N). Solo allora i valori di *fitness* riproduttivo verranno assegnati in accordo al rango. Questo può essere fatto linearmente o esponenzialmente e dà un risultato simile al *fitness scaling*: in questo il rapporto massimo/media della funzione obiettivo è normalizzato a un particolare valore. Inoltre ci assicura che il *fitness* rimappato di un individuo intermedio sarà regolarmente propagato. A causa di questo effetto di uno o due individui estremi sarà trascurabile, indipendentemente da quanto piccolo o grande sia il valore della loro *fitness* in rapporto al resto della popolazione, in questo modo i migliori cromosomi non differiscono molto dagli altri e questo può rallentare molto la convergenza. Il numero di prove riproduttive allocate, per esempio, sarà sempre lo stesso qualunque sia il valore di *fitness*. Alcuni esperimenti portano a concludere che questo metodo sia comunque migliore del *fitness scaling*.

Si parla invece di rimappamento implicito della *fitness* se si selezionano i cromosomi da riprodurre (i genitori) senza passare attraverso livelli intermedi di rimappamento. La tecnica più rappresentativa di rimappamento implicito è la '*Tournament Selection*': nella sua variante più semplice ovvero la 'selezione binaria di torneo', coppie di individui sono prese a caso tra la popolazione e confrontate. Il vincitore viene copiato nella *mating pool*: il processo si ripete fino al riempimento della vasca. Si possono anche creare tornei più grandi in cui il confronto viene effettuato non tra due individui ma tra n individui presi a caso dalla popolazione. Questo ha l'effetto di aumentare la pressione di selezione, perché gli individui sotto

la media difficilmente vinceranno tornei mentre i migliori avranno ottime possibilità.

Un ulteriore generalizzazione è la 'selezione con torneo binario statistico', dove i migliori individui vincono i tornei con probabilità p dove $0.5 < p < 1$. Usando valori più bassi di p si diminuisce la pressione di selezione, perché gli individui sotto la media sono in proporzione più avvantaggiati nel vincere un torneo, mentre quelli sopra la media perdono probabilità. Aggiustando la probabilità di vincere o la dimensione del torneo, la pressione di selezione può essere resa grande o piccola a piacere.

Studi hanno confermato che tramite opportuni aggiustamenti dei parametri tutte queste tipologie di selezione hanno performance simili e che quindi non esiste una tecnica migliore in assoluto.

5.3.4 Elitismo e *Steady-State Reproduction*

Le tecniche di riproduzione viste fin'ora prevedono che la popolazione sia sostituita completamente ad ogni generazione. Questa impostazione porta però qualche problema: se per esempio il migliore individuo non riesce a riprodursi, nel ricambio generazionale si ha una perdita di informazioni.

Sono state dunque studiate tecniche che permettono di ridurre perdite consistenti di informazioni. Una di esse prevede che la migliore stringa di ciascuna generazione venga ricopiata immutata nella generazione successiva. Questa tecnica definita 'strategia elitista', svolge due importanti funzioni: assicura che il cromosoma migliore sia presente nella generazione successiva anche se il processo di riproduzione probabilistica fallisce nel riprodurlo e preserva la struttura genetica nel caso in cui l'operatore di *crossover* distrugga lo schema ottimale sottostante. Siccome solitamente la strategia elitista produce un miglioramento del processo evolutivo, essa viene impegnata nella maggior parte degli algoritmi genetici assieme ad un processo di normalizzazione della *fitness* per evitare che l'individuo migliore domini la popolazione.

Una seconda tecnica consiste nel modificare l'operatore di riproduzione in modo da rimpiazzare solamente pochi individui alla volta: questa tecnica viene definita con il

nome di ‘riproduzione a regime stazionario’ [19, 20]. Il processo di creazione di nuovi individui è identico a quello della ruota della fortuna ma in questo caso vengono create solo le prime *k* copie (con *k* minore della grandezza della popolazione) che vengono poi incrociate, mutate e sostituite ai k individui peggiori della popolazione. Una versione più efficiente della riproduzione a regime costante consiste nel gettar via le stringhe che sono uguali a quelle già presenti nella popolazione(‘riproduzione a regime stazionario senza duplicazione’): questa soluzione assicura costantemente la diversità della popolazione allontanando così il pericolo di una convergenza prematura.

5.4 *Crossover*

Come per gli altri operatori anche per il crossover esistono numerose tecniche. Nei capitoli precedenti si è illustrato il così detto ‘*single-point crossover*’ o ‘*crossover* a taglio singolo’ in cui viene selezionato casualmente un punto attorno al quale viene scambiato il corredo genetico.

Esistono diverse versioni di ‘*crossover* a taglio multiplo’ dove vengono selezionati diversi punti casuali di scambio genetico. Questa soluzione permette un maggior numero di gradi di libertà nella ricombinazione di soluzioni parziali in strutture di complessità maggiore e potrebbe accelerare il processo evolutivo. Questo operatore diventa interessante anche quando la stringa genetica di un individuo codifica diversi parametri non strettamente collegati fra loro: in questo caso la mescolanza dei geni potrebbe risultare in un’evoluzione meno efficiente o più lunga per cui conviene effettuare tagli ed incroci separati all’interno delle rispettive aree genetiche.

Il *crossover* svolge un ruolo molto importante nella teoria degli schemi ed ha tradizionalmente occupato una posizione centrale negli algoritmi genetici; tuttavia recentemente questa tradizione è stata messa in discussione. Innanzitutto la teoria degli schemi è valida solamente in un contesto limitato, in quanto assume un alfabeto genetico discreto e finito: questo non significa che gli algoritmi genetici falliscano in tutti gli altri contesti ma che in questi casi il processo di elaborazione degli schemi non è dimostrato. In secondo luogo, esiste un vasto corpus di

esperimenti recenti su codici genetici composti da numeri reali in cui non viene utilizzato il *crossover*, ma solamente la riproduzione selettiva e una probabilità maggiore di mutazione.

5.4.1 *Crossover* a due punti

In questa tecnica (e in generale nel *crossover* a taglio multiplo) piuttosto che stringhe lineari i cromosomi possono essere considerati come circoli, formati unendo gli estremi del cromosoma (figura 16).

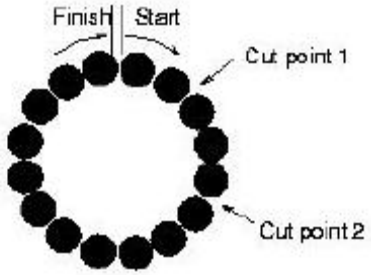

Figura 16: *Crossover* a due punti.

Per cambiare un segmento di un circolo con un altro si richiede la selezione di due punti di taglio, come mostrato in figura. In questo modo si può considerare il *crossover* a taglio singolo come un *crossover* a due punti ma con un punto di taglio fissato all'inizio della stringa. Quindi il *crossover* a due punti opera allo stesso modo di quello a taglio singolo che ne rappresenta una generalizzazione.

5.4.2 *Crossover* uniforme

Ciascun gene dei figli è creato tramite una copia del corrispondente gene da uno dei due genitori, scelto in accordo con una 'maschera di crossover' creata in maniera casuale (figura 17).

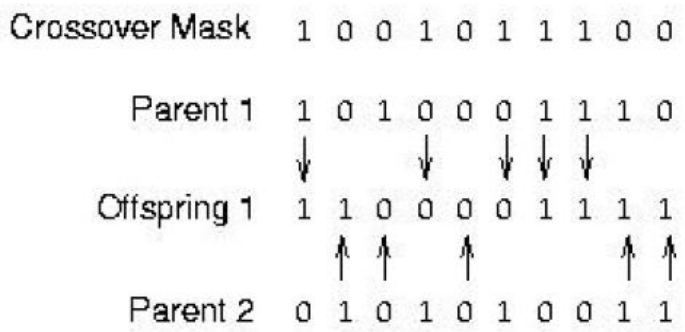

Figura 17: *Crossover* uniforme.

Dove c'è un '1' nella maschera, il gene viene copiato dal primo genitore, mentre dove c'è uno '0', il gene viene copiato dal secondo genitore. Il processo è ripetuto con i genitori scambiati per produrre un secondo figlio. Una nuova maschera di *crossover* viene generata casualmente per ciascuna coppia di stringhe genitrici. Si creano così delle nuove stringhe che racchiudono i patrimoni genetici delle progenitrici. Si può vedere questa tecnica come un *crossover* multiplo in cui il numero di tagli non è impostato a priori.

5.4.3 Confronto tra tecniche

Determinare quale sia la tecnica migliore tra le precedenti non risulta affatto semplice e rappresenta a tutt'oggi un argomento di dibattito.

Il *crossover* uniforme tende a distruggere gli schemi che hanno un particolare ordine (ovvero un certo numero di posizioni fisse) a prescindere dalla loro lunghezza di definizione (distanza tra il primo e l'ultimo valore fissato nella stringa).

Con il *2-point crossover* è la lunghezza di definizione dello schema che determina la sua predisposizione alla distruzione, non il suo ordine. Questo significa che rispetto al *2-point*, nel *crossover* uniforme gli schemi con lunghezze di definizione corte hanno maggiori probabilità di essere distrutti, mentre, per quel che riguarda schemi più lunghi vale il contrario.

Il *crossover* uniforme ha il vantaggio che l'ordinamento dei geni è del tutto irrilevante e questo significa che non ci si deve preoccupare di posizionare i geni per migliorare i blocchi costitutivi. L'efficienza dell'algoritmo genetico che utilizza il *crossover* a taglio multiplo cade drasticamente se non vengono rispettate le raccomandazioni della teoria degli schemi.

Il *crossover* a taglio multiplo dà buoni risultati quando vengono utilizzati due punti di taglio ma aggiungere più punti riduce le prestazioni dell'algoritmo e così facendo si rischia più facilmente di distruggere gli schemi racchiusi nella stringa. Al contrario, un vantaggio di avere molti punti *crossover* è che nello spazio del problema si può fare una ricerca più accurata.

Non è chiaro quindi quale tecnica sia la migliore pur essendoci molti studi che analizzano le differenze e le presunte superiorità di un metodo rispetto agli altri, senza però raggiungere un consenso comune.

5.5 Mutazione ed Evoluzione

La mutazione è tradizionalmente vista come un operatore secondario, responsabile di una reintroduzione inaspettata di valori di geni perduti che prevengono la deriva genetica, fornendo un piccolo elemento di ricerca casuale nella vicinanza della popolazione dove esso è largamente convergente. In generale, secondo Goldberg, la mutazione rappresenterebbe una sorta di polizza assicurativa nei confronti di una perdita prematura di informazioni, che può essere causata dagli altri operatori, e permetterebbe una ricerca più vasta e completa all'interno dello spazio considerato.
Tuttavia viene considerato come un operatore secondario, infatti, si ritiene che il *crossover* sia la principale forza che guida la ricerca.
Malgrado questa posizione, la mutazione viene considerata come la principale fonte di materiale per cambiamenti evolutivi. Gli esperimenti che sono stati fatti sull'ottimizzazione dei parametri per gli algoritmi genetici hanno mostrato che il *crossover* produce molti meno effetti di quanto si pensava mentre l'evoluzione *naive* (solo selezione e mutazione) agisce in modo simile all'*hillclimb* e può essere potente anche senza il *crossover*. Il *crossover* infatti produce evoluzioni più veloci rispetto a una popolazione che subisce solo mutazione, anche se la mutazione, fornisce alla fine del processo evolutivo soluzioni migliori. Mentre la popolazione si avvicina alla convergenza la mutazione diventa più produttiva del *crossover*.

6. Esempio di utilizzo in ambito aeronautico

Nei precedenti capitoli sono stati descritti ed approfonditi i meccanismi con cui operano gli algoritmi genetici: si va ora ad analizzare l'utilizzo di queste tecniche in ambito aeronautico ed in particolare verrà descritto un esempio in cui risultano applicati alla progettazione preliminare di un velivolo.

6.1 Studi in ambito aeronautico

Vista la robustezza degli algoritmi genetici, questi sono stati ampiamente utilizzati in diversi ambiti delle scienze, dalla fisica all'economia [10]: in particolare, vista la possibilità di adattamento che mostrano queste tecniche di ottimizzazione, esse vengono applicate in ogni contesto che richieda l'ausilio di tecniche di ottimizzazione multidisciplinare.

Si possono così trovare lavori che vanno dalla progettazione preliminare di un velivolo [21, 22] fino all'utilizzo di tali tecniche per la modellazione di superfici aerodinamiche [23, 24].

Oltre al caso degli aeromobili, non mancano studi sugli elicotteri come quello presentato in [25] in cui si presentano tre possibili configurazioni ottenute attraverso l'utilizzo degli algoritmi genetici.

Nel contesto della progettazione preliminare di velivolo, l'impiego di algoritmi genetici consente di definire una combinazione di primo tentativo delle variabili dimensionali e progettuali che definiscono la struttura caratteristica di un aeromobile.

In particolare è possibile ottimizzare un numero elevato di variabili [21] per riuscire a fornire una configurazione completa del velivolo analizzandone l'evoluzione attraverso configurazioni intermedie, che rappresentano i risultati forniti dall'algoritmo ad ogni generazione di individui (figura 18).

Inoltre, come visibile in figura 18, i risultati possono essere confrontati con altri ottenuti con delle scelte diversi nell'implementazione dell'algoritmo.

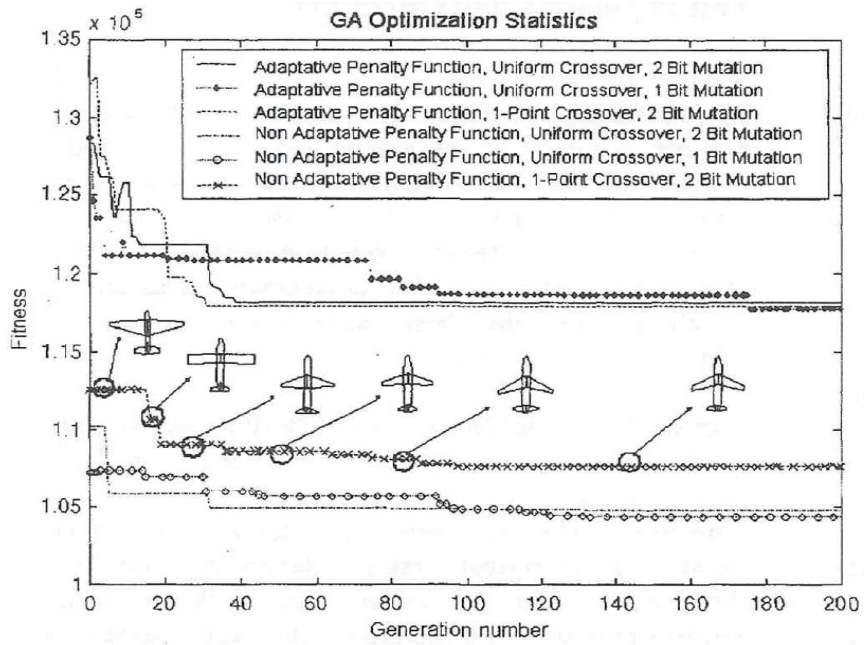

Figura 18: Convergenza dell'algoritmo verso la soluzione ottima [21].

Ovviamente tali configurazioni sono il risultato di un processo di ottimizzazione che coinvolge un certo numero di variabili (in questo caso 16) che vanno dal carico alare fino al numero di passeggeri che si vuole imbarcare (figura 19).

	Run 1	Run 2	Run 3	Run 4	Run 5	Run 6	B-717
Penalty Function	Adapt.	Adapt.	Adapt.	Fixed	Fixed	Fixed	
Crossover Type	Uniform	Uniform	1-Point	Uniform	Uniform	1-Point	
Mutation Type	2 Bit	1 Bit	2 Bit	2 Bit	1 Bit	2 Bit	
W/S	103.76	108.98	109.25	133.14	136.98	139.73	115
T/W	0.30	0.326	0.319	0.30	0.30	0.319	0.365
$C_{l_{max}}$	1.5	1.5	1.5	1.4	1.4	1.0	1.4
A_W	6.57	6.28	6.38	6.76	6.38	6.14	8.7
λ_W	0.32	0.43	0.35	0.14	0.62	0.42	0.38
Λ_W	25.9	28.4	26.9	32.5	24.6	25.7	26.9
t/c	0.11	0.11	0.11	0.12	0.11	0.11	0.11
A_{HT}	3.26	3.06	3.39	3.00	3.00	3.90	5.21
λ_{HT}	0.44	0.57	0.41	0.25	0.49	0.39	0.4
Λ_{HT}	25.47	27.05	29.57	5.31	12.56	16.02	36.87
T_{Type}	T Tail	T Tail	T Tail	T Tail	T Tail	T Tail	T Tail
A_{VT}	1.71	0.83	0.91	0.70	1.25	1.41	1.05
λ_{VT}	0.82	0.58	0.58	0.46	0.30	0.90	0.8
Λ_{VT}	41.3	39.1	35.8	38.0	39.4	50.6	41
N_{eng}	2	3	3	3	2	4	2
N_{seats}	5	4	4	4	4	3	5
W_o (lb)	108607	107692	107681	104810	104346	107663	115000
Constraints Violation %	9.4%	10.4%	10.4%	14.9%	17.2%	19.4%	

Note: Adapt. – Adaptative Penalty Function,
Fixed – Traditional Penalty Function

Figura 19: Risultati dell'ottimizzazione effettuata dall'algoritmo [21].

In tale lavoro ([21]) è stata scelta come funzione obiettivo il peso del velivolo cercando di ottenere un aeromobile che fosse il più leggero possibile.

Si sono, inoltre, imposti dei vincoli basati sia sulle performance di aeromobili simili a quello desiderato, ovvero un aeromobile commerciale a medio raggio, che sul tipo di certificazioni necessarie.

Le variabili sono poi state mappate all'interno di stringhe utilizzando un codice binario (figura 20) per poi venire raggruppate all'interno di un cromosoma che rappresenta la soluzione cercata.

Variable		Variable Description	Variable Range	String Length Bits
W/S	(c)	Wing Loading (lb/ft^2)	70 - 140	8
T/W	(c)	Thrust Weight Ratio	0.3 - 0.5	5
$C_{l_{max}}$	(c)	Maximum Airfoil Lift	1 - 1.5	3
AR_w	(c)	Wing Aspect Ratio	6 - 12	7
λ_w	(c)	Wing Taper Ratio	0.1 - 0.9	6
Λ_w	(c)	Wing Sweep Angle (deg)	0 - 50	9
t/c	(c)	Wing Thickness	0.1 - 0.15	3
A_{ht}	(c)	Horizontal Tail Aspect Ratio	3 - 5	5
λ_{ht}	(c)	Horizontal Tail Taper Ratio	0.2 - 0.6	4
Λ_{ht}	(c)	Horizontal Tail Sweep Angle (deg)	5 - 45	7
A_{vt}	(c)	Vertical Tail Aspect Ratio	0.7 - 2	1
λ_{vt}	(c)	Vertical Tail Taper Ratio	0.3 – 0.9	5
Λ_{vt}	(c)	Vertical Tail Sweep Angle (deg)	35 - 55	4
N_{eng}	(i)	Number of Engines	2 - 4	7
T_{Type}	(d)	Tail Type (T Tail – 0, Conventional Tail - 1)	0 - 1	2
N_{seats}	(d)	Seating Arrangement (1-11)	2 - 11	4

Variables Note: c- continuous, i- Integer, d- Discrete

Figura 20: Variabili di progetto [21].

I valori di queste variabili ed il loro intervallo ammissibile sono stati presi da dei dati statistici di aeromobili simili [8]. E' stata poi implementata una funzione di penalità in caso di violazione dei vincoli imposti.

Per quanto riguarda l'algoritmo è stata utilizzata una percentuale di *crossover* dell'80% e di mutazione dell'1%.

Infine per verificare e confrontare i risultati è stato preso come esempio un Boeing 717 con le sue caratteristiche prestazionali e come normativa la FAR25 che descrive, per l'appunto, questo tipo di velivoli .

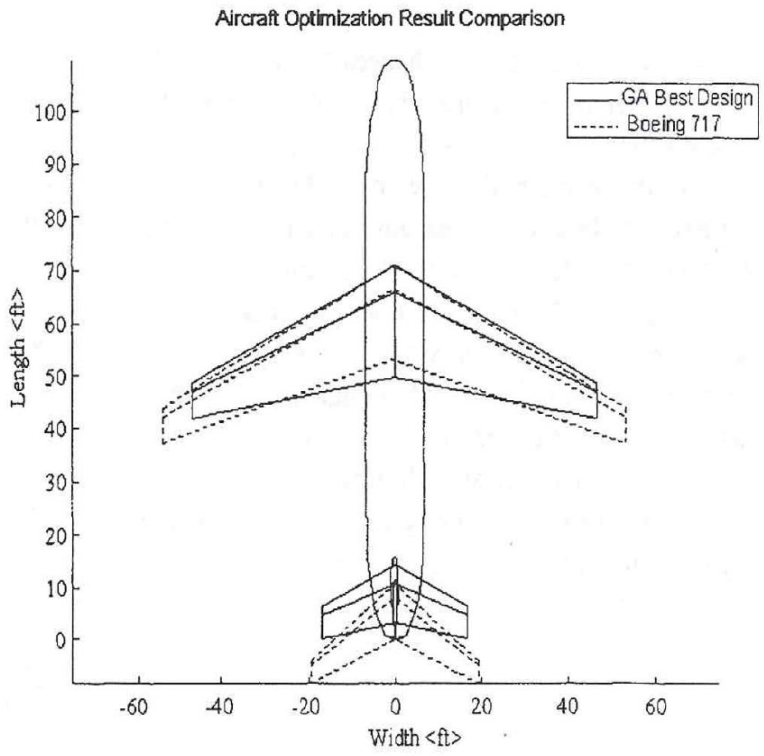

Figura 21: Confronto grafico tra i risultati dell'algoritmo genetico ed un Boeing 717 [21].

Come è possibile osservare in figura 21 i risultati ottenuti con l'utilizzo degli algoritmi genetici non appaiono molto lontani da configurazioni molto più avanzate come quella del Boeing. In particolare si possono notare delle somiglianze negli angoli di freccia utilizzati e nell'area delle superfici.

Si riesce persino ad avere in termini di peso un guadagno del'5%, a parità di carico ed utilizzando uno stesso profilo di missione.

Tutto ciò mostra sommariamente come un algoritmo genetico possa essere utilizzato in ambito aeronautico nella fase preliminare di un progetto.

Si andrà allora a studiare, molto più nel dettaglio, un esempio di applicazione di queste metodologie di ricerca.

In particolare nel lavoro di Andre C. Matra [26] i parametri più importanti della fase progettuale sono stati mappati all'interno di un cromosoma. Le variabili del progetto sono state accorpate per creare delle stringhe che caratterizzeranno i requisiti fondamentali da seguire in fase di progetto.

Tra le variabili incluse sono state considerate la geometria della fusoliera, dell'ala e del piano di coda, i requisiti di spinta necessari e i vari parametri operativi.
Gli operatori dell'algoritmo genetico sono stati eseguiti su una certa popolazione di stringhe e si è proceduto a successivi processi di selezione naturale fino al raggiungimento del risultato desiderato.
Sono stati infine testati sia diversi parametri all'interno dell'algoritmo, sia diversi metodi di selezione (*fitness* e *ranking*).
E' stato in ultimo valutato l'impatto che le diverse scelte hanno comportano in termini di efficienza dell'algoritmo e verrà studiata quella che sarà l'implementazione dei vincoli del progetto.

6.2 Studio parametrico di un algoritmo genetico utilizzando un problema di ottimizzazione per il progetto di un velivolo

Il lavoro svolto da Andre C. Marta [26] parte da una semplice considerazione: a prima vista, la definizione di miglior aeromobile è molto semplice, il più veloce, il più efficiente, il più silenzioso, il più economico e capace di decollare nel minor spazio possibile. Sfortunatamente un aereo del genere non esiste.
Infatti non si può pensare di costruire un aereo molto economico e pensare che sia anche il più veloce o che il più efficiente sia anche il più confortevole.
Si può però pensare di ottimizzare un'unica caratteristica e scendere a compromessi nelle altre (figura 22).

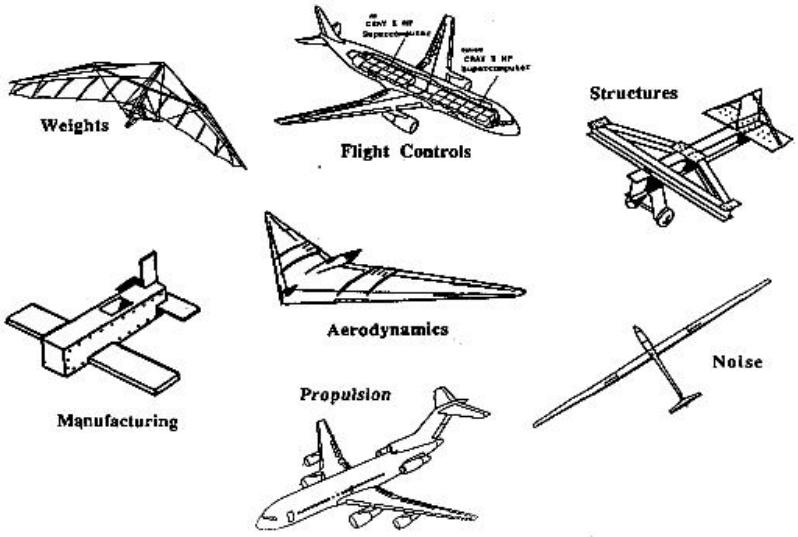

Figura 22: Esempi di ottimizzazione di un'unica caratteristica [26].

In altre parole il progetto globale di un aeromobile è sempre un compromesso di vari aspetti che devono coesistere.

Si deciderà così la misura in cui le diverse caratteristiche come il peso, i controlli di volo, le strutture, la produzione, l'aerodinamica, il rumore e le caratteristiche propulsive debbano pesare all'interno del progetto in base all'obiettivo principale per cui è pensato l'aeromobile.

Una volta che si è scelto l'obiettivo si cerca di far interagire tutte le caratteristiche di progetto per il raggiungimento dello scopo.

Questo processo può essere schematizzato come in figura 23, in cui si può vedere come le variabili di progetto vadano a definire un modello che dovrà soddisfare determinati obiettivi e che sarà necessariamente soggetto a dei vincoli.

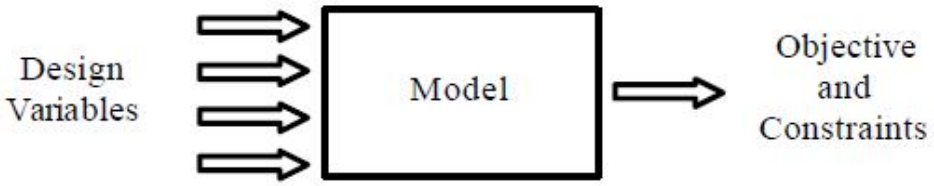

Figura 23: Modellazione [26].

Visto il numero di parametri necessari a definire completamente un aeromobile questo processo di modellazione può risultare un problema estremamente complesso. Per ridurre il numero delle variabili di progetto si cerca così di ricorrere

ad una combinazione di approssimazioni dettate sia dall'esperienza che dalle informazioni statistiche derivanti da progetti simili.

6.2.1 *Background*

Nel campo del progetto aeronautico sono stati fatti numerosi lavori di ottimizzazione, partendo da studi parametrici molto semplici fino ad arrivare a delle configurazioni molto dettagliate che prevedono l'utilizzo di migliaia di variabili di progetto.

All'attuale stato dell'arte, e tralasciando gli studi preliminari, l'attività di ricerca in ambito aeronautico si sta spingendo sempre di più verso una soluzione che permetta di ottimizzare l'intero progetto. Infatti si cerca di agire più che sulle caratteristiche specifiche su quella che viene definita ottimizzazione multidisciplinare (MDO *Multi Disciplinary Optimization*).

Tipicamente questo tipo di indagine richiede l'utilizzo dei metodi del gradiente che richiedono un elevato costo computazionale per determinare le derivate numeriche così come si può vedere nell'indagine di Alonso,Martins e Reuther (2003) [27].

Come messo in luce nei capitoli precedenti quando la funzione di fitness non è continua, i metodi che si basano sull'utilizzo del gradiente non possono essere applicati immediatamente ed in qualche caso è necessario considerare l'utilizzo di altri algoritmi di ricerca.

In ambito aeronautico gli algoritmi genetici seppur di comprovata efficienza e robustezza, sono ancora lontani dall'essere comunemente usati.

6.2.2 Descrizione del problema

Per poter descrivere come l'utilizzo di algoritmi genetici possa risolvere problematiche di carattere aeronautico all'interno del lavoro svolto da Andre C. Marta si riporta un problema di progetto preliminare in cui vengono usate un numero ristretto di variabili e come funzione di *fitness* l'equazione di Breguet.

Questa equazione una volta fissate le caratteristiche propulsive, aerodinamiche e strutturali esprime l'autonomia chilometrica R (*range*), ovvero la distanza

percorribile dal velivolo per una assegnata quantità di carburante consumato [28]. Per un programma di volo ad assetto e velocità costante si riduce alla forma:

$$R = \frac{V}{SFC} \cdot \frac{L}{D} \cdot \ln\left(\frac{W_i}{W_f}\right)$$

Dove V rappresenta la velocità di crociera, SFC il consumo specifico, ovvero la quantità di carburante necessaria nell'unità di tempo a generare un chilogrammo di spinta [28], L è la portanza, D è la resistenza, W_i e W_f rappresentano rispettivamente il peso iniziale e finale del velivolo.

Il termine a destra nell'equazione può essere ricavato attraverso diverse stime così come descritto in Kroo (2003) [29], attraverso l'utilizzo delle seguenti variabili di progetto: diametro e lunghezza della fusoliera, altitudine di crociera, peso al decollo, apertura e corda alare, apertura del piano di coda orizzontale e verticale, angolo d'attacco, posizionamento delle ali e del piano di coda.

Come analizzato in precedenza, ogni aeromobile viene progettato cercando di far coesistere i diversi vincoli imposti dalle diverse specifiche che vengono richieste.

In questa trattazione è stato preso ad esempio un aeromobile di medie dimensioni, come quelle dei tipici jet regionali, che vola ad una velocità di crociera subsonica.

Si sono imposti sette vincoli in aggiunta all'intervallo di valori ammessi per ognuna delle undici variabili di progetto scelte, i cui valori sono riportati in tabella 5.

Le variabili di progetto scelte in aggiunta ad altre dieci costanti (*fuselage nose and tail fineness*, *seat pitch*, fattore di Oswald, angolo di svergolamento dell'ala, spessore percentuale, ovvero il rapporto tra spessore e corda del profilo, allungamento alare dei piani di coda, orizzontale e verticale, numero di motori e consumo specifico al livello del mare) permettono di calcolare tutte le caratteristiche necessarie per valutare in maniera completa l'equazione di Breguet.

Design variable	Admissible values
Fuselage diameter	60 < d < 200 in
Fuselage length	70 < l < 150 ft
Cruise altitude	10,000 < h < 50,000 ft
Take-off weight	60,000 < tow < 100,000 lbs
Wing span	50 < b < 150 ft
Wing chord	5 < c < 20 ft
Horizontal tail span	20 < bh < 70 ft
Vertical tail span	5 < bv < 20 feet
Angle of attack	1 < alpha < 16 deg
Location of wing	0.2 < xwoverl < 0.7
Location of tail	0.5 < xtoverl < 1.0

Others constraints	Admissible values
Number of seat	60 < seats < 80
Mach number	M < 0.5
Tail after wing	xwoverl < xtoverl
Horizontal tail volume	0.4 < Vh < 1.0
Vertical tail volume	0.04 < Vv < 0.09
Moment coefficient about c.g.	-0.1 < cm cg < 0.1
Zero fuel weight	zfw < tow

Tabella 5: Variabili di progetto e vincoli [26].

6.2.3 Metodi utilizzati

Le undici variabili di progetto sono così state mappate all'interno di un cromosoma. Per raggiungere il risultato desiderato sono stati utilizzati diversi pesi, o *granularities*, per ognuna delle variabili considerate corrispondenti a gruppi di bit di lunghezza diversa. Unendo tutti i gruppi di bit si è generato un cromosoma della lunghezza di 71 bit che rappresenta la soluzione progettuale del problema.

Una mappa dettagliata che riporta il nome della variabile di progetto, i valori ammessi, la granularità, il numero di bit e la posizione dei bit nel cromosoma completo può essere osservata nella tabella 6.

Design variable	Admissible values	Granularity	Number of bits	Bit position in chromosome
Fuselage diameter	60 < d < 200 in	128	7	1 to 7
Fuselage length	70 < l < 150 ft	128	7	8 to 14
Cruise altitude	10,000 < h < 50,000 ft	128	7	15 to 21
Take-off weight	60,000 < tow < 100,000 lbs	128	7	22 to 28
Wing span	50 < b < 150 ft	256	8	29 to 36
Wing chord	5 < c < 20 ft	128	7	37 to 43
Horizontal tail span	20 < bh < 70 ft	128	7	44 to 50

Vertical tail span	5 < bv < 20 feet	64	6	51 to 56
Angle of attack	1 < alpha < 16 deg	32	5	57 to 61
Location of wing	0.2 < xwoverl < 0.7	32	5	62 to 66
Location of tail	0.5 < xtoverl < 1.0	32	5	67 to 71

Tabella 6: Composizione del cromosoma del problema [26].

Dalle specifiche identificate nel paragrafo precedente e dalla mappa fornita nella tabella 6 si può ricavare il tableau del problema così come si può vedere nella tabella 7.

Objective:	Find the globally optimum combination of eleven aircraft parameters
Representation scheme:	• Structure = fixed length string; • Alphabet size K = 2 (binary); • String length L = 71; • Mapping from points in search space of the problem to structures in the population = (see table 2)
Fitness cases:	Only one.
Fitness:	Aircraft range with penalty for constraint violations. More is better.
Parameters:	• Population size M = 20; • Maximum number of generations G = 100.
Termination criteria:	The genetic algorithm has run for G generations.
Result designation:	The best-so-far individual in the population.

Tabella 7: Tableau del problema [26].

A questo punto è necessario soffermarsi sull'uso che viene fatto dei vincoli di progetto. Infatti, i vincoli che vengono presi in considerazione vanno a penalizzare il valore della *fitness*.

Si sono presi in considerazione due diverse tipologie di vincolo: severi e non-severi. I primi includono la posizione della coda dopa l'ala e il peso a secco del velivolo meno il peso al decollo (intendendo positivo il peso del carburante). Se uno di questi vincoli viene violato allora sarà assegnato alla funzione di *fitness* il valore di

zero, se invece vengono violati i non-severi, la funzione di *fitness* decrescerà in maniera proporzionale alla deviazione del vincolo rispetto al valore desiderato.
Un approccio iniziale è stato effettuato considerando ogni vincolo come severo, ma così facendo, si è ridotto considerevolmente il numero di individui idonei nella popolazione iniziale generata casualmente (meno del 5% di individui validi), causando un inaccettabile perdita dell' informazione genetica.
Si è così scelto un numero di popolazioni tale da avere un corredo genetico il più ampio possibile andando così a migliorare l'efficienza dell'algoritmo. Sono state fatte diverse prove partendo da una popolazione di 50 individui ma solo quando si è passati a considerarne 200 si è riuscito ad ottenere un numero accettabile di individui validi.
Il massimo numero di generazioni è stato determinato sperimentalmente in modo che la *fitness* dell'individuo migliore all'interno della generazione ha raggiunto un valore accettabile.
L'algoritmo genetico usato da Andre C.Marta per implementare il problema è stato il *Genetic Search Implementation System* (GENESIS *version* 5.0) di John J.Grefenstette [30, 31]. Questi costituisce un ottimizzatore indipendente, nel quale l'utente deve solamente fornire una stima della funzione di *fitness* per il particolare problema considerato (per una documentazione più dettagliata si rimanda a Grefenstette (1984) [30, 31]).
Quindi una volta codificata la funzione di *fitness* utilizzando come input le variabili di progetto riportate nella tabella 5, si è riusciti ad ottenere una stima dell'equazione di Breguet.

6.2.4 Risultati

Per poter valutare l'influenza che i vari parametri hanno sull'algoritmo genetico, sono state effettuate diverse prove all'interno delle quali sono stati inclusi diversi semi per il generatore di numeri casuali (nota o elimina), diversi *crossover*, diverse probabilità di mutazione e diverse procedure di selezione.
Sono state effettuate in tutto sei diverse prove i cui risultati sono riportati nella tabella 8.

Case	Random seed	Crossover probability	Mutation probability	Selection procedure
1	1234567	60 %	3.3 %	Relative fitness spinning wheel
2	123456789	60 %	3.3 %	Relative fitness spinning wheel
3	123456789	60 %	3.3 %	Ranking algorithm R=0.75
4	123456789	60 %	3.3 %	Ranking algorithm R=0.0
5	123456789	80 %	3.3 %	Relative fitness spinning wheel
6	123456789	60 %	10 %	Relative fitness spinning wheel

Tabella 8: Parametri dell'algoritmo nelle diverse prove [26].

Ogni esperimento è stato costituito da 20,000 stime della fitness derivanti da una popolazione di 200 individui per un totale di 100 generazioni.

Il software è stato installato su una workstation Sun Enterprise 6500 con un sistema operativo Solaris 8 con sedici processori da 400 Mhz e 16 GB di RAM. Ogni processore ha impiegato 24 minuti per elaborare ogni esperimento.

L'evoluzione della configurazione migliore della stessa generazione (*best-of-generations*) e della media generazionale per ogni casistica sono rappresentati in figura 24.

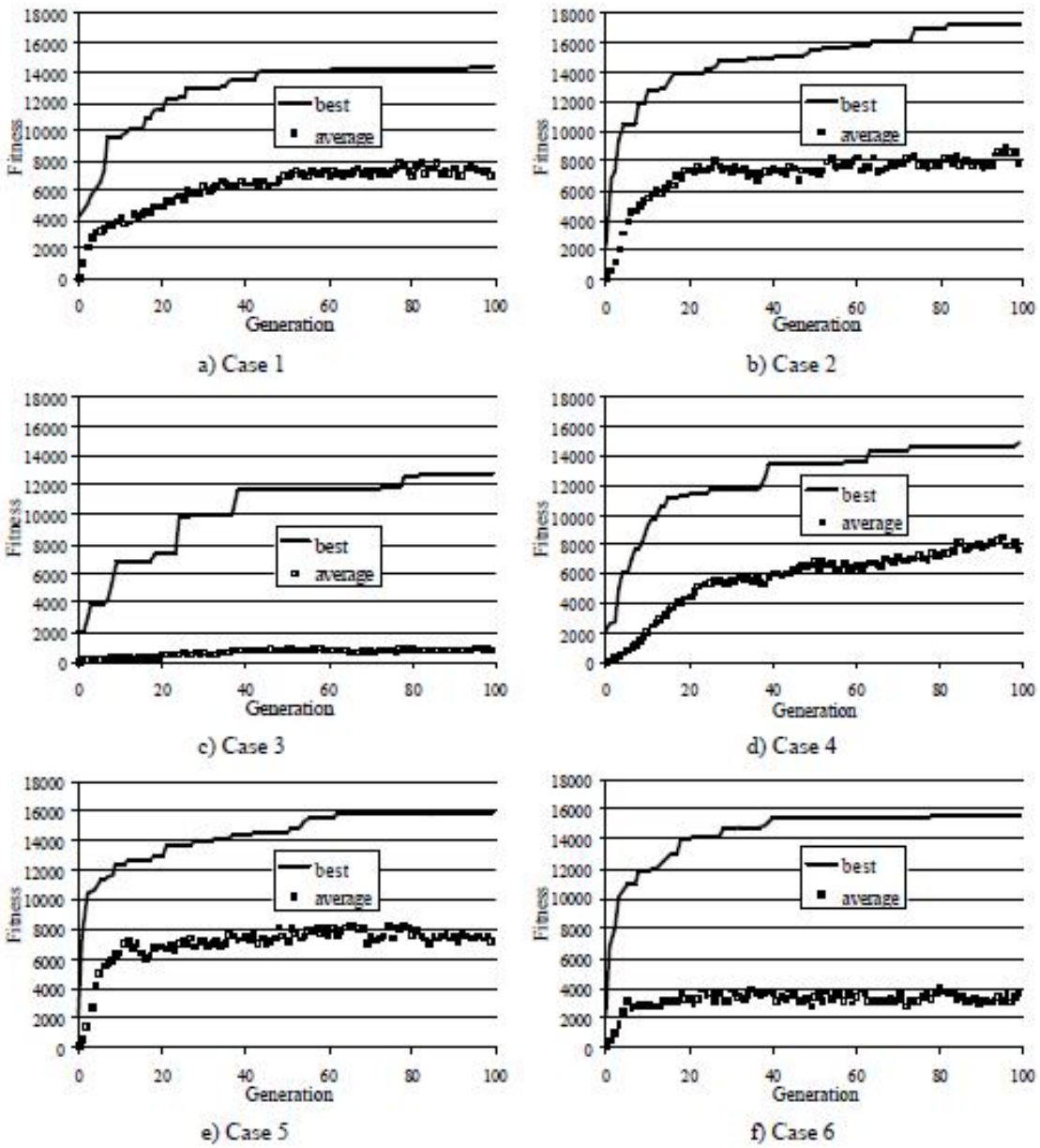

Figura 24: Evoluzione della configurazione migliore e della media generazionale [26].

Nella tabella 9 sono invece riportati i valori della *fitness* (dell'autonomia chilometrica dell'aeromobile), espressi in miglia nautiche, della migliore configurazione nell'arco della stessa prova (*best-of-run*) e della media generazionale.

Case	Best-of-run (generation produced)	Average after 100 generations
1	14,372 n.mi. (generation 94)	6,956 n.mi.
2	17,260 n.mi. (generation 82)	7,865 n.mi.
3	12,721 n.mi. (generation 82)	834 n.mi.

4	14,826 n.mi. (generation 99)	7,674 n.mi.
5	15,831 n.mi. (generation 85)	7,135 n.mi.
6	15,562 n.mi. (generation 78)	3,672 n.mi.

Tabella 9: Valori della fitness migliore e della media generazionale [26].

In figura 25 sono rappresentate le configurazioni migliori corrispondenti ai valori più alti di fitness all'interno della stessa prova (*best of run*).

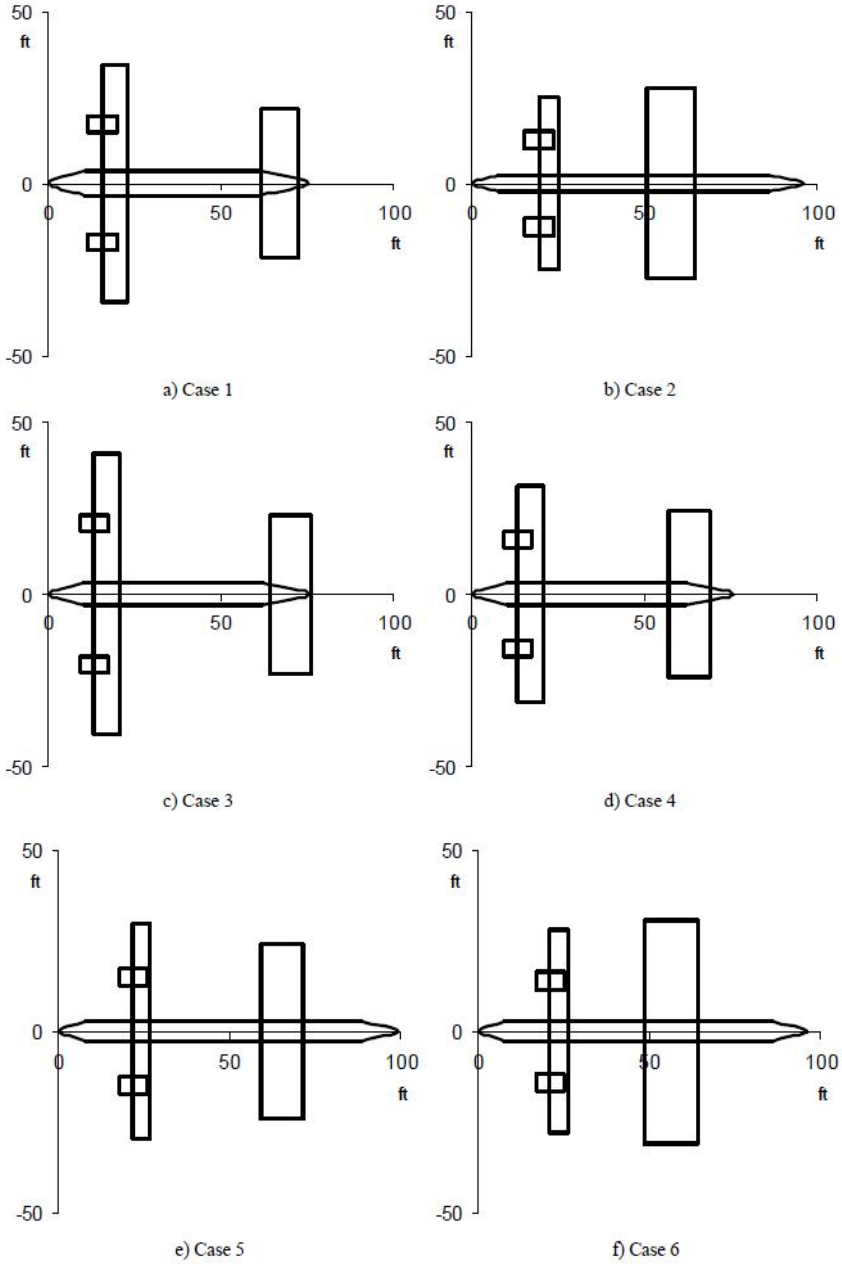

Figura 25: Configurazioni migliori di ogni prova [26].

I valori dei vincoli di progetto della configurazione migliore tra le cento considerate (*best-of-generation*) inclusi nell'elaborazione della funzione di fitness sono riportati nella tabella 10.

Constraints	Admissible values	Case 1	Case 2	Case 3	Case 4	Case 5	Case 6
Number of seat	60 < seats < 80	60	60	60	60	62	60
Mach number	M < 0.5	0.483	0.450	0.453	0.500	0.505	0.461
Tail after wing	xwoverl < xtoverl	0.232 < 0.855	0.216 < 0.565	0.200 < 0.887	0.200 < 0.790	0.232 < 0.629	0.232 < 0.548
Horizontal tail volume	0.4 < Vh < 1.0	0.405	0.402	0.409	0.469	0.409	0.417
Vertical tail volume	0.04 < Vv < 0.09	0.060	0.052	0.085	0.059	0.076	0.063
Momentum coef. about c.g.	-0.1 < cm_cg < 0.1	-0.099	-0.094	-0.099	-0.098	-0.095	-0.096
Zero fuel weight	Zfw < tow	42,422 < 99,685	41,554 < 100,000	44,983 < 100,000	42,185 < 100,000	43,007 < 100,000	43,780 < 100,00

Tabella 10: Valori dei vincoli delle configurazioni migliori [26].

A prima vista si può dire che i risultati riportati in tabella 9 sono in qualche modo delle sovra stime dell'autonomia dell'aeromobile. Questo perché il modello utilizzato per il calcolo della funzione di *fitness* prevede delle forti semplificazioni per quanto riguarda per esempio la stima della resistenza aerodinamica. Queste semplificazioni hanno reso i valori di resistenza, implementati nel modello, inferiori rispetto a quelli reali che avrebbe la configurazione scelta.

Basandosi, inoltre, sui sei esperimenti svolti si può vedere in che misura i diversi parametri scelti vadano ad influenzare i risultati forniti dall'algoritmo.

Per concludere l'analisi dei risultati verrà analizzata l'implementazione dei vincoli di progetto.

6.2.5 Effetto dei *random seed*

L'effetto dei *seed* per il generatore di numeri casuali può essere valutato mettendo a confronto il caso 1 e il caso 2.

Dalla figura 20 a) e b), si può notare come la scelta dei *seed* vada ad influenzare in maniera rilevante i risultati. La scelta iniziale dei semi va infatti a determinare la

popolazione iniziale e quindi il corredo genetico di informazioni disponibile per l'algoritmo. Tanto maggiore sarà il corredo iniziale quanto più efficiente sarà il risultato dell'algoritmo. Dalle figure si può vedere come le scelte del caso 2 abbiano portato ad una popolazione iniziale più sana che ha permesso di ottenere dei risultati migliori.

Si può inoltre vedere come nel caso 2 ci sia una crescita molto più veloce del caso 1 sia per quanto riguarda il valore della *fitness* medio, che quello ottimo. Questo risulta particolarmente evidente nella prima generazione simulata.

Guardando la tabella 9, comparando sempre i primi due casi, è possibile apprezzare l'importanza che ha la popolazione inizialmente generata sui risultati finali portando i valori medi della *fitness* (dopo 100 generazioni) a crescere del 10% e a migliorare il valore ottimo del 20%.

Molto interessante è la configurazione finale dell'aeromobile fornita dai due diversi casi. Guardando i risultati riportati nella figura 25 è possibile vedere come nel primo caso ci sia stata fornita dall'algoritmo una configurazione per così dire più classica rispetto a quella del caso 2 che in risposta al corredo iniziale di informazioni ha fornito una configurazione canard in cui il piano di coda si trova davanti all'ala principale in contrapposizione alla classica configurazione che riporta il piano di coda in posizione più arretrata rispetto all'ala.

L'algoritmo nel secondo caso ha infatti ottimizzato i dati arretrando leggermente l'ala principale e riducendone le dimensioni mentre il piano di coda è stato portato in una posizione più avanzata, al centro della fusoliera, e ne sono state aumentate le dimensioni.

6.2.6 Effetti della procedura di selezione

Per valutare gli effetti sull'efficienza dell'algoritmo delle varie procedure di selezione si prenderanno in considerazione i casi 2, 3 e 4.

Nel caso 2 la procedura di riproduzione selettiva utilizzata è la ruota della fortuna già descritta nei paragrafi precedenti che rappresenta senz'altro il processo di riproduzione comunemente usato in ogni algoritmo genetico.

La procedura *ranking* è stata usata come riproduzione selettiva per i casi 3 e 4. In questi casi la selezione della struttura che deve essere riprodotta avviene attraverso la classifica che la struttura presenta considerando tutti gli individui della popolazione della stessa generazione.

Nonostante l'algoritmo impedisca agli individui al di sopra della media di prendere il sopravvento sull'intera popolazione, questo metodo di selezione permette all'algoritmo di convergere più lentamente come è possibile notare in figura 24 b), c) e d).

I casi 3 e 4 differiscono per il parametro R (da non confondersi con l'autonomia chilometrica). Questo parametro viene utilizzato in GENESIS per classificare le strutture nella popolazione.

Il codice attribuisce il valore R al peggior individuo ed il valore di 2-R al migliore. Di conseguenza, nel caso 3 in cui R=0.75 la classifica varia da 0.75 a 1.25, mentre nel caso 4 avremo dei valori compresi tra 0 e 2 visto che si è scelto un valore di R=0. Così facendo, nel caso 3, gli individui sono classificati molto più uniformemente. Questo comporta che l'operatore seleziona per la riproduzione, all'interno della popolazione, degli individui che non sono tra i più adatti. È stata proprio questa la ragione che ha portato alla scarsa efficienza della prova corrispondente (come si vede in figura 20 c).), in cui la media generazionale praticamente non sembra migliorare al crescere delle generazioni riprodotte.

Per quanto riguarda invece i valori ottimi (*best of generation*) vediamo come migliorino all'aumentare delle generazioni dal momento in cui tutte le prove sono state fatte usando la strategia di selezione elitaria in cui vengono scelti per la riproduzione solo gli individui con le caratteristiche migliori. Di conseguenza, per effetto del *crossover* e delle mutazioni, il *best of generation* migliora di generazione in generazione. Si può così notare in figura 24 c) come il grafico del valore della fitness del *best of generation* che viene riportato assomigli ad una funzione a tratti (*step funtion*) in cui viene mantenuto un valore pressoché costante tra due miglioramenti successivi.

Confrontando la prove eseguita usando la procedura di selezione *ranking* con R=0.75 (caso 4) e la prova in cui si è scelta una procedura selettiva proporzionale

(caso 2), si può notare come il valore della *fitness* medio, nel primo caso, cresca più lentamente all'inizio, ma attraverso le generazioni continui ad aumentare il suo valore, mentre nel caso 2 si può vedere un incremento iniziale più notevole per poi mantenersi su valori molto più costanti al susseguirsi delle generazioni (figura 20 b)).

Come si può vedere dalla tabella 5 i valori medi dopo 100 generazioni, dei due diversi metodi di selezione, sono molto vicini, anche se la crescita che mostra il caso 4 fa pensare che se si aumentasse il numero di generazioni considerate si potrebbe arrivare a dei valori della fitness che potrebbero superare quelli del caso 2.

Per quanto riguarda la configurazione finale fornita dal *best of run* dei due casi (figura 21 b) e d)), si può vedere come la procedura di selezione *ranking* (caso 4) fornisca una soluzione per così dire più classica.

6.2.7 Effetto della probabilità di *crossover*

Gli effetti delle diverse probabilità di *crossover* possono essere valutati confrontando i casi 2 e 5. Questi due casi, infatti, differiscono solo per la diversa percentuale di probabilità di *crossover* introdotta nell'algoritmo. Nel caso 2 è stato utilizzato un valore più conservativo, il 60% di probabilità, mentre nel caso 5 si è scelto un valore più alto arrivando fino all'80%.

Se si va ad analizzare l'evoluzione del *best of generation* e del valor medio della *fitness* mostrata in figura 20 b) e 20 e) si nota subito come una probabilità maggiore di *crossover* comporti un incremento iniziale dei valori della *fitness*, per entrambi gli indicatori, molto più rapido anche se i valori finali risultano inferiori (circa il 10%) rispetto al caso più conservativo.

L'alta probabilità di *crossover* dell'80% sembra causare una perdita di individui con caratteristiche ottimali che vengono distrutti tra una generazione e l'altra, il che spiega i valori leggermente inferiori nei risultati finali rispetto al caso più conservativo.

Come si può vedere in figura 25 b) e 25 e) la configurazione finale a cui si giunge in entrambi i casi non sembra molto diversa, anche se nel caso in cui si è adottata una percentuale più conservativa si è giunti ad una tipica configurazione canard mentre

adottando una probabilità maggiore di *crossover* l'algoritmo restituisce una configurazione non ben definita e più simile ad una doppia ala.

6.2.8 Effetto della probabilità di mutazione

Prendendo in considerazione i casi 2 e 6 è possibile valutare gli effetti sull'algoritmo di diverse probabilità di mutazione, infatti mentre nel primo caso è stato utilizzato una valore conservativo del 3.3% nel secondo è stato incrementato fino al 10%. Che rappresentano entrambi delle percentuali alte rispetto a quanto visto in precedenza con valori decisamente più bassi.

Analizzando le figure 24 b) e 24 f) è possibile valutare come un'alta probabilità di mutazione abbia restituito dei risultati, in termini di *fitness* media, molto inferiori rispetto al caso per così dire più conservativo.

Nonostante la mutazione produca nuovi schemi che nelle generazioni precedenti non esistevano, ampliando così lo spazio di ricerca, si tende inevitabilmente a distruggere anche gran parte di individui potenzialmente idonei andando così a penalizzare fortemente i valori medi della *fitness* (come è visibile dalla figura 24 f) o dalla tabella 9).

I risultati mostrano comunque che sono solo i valor medi a subire un drastico calo mentre i valori del *best of run* rimangono accettabili o quanto meno confrontabili con il valore ottimo della fitness ottenuto nell'arco di tutte le prove.

Ciò conferma che la mutazione ha il vantaggio di creare nuovi schemi nello spazio di ricerca cosa che ha permesso di ottenere ugualmente una configurazione finale anche in questo caso canard come mostra la figura 25 f).

6.2.9 Implementazione dei vincoli

Come esposto in precedenza, i vincoli vengono classificati in severi e non-severi. I primi comportano, nel caso vengano violati, un valore di zero della funzione di *fitness* che viene così fortemente penalizzata e che, proprio per questo, devono essere completamenti soddisfatti nella soluzione fornita dal *best of run.*

Dalla tabella 10 è possibile vedere come questi vincoli, più precisamente i valori del posizione del piano di coda e del peso al decollo, siano stati soddisfatti in tutti e sei i casi considerati.

Per quanto riguarda invece i vincoli non-severi si ricorda che possono o non possono essere completamente soddisfatti a discapito di una penalizzazione del valore della *fitness* proporzionale alla deviazione dai valori ammessi. Ci si aspetta così che in alcuni casi ci possa essere una leggera deviazione dai valori imposti.

Nonostante siano ammesse delle piccole oscillazioni dei valori di questi vincoli, in tutti i casi vengono comunque rispettati.

L'unica eccezione è rappresentata dal caso 5 dove l'unico vincolo violato è il numero di Mach che si discosta dalle direttive di progetto di un valore vicino all'1%.

È interessante notare come l'algoritmo abbia massimizzato il valore della funzione di *fitness* rendendo attivi solo quei vincoli che risultano strettamente necessari.

Come è possibile vedere dalla tabella 10, il numero di posti, la posizione dell'ala, il volume orizzontale di coda e il coefficiente di momento attorno al baricentro sono stati impostati al loro minimo valore ammissibile, mentre il numero di Mach ed il peso al decollo sono stati impostati al valore massimo.

L'unico vincolo non attivo è il volume di coda verticale i cui valori variano rimanendo comunque all'interno dell'intervallo ammesso.

6.3 Discussione dei risultati

L'intero spazio di lavoro del problema espresso nella tabella 7 abbia la dimensione di 2^71 = 2.36e21. Uno spazio di ricerca così vasto rende virtualmente impossibile per un algoritmo di ricerca totalmente casuale trovare un valore ottimo. Ciò nonostante attraverso l'utilizzo di un algoritmo genetico è possibile risolvere il problema con solo 20,000 = 2e4 valutazioni del valore della *fitness*, rappresentando un impressionante guadagno di diciassette ordini di grandezza.

Poiché l'algoritmo genetico utilizza un approccio probabilistico, i risultati che fornisce non sono deterministici, fornendo, per l'appunto, risultati diversi ad ogni prova.

Le differenze che emergono scegliendo delle popolazioni iniziali diverse sono un esempio di questo comportamento probabilistico. Per poter raggiungere un valore di ottimizzazione globale è quindi necessario effettuare diverse prove con popolazioni iniziali differenti.

Dal paragone delle diverse metodologie di selezione, tra la ruota della fortuna e il metodo *ranking*, i risultati migliori sono statti ottenuti nel primo caso. Quando è stato utilizzato il metodo *ranking* con un valore di R=0 solo gli individui migliori sono stati riprodotti da una generazione all'altra. Come atteso, questo comporta una bassa efficienza dell'algoritmo dovuta alla riduzione della popolazione contenente le informazioni genetiche.

Sebbene sia desiderabile un alto tasso di *crossover* per riuscire a combinare diversi schemi, quando diventa eccessivo, per l'esattezza l'80% nelle prove fatte, l'efficienza dell'algoritmo genetico può diminuire a causa della distruzione di un alto numero di individui.

Vista la natura altamente non lineare del problema rappresentato dal progetto preliminare di un aeromobile, il tasso di mutazione può avere un impatto significante sulla scoperta di soluzioni innovative ampliandone gli spazi di ricerca. Ciò ha permesso con una probabilità di mutazione del 10% di imbattersi in una configurazione non del tutto convenzionale come quella canard.

Per quanto riguarda l'implementazione dei vincoli di progetto è chiaro che devono essere utilizzati in modo che il valore fornito della *fitness* sia penalizzato proporzionalmente alla deviazione dai valori ammessi, e non semplicemente imponendo un valore nullo, come è stato fatto nelle prove iniziali dell'esperimento. Infatti se il problema venisse affrontato annullando il valore della *fitness* ogni volta che vengono violati dei vincoli, il numero di individui validi nella popolazione iniziale sarebbe rappresentato da una percentuale molto bassa, circa il 5% della popolazione totale, eliminando così una quantità troppo alta di informazioni genetiche.

Considerando ancora i vincoli di progetto, l'efficienza dell'algoritmo genetico è stata provata mantenendo attivi tutti i vincoli inizialmente imposti.

Questo conferma come le tecniche di ottimizzazione che prevedono l'utilizzo di algoritmi genetici rappresentino, a tutti gli effetti, metodologie di ricerca molto efficienti.

L'esempio proposto in questa trattazione si riferisce al progetto preliminare di un piccolo jet regionale. Tuttavia, cambiando semplicemente lo spazio di ricerca, o meglio, variando sia il dominio delle variabili di progetto che i vincoli inclusi nella valutazione della funzione di *fitness*, è possibile progettare diverse tipologie di aeromobili.

Questa possibilità di implementare velocemente nuovi problemi può essere utilizzata non solo in ambito aeronautico ma in ogni campo in cui si presentino delle problematiche risolvibili attraverso l'utilizzo di algoritmi genetici a dimostrazione ancora una volta della robustezza di tali metodologie.

Conclusioni e sviluppi futuri

L'impiego di algoritmi genetici come metodologie di ottimizzazione e progettazione si presenta come un'utile risorsa, offrendo la possibilità di poter risolvere problemi di natura molto diversa, in modo semplice, fornendo risultati validi anche laddove le metodologie più classiche rischiano di fallire.

In ambito aeronautico, problemi e contesti molto diversi tra di loro sono strettamente connessi ed il numero di variabili che si inseriscono in un progetto, seppur preliminare, risultano di una complessità e di una grandezza dell'ordine delle migliaia, avere a disposizione questo tipo di tecniche rappresenta per un progettista uno strumento molto utile.

Si possono, infatti, utilizzare sia in una fase preliminare, come strumento all'interno di un'indagine non ancora dettagliata, sia come strumenti di verifica dei risultati forniti da altre tecniche, che risultano ormai convalidate sia dall'esperienza comune che dal tempo.

Possibili sviluppi su questo tipo di attività possono essere effettuati in due diversi campi applicativi: 1) aumento dell'efficienza dell'algoritmo genetico; 2) aumento del numero delle variabili di progetto ed implementazione di una funzione di valutazione della *fitness* più affidabile.

Per quanto riguarda l'efficienza dell'algoritmo genetico, risulta necessario un processo di ottimizzazione parametrica, per quanto riguarda sia le percentuali ottimali di *crossover* che di mutazione. Ulteriori studi potrebbero inoltre migliorare le procedure di selezione utilizzate includendo la selezione a torneo o modificando la ruota della fortuna.

Si può inoltre considerare l'implementazione parallela di un algoritmo genetico più ampio, andando ad utilizzare un numero maggiore di parametri di progetto, partendo dal problema relativamente piccolo proposto in questa trattazione.

Per affrontare problematiche di simulazione con un numero maggiore di variabili di progetto e, di conseguenza, valutazioni del valore della funzione di *fitness* più complesse, sia lo spazio di ricerca che il tempo che richiede la stima di ogni singola funzione di *fitness* aumentano considerevolmente.

Sarebbe, inoltre, molto interessante studiare a fondo il legame tra la dimensione della popolazione ed il numero di generazioni che si vogliono sviluppare. Si potrebbe, ad esempio, investigare se ad un aumento del numero di individui che compongono la popolazione considerata corrisponda una diminuzione delle generazioni necessarie per far convergere l'algoritmo a soluzione. L'obiettivo finale da raggiungere sarebbe quello di minimizzare il numero di valutazioni della *fitness* (dimensione della popolazione x numero di generazioni) necessarie per ottenere un risultato soddisfacente.

Il problema analizzato in questa trattazione riguarda la fase preliminare di un progetto aeronautico dove vengono considerate solo le caratteristiche di massima necessarie alla progettazione di un aeromobile. Sarebbe quindi necessario dopo questa prima stima scendere nel dettaglio del progetto andando ad aumentare notevolmente il numero delle variabili di progetto e aumentando di pari passo le valutazioni necessarie della funzione di *fitness*. Sarebbe necessario sviluppare un'interfaccia che si inserisca tra la soluzione dell'algoritmo genetico ed una combinazione tra l'analisi strutturale ed aerodinamica e che permetta di risolvere così il problema andando a sostituire il ruolo delle valutazioni della funzione di *fitness*.

Infine è giusto ricordare che, finora, nessun progetto dettagliato è stato ancora risolto utilizzando gli algoritmi genetici, ma rappresenterebbe senz'altro un interessante campo di ricerca in un ambito, che vede ancora dominanti altre metodologie di ottimizzazione come i metodi del gradiente con le loro inevitabili limitazioni.

Bibliografia

[1] Antona E., Chiesa S., Corpino S. e Viola N., *L'avamprogetto dei velivoli,* Memorie della Accademia delle Scienze di Torino, Classe di Scienze Fisiche Matematiche e Naturali, vol.32, 2009, pp.65-115.

[2] Antona E., *Fondamenti teorici dell'avamprogetto degli aeromobili,* Memorie della Accademia delle Scienze di Torino, Classe di Scienze Fisiche Matematiche e Naturali, vol.31, 2007, pp.147-170.

[3] Stanton R.N., *Weight estimation methods*, "SAWE Journal", vol.31, April-May 1972.

[4] Beltramo M.N., Morris M.A., Anderson J.L., *Application of parametric weight and cost estimating relationship to future transport aircraft,* 38th Annual Conference of the SAWE, 7-10 May 1979, New York.

[5] Gabrielli G., *Peso teorico e peso reale delle ali a sbalzo*, "L'Aerotecnica 1", 1953.

[6] Gabrielli G., *Su una espressione del peso ideale delle fusoliere a guscio,* Atti Accademia delle Scienze di Torino, Classe di Scienze Fisiche Matematiche e Naturali, vol.105, 1970-1971, pp.227-232.

[7] Gabrielli G., *Un metodo per la determinazione della superficie alare, e del suo allungamento nel progetto dei velivoli,* Atti Accademia delle Scienze di Torino, Classe di Scienze Fisiche Matematiche e Naturali, vol.86, 1951-1952, pp.146-155.

[8] Roskam J., *Airplane Design: preliminary sizing of airplanes*, vol.1-8, University of Kansas, USA, 1988, pp.1-5.

[9] Torenbeek E., *Synthesis of Subsonic Airplane Design*, Delft University Press, 1982, pp. 1-26.

[10] Goldberg D. E., *Genetic Algorithms in Search, Optimization, and Machine Learning*, Reading, MA, Addison-Wesley, 1989.

[11] Beightler C.S., Philipps D.T. & Wilde D.J., *Foundations of optimization*, Englewood Cliffs, NJ: Prentice-Hall, 1979.

[12] Simon H.A., *The sciences of the artificial*, Cambridge, MA:MIT Press, 1969.

[13] Darwin C., *The Origin of the Species*, Cambridge, Ma., Harvard University Press, 1967.

[14] Holland J.H., *Adaptation in natural and artificial systems*, Ann Arbor: The University of Michigan Press, 1975.

[15] Bethke A.D., Genetic Algorithms as Function Optimizers, Doctoral Dissertation, University of Michigan, 1980.

[16] Baker J.E., *Reducing bias and inefficiency in the selection alghorithm*, Proceedings of the second International conference on genetic algorithms a cura di J.J. Grefenstette, NJ, Lawrence –Erlbaum Associates, 1987, pp.14-21.

[17] Goldberg D.E., Deb K., *A comparative analysis of selection schemes used in genetic algorithms*, Foundations of genetic algorithms, a cura di Rawlins G.J.E., San Mateo, CA, Morgan Kauffman, 1991.

[18] Grefenstette J.J., *Optimization of control parameters for genetic algorithms*, IEEE Transctions on systems, man and cybernetics, vol.16,1986, pp.122-128.

[19] Syswerda G., *Uniform Crossover in Genetic Algorithms*, Proceedings of the Third Internation Conference on Genetic Algorithms, a cura di Schaffer J.D., San Mateo, CA, Morgan Kaufmann, 1989.

[20] Whitley D., *GENITOR: A Different Genetic Algorithm*, Proceeding of the Rocky Mountain Conference on Artificial Intelligence, Denver, CO, 1988.

[21] Perez R.E., Chung J., Behdinan K., *Aircraft conceptual design using genetic algorithms*, 8th Symposium on Multidisciplinary Analysis and Optimization, AIAA 2000-4938, 2000.

[22] Kroo I., Altus S., Braun R., Gage P., Sobieski I., *Multidisciplinary Optimization for Aicraft Preliminary Design*, AIAA 94-4325, 1994, pp.697-707.

[23] Takayasu K., Shinkyu J., Shiregu O., Yasushi I., Keita H., Hiroyuki M., *Multidisciplinary Design Optimization of Wing Shape for a Small Jet Aircraft Using Kriging Model*, 44th AIAA Aerospace Sciences Meeting and Exhibit, AIAA 2006-932, Reno, Nevada, 2006.

[24] Vicini A., Quagliarella D., *Inverse and Direct Airfoil Design Using a Multiobjective Genetic Algorithm*, AIAA Journal, Vol. 35, No. 9, 1997, pp. 1499-1505.

[25] Crossley W.A., Laananen D.H., *Conceptual design of helicopters via genetic algorithm*, Journal of Aircraft, Vol. 33, No. 6, 1996, pp. 1062-1070.

[26] Marta A.C., *Parametric Study of a Genetic Algorithm using a Aircraft Design Optimization Problem*, Department of Aeronautics and Astronautics of Stanford University, Stanford, California 94305, 2003.

[27] Alonso J., Martins J. and Reuther J., *High-Fidelity Aero-Structural Design of Complete Aircraft Configurations with Aeroelastic Constrains*, 16th AIAA Computational Fluid Dynamics Conference, AIAA Paper AIAA-2003-3429, Orlando, FL, June 23-26, 2003.

[28] Giulietti F., *Lezioni di Meccanica del Volo: Le prestazioni dei velivoli*, Bologna, Progetto Leonardo: Esculapio, 2006.

[29] Kroo I., *Aircraft Design: Synthesis and Analysis*, Sanford, CA, Desktop Aeronautics Inc., 2003.

[30] Grefenstette J.J., *A user's guide to GENESIS*, Vanderbilt University Computer Science Departement technical report CS-83-11, 1984.

[31] Grefenstette J.J., *GENESIS: a system for using genetic search procedures*, Proceedings of the Conference on Intelligent Systems and Machines, Rochester, MI, 1984, pp. 161-165.