DOTTORATO IN MATEMATICA COMPUTAZIONALE E RICERCA OPERATIVA

VII CICLO

—

ALGORITMI EVOLUTIVI PER L'OTTIMIZZAZIONE

—

ANDREA TETTAMANZI

Supervisori:

Prof. Gianni Degli Antoni

Prof. Vincenzo Capasso

INDICE

INTRODUZIONE 1

CAP. I. INTRODUZIONE AGLI ALGORITMI EVOLUTIVI 3
1. Generalità 3
2. Prospettiva storica 6
3. Notazione 9
4. Le Strategie Evolutive 12
5. La Programmazione Evolutiva 17
6. Gli Algoritmi Genetici 19
7. La Programmazione Genetica 25
8. Sintesi 27
9. Modelli di selezione 33
10. Applicazioni ai problemi di ottimizzazione 38

CAP. II. ANALISI STOCASTICA 45
1. Misure su popolazioni e individui 45
2. Un Algoritmo Evolutivo Astratto 48
3. L'Algoritmo Evolutivo Astratto come Processo Stocastico 50
4. Statistiche di Algoritmi Evolutivi 53
5. Convergenza ad almeno un genotipo ottimale 57
6. Il Teorema degli Schemi rivisitato 61
7. Algoritmi Evolutivi come Martingale 64
8. Velocità di convergenza 68

CAP. III. UN MODELLO DISTRIBUITO DI SELEZIONE 76
1. Motivazioni 76
2. Un modello basato sulla competizione 77
3. Competizione e idoneità 80
4. Analisi della selezione competitiva 87
5. Conclusioni 95

CAP. IV. ALGORIMI EVOLUTIVI E LOGICA SFUMATA 96
1. Motivazioni 96
2. Un caso di studio: l'inferenza simbolica di formule 97
3. Descrizione dell'algoritmo 102
4. Elementi di Teoria degli Insiemi Sfumati 107
5. Il Governo Fuzzy 109
6. Conclusioni 116

BIBLIOGRAFIA 117

INTRODUZIONE

Gli Algoritmi Evolutivi sono metodi stocastici di ottimizzazione ispirati ai fenomeni biologici dell'evoluzione naturale.

Le prime proposte in questa direzione risalgono alla metà degli anni sessanta, quando, negli Stati Uniti, John Holland introdusse gli Algoritmi Genetici [65] e Lawrence Fogel la Programmazione Evolutiva [47] mentre in Europa, contemporaneamente e indipendentemente, Ingo Rechenberg e Hans-Paul Schwefel iniziavano il loro lavoro sulle Strategie Evolutive [124, 131]. I loro sforzi pionieristici diedero luogo col tempo ad una classe di metodi adatti a trattare problemi complessi dove poco si conosca dello spazio di ricerca soggiacente.

Un algoritmo evolutivo simula l'evoluzione di una popolazione di soluzioni candidate per il problema oggetto applicando iterativamente un insieme di operatori stocastici, di cui i più importanti sono la *mutazione,* la *ricombinazione,* la *riproduzione* e la *selezione.* La mutazione perturba casualmente una soluzione candidata; la ricombinazione scompone due soluzioni distinte e ne produce una nuova riassemblando casualmente le loro parti; la riproduzione replica con probabilità maggiore le migliori soluzioni presenti nella popolazione e la selezione elimina quelle scadenti.

Il processo che ne risulta tende a trovare soluzioni globalmente ottime per il problema oggetto in modo del tutto simile a come in Natura popolazioni di organismi si adattano all'ambiente che le circonda.

Dagli inizi di questo decennio gli Algoritmi Evolutivi hanno cominciato a trovare un numero sempre crescente di applicazioni industriali e gestionali, a testimonoanza di una raggiunta maturità scientifica del campo.

Tuttavia, nella semplicità con la quale è definito e implementato, un algoritmo evolutivo esibisce comportamenti di una complessità tale da costituire una sfida per l'investigazione teorica. Per esempio, a tutt'oggi, pur essendo disponibili risultati di convergenza in probabilità, non si è in grado di affermare alcunché di generale sulla velocità di convergenza degli Algoritmi Evolutivi e di come questa si rapporti con la complessità computazionale di un problema. Inoltre, molto del successo applicativo degli Algoritmi Evolutivi poggia su basi empiriche e su una dose di ciò che, scherzosamente ma non troppo, alcuni chiamano "magia nera".

I contributi originali di questa Tesi sono tre:

- un risultato di convergenza in probabilità ottenuto sfruttando il concetto di martingala: sebbene il risultato in sé non sia nuovo, l'approccio seguito è in grado di ricavarlo in maniera significativamente più semplice ed elegante;
- un modello di selezione, basato sul concetto di competizione, che per una certa classe di problemi consente un risparmio consistente di risorse computazionali e,

allo stesso tempo, riduce il numero di scelte arbitrarie nella definizione dei criteri di valutazione delle soluzioni;
- una tecnica, basata sulla Logica Sfumata, per il controllo dinamico di alcuni parametri del processo evolutivo e la rilevazione automatica dell'emergenza di una soluzione ottimale nel corso dell'evoluzione, che riduce l'esigenza di intervento umano per l'utilizzo pratico di un algoritmo evolutivo.

Il primo capitolo di questa Tesi è un'introduzione agli Algoritmi Evolutivi, con cenni storici e un ampio resoconto delle problematiche, dei principali risultati e dello stato dell'arte in questo campo di ricerca. Il capitolo ha lo scopo di predisporre il retroterra culturale necessario per affrontare e situare in un contesto scientifico il materiale presentato nel resto della Tesi.

Il secondo capitolo sviluppa un'analisi stocastica degli Algoritmi Evolutivi facendo ricorso agli strumenti matematici forniti dalla Teoria dei Processi Stocastici, delle Catene di Markov e delle Martingale, per stabilire le condizioni necessarie e sufficienti per la loro convergenza alle soluzioni ottimali di un problema e, ove possibile, determinare la velocità di questa convergenza.

In tempi recenti, molta attenzione è stata dedicata al calcolo parallelo ed in particolare al parallelismo massiccio, ovvero al caso in cui il numero di processori è elevato o addirittura elevatissimo. L'ostacolo principale che si frappone allo sfruttamento proficuo di queste potenti risorse di calcolo è costituito dalla cosiddetta scalabilità. In quest'ottica, il terzo capitolo introduce ed analizza un modello di selezione rivolto all'implementazione efficiente degli Algoritmi Evolutivi su elaboratori ad elevato parallelismo.

Gli Algoritmi Evolutivi sono relativamente facili da implementare e, in generale, le loro prestazioni su problemi difficili si rivelano abbastanza soddisfacenti, soprattutto se confrontate con la piccola quantità di conoscenza specifica sul problema che deve essere da essi incorporata per funzionare. Tuttavia, un algoritmo evolutivo richiede la supervisione di un operatore per essere sfruttato come strumento pratico. Il quarto capitolo affronta una serie di questioni che hanno a che fare da un lato con la rilevazione automatica dell'emergenza di una soluzione ottimale all'interno di una popolazione e dall'altro con il controllo dinamico del processo evolutivo, con lo scopo di ridurre o eliminare l'esigenza di intervento umano. Lo strumento matematico adottato a questo scopo è la Logica Sfumata.

Capitolo I

INTRODUZIONE AGLI ALGORITMI EVOLUTIVI

§ 1. — **Generalità.**

Sommario del capitolo — la metafora dell'evoluzione naturale — due prospettive — nota sulla terminologia — struttura di un Algoritmo Evolutivo.

1.1. — Gli *Algoritmi Evolutivi* sono una vasta classe di algoritmi stocastici di ricerca per problemi di ottimizzazione che si basano sull'imitazione di quei processi biologici che permettono a popolazioni di organismi di adattarsi all'ambiente in cui vivono: l'ereditarietà genetica e la sopravvivenza del più adatto, concetti introdotti nel secolo scorso da Charles Darwin [34] e ancora oggi ampiamente riconosciuti come validi, anche se corredati da altri dettagli complementari (si veda, per una trattazione divulgativa di buon livello, [37]).

Questo capitolo introduce gli Algoritmi Evolutivi e ne fornisce una breve prospettiva storica (§2); viene fissata una notazione uniforme per il resto della trattazione (§3); segue una descrizione delle principali correnti all'interno di questo campo di ricerca (§4–8): una rassegna sintetica su cui questa presentazione si basa, espandendola, è [7]. Il paragrafo successivo (§9) entra nei dettagli che distinguono diverse versioni di Algoritmi Evolutivi, come la codifica del problema e la definizione dei vari operatori genetici. Il capitolo si conclude (§10) descrivendo alcune delle applicazioni in cui gli Algoritmi Evolutivi sono stati impiegati con successo.

1.2. — La metafora Essenzialmente gli Algoritmi Evolutivi fanno uso di una metafora in cui il problema oggetto prende il posto dell'ambiente; le soluzioni ammissibili sono viste come individui che vivono in questo ambiente; il grado di adattamento di un individuo all'ambiente è infine il corrispettivo della funzione obiettivo calcolata su una soluzione ammissibile. Nella metafora degli algoritmi evolutivi insiemi di soluzioni ammissibili prendono il posto delle popolazioni di organismi.

Negli Algoritmi Evolutivi la selezione opera su strutture di dati immagazzinate nella memoria dell'elaboratore elettronico e, con il tempo, la funzionalità di queste strutture evolve in maniera sostanzialmente analoga a come evolvono in natura popolazioni di organismi viventi.

1.3. — Sebbene il modello computazionale introduca delle semplificazioni drastiche

rispetto al mondo naturale, gli algoritmi evolutivi si sono rivelati capaci di far emergere strutture sorprendentemente complesse e interessanti. Ogni struttura, o *individuo*, può essere la rappresentazione, secondo un'opportuna codifica, di una particolare soluzione di un problema, di una strategia per affrontare un gioco, di un'immagine o addirittura di un semplice programma per calcolatore.

Due prospettive.

1.4. — Introdotti a partire dagli anni sessanta da diversi studiosi e con diversi nomi, gli Algoritmi Evolutivi cominciarono ad attirare un'attenzione sempre crescente dalla fine degli anni ottanta e si trovano attualmente al centro di promettenti sforzi di ricerca in campi (parzialmente sovrapposti) quali:

- ottimizzazione funzionale,
- apprendimento automatico,
- elaborazione delle immagini,
- robotica,
- programmazione non lineare,
- teoria dei grafi.

1.5. — I tratti caratteristici che distinguono gli Algoritmi Evolutivi da tutti gli altri metodi di ottimizzazione si possono riassumere nei seguenti quattro aspetti principali [54]:

1. gli Algoritmi Evolutivi operano indirettamente su una opportuna codifica dell'insieme dei parametri, piuttosto che direttamente sui parametri stessi;
2. la ricerca dell'ottimo viene effettuata in parallelo considerando contemporaneamente una popolazione di punti, piuttosto che un singolo punto dell'insieme ammissibile per volta;
3. non viene richiesta alcuna condizione di regolarità sulla funzione obiettivo, come continuità, derivabilità, convessità, ecc.: gli Algoritmi Evolutivi sono infatti algoritmi di ricerca "cieca";
4. gli Algoritmi Evolutivi sono definiti attraverso regole di transizione probabilistiche, piuttosto che deterministiche.

Si noti che gli aspetti 3 e 4 sono condivisi da altri metodi stocastici di ottimizzazione, come i Metodi Monte Carlo e l'Annealing Simulato.

1.6. — Come si è già rilevato, gli Algoritmi Evolutivi sono studiati e utilizzati soprattutto come strumenti per l'ottimizzazione e la risoluzione di problemi; tuttavia essi possono essere sfruttati per meglio comprendere l'evoluzione in sé (cfr. §8) e per modellare alcuni sistemi dinamici.

Molti sistemi che evolvono nel tempo infatti possono essere modellati per mezzo di un Algoritmo Evolutivo: tra gli altri ricadono in questa classe sistemi biologici, come ecosistemi, sistemi immunologici e sistemi genetici, e sociali, come economie e sistemi politici.

Questa seconda visione degli Algoritmi Evolutivi come modello astratto dei sistemi

evolutivi suggerisce una prospettiva computazionale dell'evoluzione, in cui i meccanismi della selezione naturale, dell'ereditarietà e della mutazione casuale svolgono essenzialmente la funzione di trasmettere ed elaborare informazione.

Esiste perciò la possibilità concreta che lo studio di modelli altamente idealizzati come gli Algoritmi Evolutivi possa aprire l'accesso a livelli più profondi di comprensione dei sistemi evolutivi naturali.

Nota sulla terminologia.

1.7. — Sebbene il termine di gran lunga più noto ed utilizzato per indicare quella vasta classe di metodi che in qualche modo si rifanno alle idee dell'evoluzione naturale sia quello di *Algoritmo Genetico*, dal punto di vista tassonomico esso è ambiguo e potrebbe essere fonte di equivoci.

In pratica, la letteratura più recente del settore preferisce riservare il nome di *Algoritmo Genetico* a quella classe più ristretta di Algoritmi Evolutivi, introdotti da John Henry Holland dell'Università del Michigan di Ann Arbor, verso la fine degli anni sessanta.

Esistono infatti diverse classi di algoritmi basati sull'imitazione dell'evoluzione naturale, tutte strettamente imparentate con gli Algoritmi Genetici intesi nel senso stretto del termine: esse comprendono le *Strategie Evolutive* (*Evolutionsstrategien*), sviluppate da Ingo Rechenberg e Hans-Paul Schwefel al Politecnico di Berlino a partire dal 1964 parallelamente al lavoro di Holland e indipendentemente da quello; la *Programmazione Evolutiva* (*Evolutionary Programming*), introdotta da Lawrence J. Fogel, A. J. Owens e M. J. Walsh nella seconda metà degli anni sessanta e recentemente raffinata da David B. Fogel, la più o meno contemporanea *Scatter Search* di F. Glover e, nuova nata della famiglia, la *Programmazione Genetica* (*Genetic Programming*) proposta da John Koza dell'Università Stanford di Palo Alto all'inizio di questo decennio.

1.8. — Purtroppo a tutt'oggi non c'è accordo completo sul nome da usarsi per designare la classe più ampia di algoritmi, comprendente tutte quelle elencate sopra ed eventualmente altre, che costituisce l'oggetto di questa Tesi: Zbigniew Michalewicz ha di recente proposto il termine *Evolution Programs* [103], che, se da un lato gli ha permesso di intitolare il suo libro con un suggestivo richiamo a quel classico dell'Informatica che è [148], dall'altro pare non avere soddisfatto la comunità degli studiosi di questo campo, che attualmente sembrano preferire il termine *Algoritmo Evolutivo* (*Evolutionary Algorithm*) [7].

Struttura di un Algoritmo Evolutivo.

1.9. — Un Algoritmo Evolutivo mantiene una popolazione di possibili soluzioni del problema oggetto e su di essa simula un processo evolutivo, applicando ripetutamente un certo numero di operatori stocastici noti con i nomi di *mutazione*, *ricombinazione*, *riproduzione* e *selezione*.

La mutazione perturba in modo casuale una soluzione candidata, la ricombinazione decompone due o più soluzioni distinte, mischiandone in modo casuale le parti per ricomporre una nuova soluzione, la riproduzione replica le soluzioni migliori presenti nella popolazione, tipicamente in maniera proporzionale alla bontà relativa, mentre la selezione elimina dalla popolazione le soluzioni peggiori.

La popolazione di partenza può essere costituita da un campione casuale dello spazio delle soluzioni oppure da un certo numero di soluzioni trovate con metodi di ricerca locali, se questi sono disponibili, così come da un misto delle due opzioni.

Il processo risultante tende a trovare soluzioni globalmente ottime più o meno nella stessa maniera in cui in Natura una popolazione di organismi tende ad adattarsi sempre meglio all'ambiente che la circonda.

§ 2. — **Prospettiva storica.**

Primi tentativi — Holland e gli Algoritmi Genetici — Rechenberg e le Strategie Evolutive — Fogel e la Programmazione Evolutiva — il problema della rappresentazione — Koza e la Programmazione Genetica.

2.1. — L'idea di usare selezione e mutazione casuale per un compito di ottimizzazione risale almeno agli anni cinquanta, con il lavoro di G. E. P. Box [18], che tuttavia non fece uso dell'elaboratore elettronico. Contemporaneamente diversi studiosi, tra cui A. Fraser [49] e H. J. Bremermann [21] concepirono l'idea di simulare l'evoluzione su un elaboratore elettronico. Alla fine degli anni cinquanta e inizio del decennio successivo ci furono diversi sforzi tesi a incorporare idee tratte dall'evoluzione naturale in sistemi computazionali. Di questi, i più noti sono quelli di Holland, Fogel e Rechenberg.

Rechenberg insistette sull'importanza di selezione e mutazione come meccanismi atti a risolvere difficili problemi di ottimizzazione con parametri continui; Fogel sviluppò idee simili per far evolvere agenti intelligenti sotto forma di automi a stati finiti; Holland si concentrò sulle proprietà adattative di popolazioni nel loro complesso e sull'importanza di meccanismi di ricombinazione come l'incrocio (*cross-over*).

Holland e gli Algoritmi Genetici.

2.2. — Gli interessi di ricerca di John Henry Holland negli anni sessanta erano dedicati allo studio dei processi generali di adattamento e si concentravano sull'idea di un sistema che ricevesse dati sensoriali dall'ambiente per mezzo di rilevatori binari [65]. In questo modello, le strutture appartenenti allo spazio di ricerca venivano progressivamente modificate da operatori selezionati da un *piano adattativo*, che giudicava la qualità dei tentativi precedenti servendosi di misure di valutazione. Holland indicò come interpretare i cosiddetti *piani riproduttivi* in termini di Genetica, Economia, Teoria dei Giochi, riconoscimento di strutture e ottimizzazione di parametri.

I *piani genetici*, o, come furono poi ribattezzati dagli studenti di Holland, Algoritmi Genetici, furono applicati all'ottimizzazione di parametri per la prima volta nel 1975 da Kenneth A. De Jong [78], che in pratica gettò le fondamenta di questa tecnica applicativa. Il numero di applicazioni alla risoluzione di problemi pratici basate sugli Algoritmi Genetici crebbe durante gli anni ottanta, tanto da spingere un altro studente di Holland, David Edward Goldberg, a pubblicare una monografia introduttiva [54] e Lawrence Davis a compilare, addirittura, un manuale pratico-applicativo [36] di sapore prettamente ingegneristico.

Rechenberg e le Strategie Evolutive.

2.3. — I primi sforzi verso la formulazione del concetto di Strategia Evolutiva [124] ebbero luogo nel 1964 presso il Politecnico di Berlino (TUB), in Germania, da parte di Ingo Rechenberg e Hans-Paul Schwefel. Le applicazioni erano principalmente sperimentali e avevano a che fare con problemi di idrodinamica come l'ottimizzazione di tubi piegati e di ugelli per motori a getto. Diverse versioni della strategia furono simulate sul primo elaboratore disponibile al TUB [131].

Ingo Rechenberg sviluppò una teoria della velocità di convergenza per la cosiddetta strategia $(1 + 1)$, un semplice meccanismo mutazione-selezione su una popolazione di un solo individuo che crea un discendente ad ogni generazione per mezzo di una mutazione gaussiana, e propose una regola, confermata dagli esperimenti, per modificare la deviazione standard delle mutazioni in maniera esogena; inoltre egli propose una prima strategia evolutiva multimembro, $(n + 1)$, in cui $n > 1$ individui si ricombinano per formare un discendente, che, dopo aver subito una mutazione, va a sostituire il peggiore dei genitori.

Questa strategia, se non fu mai impiegata a fondo, tuttavia suggerì le idee fondamentali che portarono alle strategie (n, m) e $(n+m)$ introdotte in seguito da Hans-Paul Schwefel, le quali, in particolare la seconda, costituiscono lo stato dell'arte nella ricerca sulle Strategie Evolutive.

Fogel e la Programmazione Evolutiva.

2.4. — All'inizio degli anni sessanta Lawrence J. Fogel, studente all'Università di California a Los Angeles, ebbe l'idea di sfruttare un'evoluzione simulata per fare emergere dell'intelligenza artificiale in una popolazione di algoritmi in competizione tra di loro [46]. Quest'idea fu esplorata in una serie di studi [47] e, qualche anno più tardi, Fogel e colleghi descrissero una tecnica per l'evoluzione di automi a stati finiti per compiti di predizione, che chiamarono Programmazione Evolutiva [48]. Le regole stato-transizione di questi automi venivano modificate tramite mutazioni casuali con probabilità uniforme sul corrispondente alfabeto finito e discreto. Gli automi erano valutati in base al numero di simboli che erano in grado di predire correttamente; ogni automa di una popolazione generava per mutazione esattamente un discendente per la generazione successiva, quindi la metà migliore dei genitori e dei figli sopravviveva

andando a costituire una nuova popolazione.

Il problema della rappresentazione.

2.5. — Quello della rappresentazione è un problema centrale negli Algoritmi Evolutivi, poiché ciò che essi manipolano è una versione codificata delle soluzioni di un problema e lo schema di codifica può limitare in modo significativo l'efficacia dell'algoritmo.

Gli Algoritmi Genetici di Holland, lavorando esclusivamente su sequenze di cifre binarie di lunghezza fissa, permisero di sviluppare un certo corpo di risultati teorici, ma ben presto ci si accorse che un tale schema mal si adattava a trattare certi tipi di problemi, come per esempio i problemi di ordinamento (classe in cui ricadono i problemi di scheduling, time-tabling, pianificazione e il noto problema del commesso viaggiatore). Per questo motivo negli anni ottanta si cominciarono a sviluppare estensioni degli Algoritmi Genetici in cui si optava per schemi di rappresentazione *ad hoc*, dettati cioè dalla natura dello specifico problema che si voleva trattare.

Koza e la Programmazione Genetica.

2.6. — La Programmazione Genetica porta alle estreme conseguenze il problema della rappresentazione e propone un nuovo modello generale, in cui gli individui sono addirittura semplici programmi per calcolatore. Il lavoro originale di John Koza [84] utilizza a questo scopo programmi in LISP, un linguaggio di programmazione che, per alcune sue caratteristiche, ben si presta a questo tipo di applicazione, anche se la scelta di un particolare linguaggio in realtà non è determinante.

La "Vita Artificiale".

2.7. — "Vita Artificiale" (*Artificial Life*) è il nome dato a una nuova disciplina scientifica che studia la vita "naturale" tentando di ricreare dal nulla fenomeni biologici utilizzando elaboratori elettronici ed altri strumenti "artificiali". Nei suoi intenti, quindi, la Vita Artificiale, con il suo approccio sintetico che, piuttosto che studiare fenomeni biologici osservando il funzionamento di organismi viventi, cerca di produrre sistemi che si comportano come quelli, è complementare all'approccio analitico della Biologia tradizionale.

Un testo divulgativo che tenta di tracciare una storia di questa disciplina, dalle idee pioneristiche di Von Neumann fino agli sviluppi più recenti, è [90].

§ 3. — Notazione.

Il problema oggetto — lo spazio dei genotipi — la funzione di espressione genetica — gli individui — popolazioni — la funzione di valutazione — la funzione di idoneità.

In questo paragrafo si introducono le notazioni generali e le definizioni fondamentali di cui sarà fatto uso nel seguito.

3.1. — Il problema oggetto Nella formulazione più generale [116], un problema di ottimizzazione consiste nel minimizzare una funzione di costo $c: \mathcal{S} \to \mathbb{R}$,

$$\min\{c(\mathbf{s})\},$$

dove $\mathbf{s}$ indica una generica soluzione, soggetta al vincolo

$$\mathbf{s} \in \mathcal{S}.$$

L'insieme $\mathcal{S}$ è chiamato *spazio ammissibile* e una soluzione $\mathbf{s} \in \mathcal{S}$ viene detta *ammissibile*.

Nel seguito ci si riferirà a questo problema di ottimizzazione come al *problema oggetto*, cioè il problema che si intende risolvere per mezzo di un Algoritmo Evolutivo.

3.2. — Lo spazio dei Genotipi Si indichi con Γ lo spazio dei genotipi. Un genotipo è una struttura di dati arbitraria che codifica in una maniera opportuna una particolare soluzione del problema oggetto. Al massimo livello di astrazione si può affermare che un genotipo deve essere composto da un certo numero l di costituenti fondamentali (i geni) organizzati secondo una qualche struttura. Ogni gene può assumere un certo numero (finito) di valori (alleli).

3.3. — La funzione di Espressione Genetica Associato allo spazio dei genotipi vi è lo spazio $\Phi \subseteq \mathcal{S}$ delle soluzioni rappresentabili per mezzo di strutture appartenenti allo spazio Γ, o, per continuare con la metafora biologica, dei fenotipi. Un fenotipo è l'espressione di un genotipo secondo una funzione di decodifica $M: \Gamma \to \Phi$. In generale nulla può essere detto a riguardo della funzione di espressione genetica M, tranne che essa mappa ogni genotipo in uno e un solo fenotipo, mentre potrebbe benissimo darsi che due genotipi siano mappati nello stesso fenotipo, o che esistano fenotipi ai quali non corrisponde alcun genotipo.

3.4. — Gli Individui Si parlerà, nel seguito, di *individui*: un individuo, così chiamato in quanto membro di una popolazione, può essere considerato sotto l'aspetto genotipico, e allora è identificato dal suo genotipo, oppure sotto l'aspetto fenotipico, e allora è identificato dalla soluzione che esso rappresenta; tuttavia due individui distinti di una popolazione possono condividere lo stesso genotipo o lo stesso fenotipo; si intende cioè che ciò che li distingue è una caratteristica non rilevante rispetto al livello di astrazione in cui si opera; meglio, nulla vieta che in una popolazione siano presenti più

copie dello stesso genotipo o fenotipo: tali copie, pur essendo tra loro identiche, sono entità distinte.

Popolazioni.

3.5. — Lo spazio delle Popolazioni Si indichi con $\Gamma^{(n)}$ lo spazio delle popolazioni formate da n individui considerati come genotipi. Un elemento di $\Gamma^{(n)}$ può essere considerato come una classe di equivalenza di n-uple di Γ^n identiche a meno di permutazioni.

Si converrà di denotare la classe di equivalenza delle n-uple identiche a $(\gamma_1, \gamma_2, \ldots, \gamma_n)$ a meno di permutazioni con la comoda scrittura $\{|\gamma_1, \gamma_2, \ldots, \gamma_n|\}$. Questa notazione vuole suggerire che, come nel caso degli insiemi, l'ordine degli elementi è irrilevante, mentre, a differenza dagli insiemi, è importante il numero di occorrenze di ciascun elemento (per inciso, questo tipo di oggetto algebrico è noto anche con il nome di *multi-insieme*).

Si indichi poi con Γ^* lo spazio delle popolazioni formate da un numero qualsiasi di individui:

$$\Gamma^* = \bigcup_{n=1}^{\infty} \Gamma^{(n)}.$$

3.6. — Le funzioni di Occupazione Ogni popolazione $x \in \Gamma^{(n)}$ è in corrispondenza biunivoca con una funzione "di occupazione" $n_x : \Gamma \to \mathbb{N}$, tale che, per ogni $\gamma \in \Gamma$, $n_x(\gamma)$ è il numero di occupazione di γ in x, cioè il numero di individui con genotipo γ presenti nella popolazione x; ovviamente si ha che, per ogni $x \in \Gamma^{(n)}$,

$$\sum_{\gamma \in \Gamma} n_x(\gamma) = n;$$

chiaramente, se $x \in \Gamma^*$, questa ipotesi si rilassa; si può allora definire la dimensione della popolazione x come

$$\|x\| \equiv \sum_{\gamma \in \Gamma} n_x(\gamma).$$

Si osservi come un sottoinsieme $G \subseteq \Gamma$ non sia altro che un caso particolare di popolazione per cui vale

$$n_G(\gamma) = \begin{cases} 1, & \text{se } \gamma \in G; \\ 0, & \text{altrimenti.} \end{cases}$$

In alcuni casi sarà più conveniente considerare la quota di occupazione di un genotipo γ in x, cioè la porzione degli individui della popolazione che portano il genotipo γ, $q_x(\gamma) = n_x(\gamma)/\|x\|$; si ha allora che, per ogni $x \in \Gamma^*$,

$$\sum_{\gamma \in \Gamma} q_x(\gamma) = 1.$$

Posto $r = \|\Gamma\|$, si dimostra (cfr. [42], §II.5) il seguente

3.7. — Teorema Il numero di popolazioni distinte che si possono formare con n individui estratti dall'insieme Γ è:

$$\|\Gamma^{(n)}\| = \binom{r+n-1}{n} = \binom{r+n-1}{r-1}.$$

3.8. — Dimostrazione È possibile rappresentare una popolazione di n individui mediante $r+1$ sbarrette verticali che delimitano r spazi, corrispondenti ciascuno a un elemento di Γ, e inserendo in ciascuno di questi spazî un numero di palline uguale al numero di occorrenze (nella popolazione) dell'individuo corrispondente. Per esempio, il simbolo $|\bullet\bullet\bullet|\bullet||||\bullet\bullet\bullet\bullet|$ identifica la popolazione di $n = 8$ individui estratti da un insieme di $r = 6$ elementi, nella quale compaiono 3 copie dell'individuo 1, un individuo 2 e 4 copie dell'individuo 6.

Un simbolo siffatto comincia e finisce sempre con una sbarretta verticale, ma le $r-1$ sbarrette che restano e le n palline possono stare in qualunque ordine. In questo modo risulta chiaro che il numero di distribuzioni distinte è uguale al numero di modi di scegliere n posizioni tra $n+r-1$, da cui discende la tesi. ∎

Definiamo l'inclusione e, di conseguenza, l'unione e l'intersezione per popolazioni di individui nel modo seguente:

3.9. — Definizione Date due popolazioni $x, y \in \Gamma^*$, x è una *sottopopolazione* di y, in simboli $x \sqsubseteq y$, se e solo se, per ogni $\gamma \in \Gamma$, $n_x(\gamma) \leq n_y(\gamma)$.

3.10. — Definizione Date due popolazioni $x, y \in \Gamma^*$, l'*unione* di x e y è la popolazione $u = x \sqcup y \in \Gamma^*$ tale che, per ogni $\gamma \in \Gamma$, $n_u(\gamma) = n_x(\gamma) + n_y(\gamma)$.

3.11. — Definizione Date due popolazioni $x, y \in \Gamma^*$, l'*intersezione* di x e y è la popolazione $z = x \sqcap y \in \Gamma^*$ tale che, per ogni $\gamma \in \Gamma$, $n_z(\gamma) = \min\{n_x(\gamma), n_y(\gamma)\}$.

3.12. — Definizione Date due popolazioni $x, y \in \Gamma^*$, la *differenza* di x e y è la popolazione $d = x \setminus y \in \Gamma^*$ tale che, per ogni $\gamma \in \Gamma$, $n_d(\gamma) = \max\{0, n_x(\gamma) - n_y(\gamma)\}$.

Adattamento.

3.13. — La funzione di Valutazione Implicita nell'idea di evoluzione vi è la disponibilità di una misura del grado di adattamento all'ambiente da parte di un individuo. Siccome nella metafora degli algoritmi evolutivi la parte dell'ambiente viene svolta dal problema oggetto, è il fenotipo di un individuo, cioè la soluzione da esso rappresentata, ad essere valutato tramite la funzione obiettivo $c\colon \Phi \to \mathbb{R}$: in generale si assume l'esistenza di una funzione di valutazione $\mathit{Eval}\colon \Phi \to \mathbb{R}^+$, che associa ad ogni fenotipo un numero reale positivo tramite un'opportuna trasformazione F della funzione obiettivo del problema: per ogni $\mathbf{s} \in \Phi$,

$$\mathit{Eval}(\mathbf{s}) = F[c(\mathbf{s})].$$

3.14. — La funzione di Idoneità Di riflesso risulta definita una funzione di idoneità (*fitness* in inglese) per i genotipi $f: \Gamma \to \mathbb{R}^+$ come la composizione delle funzioni di espressione genetica e di valutazione: $f \equiv Eval \circ M$. Così, per ogni $\gamma \in \Gamma$,

$$f(\gamma) = Eval[M(\gamma)] = F\{c[M(\gamma)]\}.$$

La funzione di idoneità può essere estesa alle popolazioni, definendo $f: \Gamma^* \to \mathbb{R}$ come l'idoneità media degli individui che fanno parte di una popolazione, cioè, per ogni $x \in \Gamma^*$:

$$f(x) \equiv \sum_{\gamma \in \Gamma} f(\gamma) q_x(\gamma) = \frac{1}{n} \sum_{\gamma \in \Gamma} f(\gamma) n_x(\gamma).$$

Un'altra misura del grado di adattamento di una popolazione è data dall'idoneità massima $f^*: \Gamma^* \to \mathbb{R}$ degli individui che la compongono: per ogni $x \in \Gamma^*$,

$$f^*(x) \equiv \max_{\gamma \in x} f(\gamma).$$

§ 4. — Le Strategie Evolutive.

Rappresentazione e idoneità — mutazione — ricombinazione — selezione — la regola del quinto dei successi — convergenza.

Rappresentazione e idoneità.

Le Strategie Evolutive [131, 124, 132] affrontano problemi di ottimizzazione funzionale nello spazio reale ad l dimensioni sfruttando una codifica reale dei parametri della funzione obiettivo del problema oggetto.

4.1. — Nelle Strategie Evolutive i fenotipi sono vettori l-dimensionali, cioè $\Phi = \mathbb{R}^l$. Un genotipo è costituito dallo stesso vettore del fenotipo associato, al quale si possono aggiungere fino ad l varianze $c_{ii} = \sigma_i^2$, con $i = 1, \ldots, l$, e fino a un massimo di $l(l-1)/2$ covarianze c_{ij}, con $i, j = 1, \ldots, l$, della distribuzione normale congiunta a l-dimensioni con valore atteso il vettore $\mathbf{0}$, con funzione di densità, per ogni $\mathbf{z} \in \mathbb{R}^l$,

$$p(\mathbf{z}) = \sqrt{\frac{\det \mathbf{C^{-1}}}{(2\pi)^l}} e^{-\frac{1}{2}\mathbf{z}^T \mathbf{C}^{-1} \mathbf{z}},$$

dove $\mathbf{C} = (c_{ij})$ rappresenta la matrice varianza/covarianza. La scelta di una distribuzione normale, che verrà usata per perturbare i genotipi, è ovviamente arbitraria. Quindi, in tutto, un individuo conterrà $k \leq l(l+1)/2$ parametri relativi alla "strategia" oltre agli l parametri del problema oggetto, per cui in generale $\Gamma = \mathbb{R}^{k+l}$; spesso,

tuttavia, vengono considerate solo le varianze, e allora $\Gamma = \mathbb{R}^{2l}$, mentre a volte è sufficiente considerare un'unica varianza valida per tutti i parametri del problema oggetto, per cui $\Gamma = \mathbb{R}^{l+1}$.

4.2. — Nel caso in cui siano prese in considerazione solo le varianze, per assicurare la definita positività della matrice $\mathbf{C}$ l'algoritmo usa gli angoli di rotazione equivalenti α_{ij}, definiti dalla relazione

$$\tan 2\alpha_{ij} = \frac{2c_{ij}}{\sigma_i^2 - \sigma_j^2},$$

a partire dai quali si possono calcolare le singole covarianze. Inoltre, nell'implementazione dell'algoritmo, alle varianze σ_i^2 vengono sostituite le deviazioni standard σ_i, interpretabili come i passi medi di mutazione per il parametro i. Un genotipo ha quindi la forma $\gamma = (\mathbf{z}, \sigma, \alpha) \in \Gamma$, dove $\mathbf{z} \in \mathbb{R}^l$ e σ e α rappresentano rispettivamente le deviazioni standard e gli angoli di rotazione.

4.3. — L'idoneità di un individuo $\gamma \in \Gamma$ viene ottenuta scalando il valore della funzione obiettivo in modo che risulti positivo e, a volte, sovrapponendo un rumore casuale descritto come una variabile aleatoria W,

$$f(\gamma) = F\{c[M(\gamma)], W\},$$

dove F scala la funzione obiettivo e la combina in qualche modo con una realizzazione di W. Si noti che per scalare la funzione obiettivo non è necessario conoscerne gli estremi; si consideri per esempio

$$F(x, w) = e^{x+w},$$

che darebbe luogo alla funzione di idoneità $f(\gamma) = e^{c[M(\gamma)]+W}$.

Mutazione.

4.4. — Nella forma più generale nel quadro delle Strategie Evolutive (quindi con distribuzione normale) l'operatore di mutazione perturba un individuo $\gamma = (\mathbf{z}, \sigma, \alpha)$ modificando prima casualmente le deviazioni standard e gli angoli di rotazione e poi i parametri del problema oggetto secondo le nuove funzioni di densità di probabilità fornite dagli altri parametri, producendo così un nuovo individuo $\gamma' = (\mathbf{z}', \sigma', \alpha')$; le leggi di queste variazioni casuali sono le seguenti, per ogni $i = 1, \ldots, l$ e $j = 1, \ldots, l(l-1)/2$:

$$\begin{aligned} \sigma_i' &= \sigma_i \cdot e^{\tau' \mathcal{N}(0,1) + \tau \mathcal{N}_i(0,1)} \\ \alpha_j' &= \alpha_j + \beta \mathcal{N}_j(0,1) \\ \mathbf{z}' &= \mathbf{z} + \mathcal{N}(\mathbf{0}, \sigma', \alpha'), \end{aligned}$$

dove $\mathcal{N}(0,1)$ denota una variabile aleatoria con distribuzione normale con media 0 e varianza 1, mentre $\mathcal{N}(\mathbf{0}, \sigma', \alpha')$ denota un vettore aleatorio con distribuzione congiunta normale con media il vettore $\mathbf{0}$ e matrice di varianza/covarianza ricavata dalle deviazioni standard σ' e dagli angoli di rotazione α'.

4.5. — Le mutazioni dei parametri del problema oggetto possono essere linearmente correlate secondo i valori di α, mentre σ fornisce delle metriche scalate in modo diverso per ciascun asse dello spazio delle soluzioni. Il fattore τ' controlla la variazione globale della mutabilità, τ influenza le variazioni individuali dei "passi medi" σ_i e β controlla il cambiamento della matrice varianza/covarianza. Questi parametri esogeni sono abbastanza robusti, nel senso che una loro variazione non influenza in modo sostanziale il funzionamento dell'algoritmo; i valori suggeriti da Schwefel [132] sulla base di valutazioni empiriche sono i seguenti:

$$\tau \propto \frac{1}{\sqrt{2\sqrt{n}}}, \quad \tau' \propto \frac{1}{\sqrt{2n}}, \quad \beta \approx 0.0873.$$

Questo meccanismo di mutazione fa sì che l'algoritmo evolva autonomamente i parametri relativi alla "strategia" durante la ricerca della soluzione, sfruttando il legame implicito che esiste tra un modello interno appropriato e valori di idoneità elevati; l'evoluzione e l'adattamento dei parametri interni della strategia che ne consegue è stato chiamato *autoadattamento* [133].

Ricombinazione.

4.6. — Le Strategie Evolutive utilizzano diversi meccanismi di ricombinazione, che nel caso più semplice producono un inidividuo figlio a partire da una coppia di genitori oppure, nel caso globale, possono formare il nuovo individuo utilizzando componenti prese da tutti gli individui presenti nella popolazione. Ovviamente la ricombinazione interessa tutto il genotipo e non si limita ai soli parametri del problema oggetto, anche se il meccanismo può variare a seconda che agisca sui parametri, sulle deviazioni standard o sugli angoli di rotazione.

4.7. — I meccanismi di ricombinazione più utilizzati sono la ricombinazione discreta e quella cosiddetta intermedia: nella prima ciascuna componente viene copiata da un genitore a caso; nella seconda il valore di ciascuna componente nell'individuo figlio è una combinazione lineare delle relative componenti di tutti i genitori che partecipano all'operazione: storicamente questa combinazione è semplicemente la media, ma nulla vieta di immaginarne altri tipi più generali, che infatti sono stati proposti di recente.

Si è osservato empiricamente che i migliori risultati si ottengono applicando la ricombinazione discreta ai parametri del problema oggetto e quella intermedia ai parametri della strategia. Inoltre si è dimostrato che la ricombinazione di questi ultimi è necessaria affinché si attuino i fenomeni di autoadattamento.

Selezione.

4.8. — La selezione nelle Strategie Evolutive avviene in modo completamente deterministico secondo due modalità alternative, che definiscono due classi di strategie,

(n, m) e $(n + m)$. Nelle strategie (n, m) ad ogni generazione da una popolazione di n individui ne vengono prodotti $m > n$ nuovi e di questi i migliori n vengono mantenuti per produrre la generazione successiva: gli n individui genitori vengono sempre scartati per far posto ai figli migliori. Nelle strategie $(n + m)$, invece, gli individui che sopravvivono per la generazione successiva sono i migliori n tra gli m nuovi e gli n già presenti nella popolazione: in sostanza una strategia $(n + m)$ non scarta mai le soluzioni migliori (elitismo) e quindi garantisce un miglioramento monotono della popolazione; tuttavia una strategia del genere fa fatica a reagire a problemi che cambiano nel tempo e non supporta in maniera soddisfacente l'evoluzione dei parametri della strategia, in particolare per popolazioni di cardinalità n bassa, ragione per cui in genere oggi si preferiscono strategie (n, m), dove gli esperimenti indicano un rapporto ottimale $n/m \approx \frac{1}{7}$ [133].

Risultati teorici.

4.9. — Regola del quinto di successi ($\frac{1}{5}$-*success rule*) Questa regola, formulata da Rechenberg (cfr. [124], pag. 122), fornisce un metodo per controllare la deviazione standard a partire dalla frequenza osservata di mutazioni che risultano in un miglioramento dell'individuo ad esse sottoposto (successi):

> La frazione ottimale dei successi su tutte le mutazioni è $\frac{1}{5}$. Se è maggiore di $\frac{1}{5}$, aumenta la deviazione standard; se è minore, diminuiscila.

4.10. — Giustificazione Rechenberg ricavò questa regola studiando due modelli di problema: il "corridoio", di larghezza b,

$$\min c(\mathbf{z}) = c_0 + c_1 z_1,$$

soggetto ai vincoli

$$-\frac{b}{2} \leq z_i \leq \frac{b}{2}, \quad i = 2, \ldots, l,$$

e il modello "sferico" centrato in $\mathbf{z}^*$

$$\min c(\mathbf{z}) = c_0 + c_1 \sum_{i=1}^{l} (z_i - z_i^*)^2 = c_0 + c_1 r^2,$$

dove r è la distanza euclidea tra $\mathbf{z}$ e $\mathbf{z}^*$. Si noti che quello del corridoio è un problema illimitato. Ciò tuttavia non costituisce un problema in questo contesto, poiché quello che si vuole studiare è il tasso di miglioramento della popolazione e non la sua convergenza all'ottimo, che, in questo caso, non esiste.

Per entrambe le funzioni obiettivo si calcolano i valori attesi φ_c e φ_s dei tassi di convergenza della strategia, espressi in termini delle quantità adimensionali e normalizzate

$$\bar{\sigma}_c = \frac{\sigma_c l}{b}, \quad \bar{\varphi}_c = \frac{\varphi_c l}{b}, \quad \bar{\sigma}_s = \frac{\sigma_s l}{r} \quad \text{e} \quad \bar{\varphi}_s = \frac{\varphi_s}{r}$$

come segue, per $l \gg 1$:

$$\bar{\varphi}_c \approx \frac{\bar{\sigma}_c}{\sqrt{2\pi}} e^{-\sqrt{\frac{2}{\pi}}\bar{\sigma}_c}$$
$$\bar{\varphi}_s = \frac{\bar{\sigma}_s}{\sqrt{2\pi}} e^{-\frac{\bar{\sigma}_s^2}{8}} - \frac{\bar{\sigma}_s^2}{4}\left[1 - \operatorname{erf}\left(\frac{\bar{\sigma}_s}{\sqrt{8}}\right)\right].$$

Ora, è possibile determinare le deviazioni standard ottimali $\bar{\sigma}_c^*$ e $\bar{\sigma}_s^*$ ponendo

$$\left.\frac{d\bar{\varphi}_c}{d\bar{\sigma}_c}\right|_{\bar{\sigma}_c^*} = 0 \quad \text{e} \quad \left.\frac{d\bar{\varphi}_s}{d\bar{\sigma}_s}\right|_{\bar{\sigma}_s^*} = 0$$

e calcolare di conseguenza i tassi massimi di convergenza sostituendo le deviazioni standard ottimali nelle espressioni precedenti:

$$\bar{\sigma}_c^* = \sqrt{\tfrac{\pi}{2}} \approx 1.253 \qquad \bar{\sigma}_s^* \approx 1.224$$
$$\bar{\varphi}_c^* = \tfrac{1}{2e} \approx 0.184 \qquad \bar{\varphi}_s^* \approx 0.2025.$$

Inoltre si calcolano le probabilità che una mutazione comporti un miglioramento dell'individuo mutato per entrambe le funzioni, ottenendo, per $l \gg 1$,

$$p_c \approx \frac{1}{2} e^{-\sqrt{\frac{2}{\pi}}\bar{\sigma}_c} \quad \text{e} \quad p_s = \frac{1}{2}\left[1 - \operatorname{erf}\left(\frac{\bar{\sigma}_s}{\sqrt{8}}\right)\right].$$

Ne risulta che le probabilità ottimali di successo, quando le deviazioni standard sono, rispettivamente, $\bar{\sigma}_c^*$ e $\bar{\sigma}_s^*$, valgono

$$p_c^* = \frac{1}{2e} \approx 0.184 \quad \text{e} \quad p_s^* \approx 0.270,$$

entrambe molto vicine al valore di $\frac{1}{5}$ proposto dalla regola. ∎

4.11. — Ci sono due problemi per quanto riguarda l'uso della Regola 4.9. Il primo sorge quando la conformazione topologica della funzione obiettivo è tale per cui il tasso di successo rimane comunque al di sotto di $\frac{1}{5}$ anche abbassando la deviazione standard a zero: questo è spesso il caso lungo i vincoli attivi o in corrispondenza di frontiere di discontinuità o di non differenziabilità.

Il secondo consiste nel fatto che la Regola 4.9 non dà alcun suggerimento su come trattare le singole deviazioni standard σ_i individualmente e perciò non permette di scalare il passo medio di mutazione in modo differente lungo assi distinti del sistema di coordinate. Ne segue che questo risultato è veramente utile solo per strategie con una sola deviazione standard, che non usano una vera popolazione, la ricombinazione o l'autoadattamento.

4.12. — Il tasso di convergenza L'espressione generale del tasso di convergenza, cioè il tasso atteso di miglioramento dell'idoneità media della popolazione $\varphi_t = E[f(X_{t+1})/f(X_t)|X_t]$ per le strategie (n, m) ed $(n + m)$ facenti uso di un'unica

deviazione standard e senza ricombinazione od autoadattamento fu ricavata da Schwefel [132]:

$$\varphi = \frac{m}{n} \int_{\Delta_{\min}}^{\infty} \Delta \sum_{i=1}^{n} \binom{m-1}{i-1} P[\Delta' = \Delta] P[\Delta' \geq \Delta]^{i-1} P[\Delta' < \Delta]^{m-i} d\Delta,$$

dove $\Delta_{\min}$ vale $-\infty$ per le strategie (n, m) e zero per le strategie $(n + m)$ e $P[\Delta' = \Delta]$, $P[\Delta' \geq \Delta]$ e $P[\Delta' < \Delta]$ sono le probabilità che il miglioramento $\Delta' = f(\gamma') - f(\gamma)$ di uno degli m individui figli γ', ottenuto per mutazione dall'individuo genitore γ, sia, rispettivamente, Δ, almeno Δ e minore di Δ.

4.13. — Volendo determinare φ per una strategia $(1, m)$, siccome l'integrale non può essere risolto per via analitica nemmeno nel caso $n = 1$, Schwefel dovette imporre alcune ipotesi aggiuntive e, alla fine, arrivò a stabilire i valori ottimali $m_c^* = 6$ e $m_s^* = 4.7 \approx 5$ per i due modelli di problema studiati in 4.10, che massimizzano le funzioni φ_c^*/m_c e φ_s^*/m_s, fornendo così il compromesso più ragionevole tra onere computazionale, rappresentato dal numero di individui valutati ad ogni generazione, e tasso di miglioramento, quando la strategia $(1, m)$ è implementata su una macchina sequenziale.

4.14. — Convergenza Un recente risultato per le Strategie Evolutive è un teorema di convergenza in probabilità per la strategia $(1 + 1)$ dimostrato da Günter Rudolph [126]. Denotando con P_t la probabilità che l'algoritmo raggiunga l'insieme di livello $\Gamma_{f^*-\epsilon} \subset \Gamma$ alla t-esima generazione, posto che $\sum_{t=0}^{\infty} P_t = \infty$ e che $\mu(\Gamma_{f^*-\epsilon}) > 0$, dove μ è la misura di Lebesgue su Γ, e che la distribuzione di probabilità usata per la mutazione abbia densità continua, si dimostra che

$$P[\lim_{t \to \infty} \gamma_t \in \Gamma_{f^*-\epsilon}] = 1.$$

La dimostrazione può essere estesa al caso generale $(n + m)$, ma non alle strategie (n, m).

Sebbene un tale risultato non abbia delle particolari conseguenze pratiche, tuttavia esso dimostra che questa classe di metodi soddidfa i requisiti minimi di un algoritmo di ottimizzazione globale.

§ 5. — La Programmazione Evolutiva.

Rappresentazione e idoneità — mutazione — selezione — convergenza.

La Programmazione Evolutiva [47, 48] è un approccio all'Intelligenza Artificiale che fa uso di automi a stati finiti. Un comportamento intelligente presuppone sia la capacità di prevedere l'ambiente, sia dei meccanismi per tradurne le previsioni in reazioni commisurate al raggiungimento di uno scopo. Al massimo livello di generalità, l'ambiente è descritto come una sequenza di simboli appartenenti a un certo alfabeto finito. Il problema è dunque evolvere un algoritmo che operi su una sequenza di simboli

osservati e produca un simbolo in uscita che ne massimizzi le prestazioni in rapporto al simbolo successivo nell'ambiente secondo una ben definita funzione di ricompensa.

Rappresentazione e idoneità.

5.1. — Un automa a stati finiti [68] è una quintupla $\langle Q, \Sigma, Z, \delta, \omega \rangle$, dove Q è l'insieme degli stati interni, Σ è l'alfabeto d'ingresso, cioè l'ambiente, Z è l'alfabeto di uscita, $\delta: \Sigma \times Q \to Q$ è la funzione di transizione che mappa uno stato nello stato successivo e $\omega: \Sigma \times Q \to Z$ è la funzione di uscita.

Nella Programmazione Evolutiva lo spazio dei fenotipi Φ è l'insieme degli automi a stati finiti con Q, Σ e Z fissati a seconda del problema affrontato. Lo spazio dei genotipi Γ consiste in descrizioni delle funzioni δ e ω, tipicamente sotto forma di tabelle $\|Q\| \times \|Z\|$ con una coppia (q, a), $q \in Q$ e $a \in Z$, in ciascuna casella; spesso vengono usate codifiche più compatte quando la natura del problema lo consente.

5.2. — L'idoneità degli automi presenti in una popolazione viene calcolata esponendoli all'ambiente, cioè a una sequenza osservata di simboli. Un simbolo per volta viene dato in pasto a ciascun automa e il simbolo da questo prodotto in uscita viene confrontato con il simbolo successivo nella sequenza; l'accuratezza della previsione viene misurata in relazione alla funzione di ricompensa assegnata (per esempio tutto o niente, errore assoluto, errore quadratico o qualsiasi altra espressione del significato dei simboli). Quando tutti i simboli della sequenza sono stati letti, l'idoneità di ogni automa è data da una funzione delle singole ricompense (per esempio la ricompensa media per simbolo).

Mutazione.

5.3. — Nuovi automi vengono generati per mutazione casuale da ciascun automa già presente nella popolazione: tipicamente ciascun genitore produce un solo figlio. Esistono cinque modalità di mutazione suggerite dalla descrizione di un automa:

1. sostituzione di un simbolo di uscita;
2. sostituzione di uno stato nella funzione di transizione;
3. aggiunta di un nuovo stato interno;
4. eliminazione di uno stato interno;
5. cambiamento dello stato iniziale.

Queste operazioni sono soggette a vincoli sul numero massimo e minimo degli stati interni; la mutazione opera secondo una distribuzione di probabilità assegnata, tipicamente uniforme e inoltre il numero di mutazioni per individuo generato può a sua volta essere governato da una legge di probabilità, per esempio una distribuzione di Poisson.

Nella sua formulazione originale la Programmazione Evolutiva non prevede un operatore di ricombinazione.

Selezione.

5.4. — Come nel caso delle Strategie Evolutive, la selezione nella Programmazione Evolutiva è deterministica e consiste nello scartare, dei $2n$ automi ottenuti a partire da una popolazione di n genitori producendo n nuovi automi per mutazione, gli n peggiori: usando una notazione analoga a quella utilizzata per le Strategie Evolutive, si può dire che la Programmazione Evolutiva è un Algoritmo Evolutivo di tipo $(n + n)$.

Risultati teorici.

5.5. — CONVERGENZA Nella sua Tesi [45], David B. Fogel analizza un algoritmo standard di Programmazione Evolutiva e dimostra un teorema di convergenza globale in probabilità.

Tale risultato viene ricavato definendo una catena di Markov sullo spazio discreto degli stati; combinando tutte le possibili popolazioni che contengono l'automa ottimo, si definisce uno stato assorbente in cui il processo alla fine rimane intrappolato, per via del carattere elitista della selezione.

§ 6. — Gli Algoritmi Genetici.

Rappresentazione — idoneità — mutazione e ricombinazione — selezione — il Teorema degli Schemi — parallelismo implicito.

Rappresentazione.

6.1. — Gli Algoritmi Genetici propriamente detti richiedono che le soluzioni siano codificate sotto forma di stringhe di cifre binarie di lunghezza fissata l e quindi lo spazio dei genotipi su cui operano è $\Gamma = \{0, 1\}^l$. Da un certo punto di vista questa è la codifica più generale, poiché qualsiasi altra struttura dati viene compilata, nella memoria dell'elaboratore, in una sequenza di bit, ma anche la meno informativa, poiché qualsiasi nozione della struttura logica di una soluzione viene persa e non può essere sfruttata dagli operatori genetici.

È chiaro che, in questo contesto, la funzione di espressione genetica (cfr. §3.3) assume una rilevanza centrale per l'implementazione dell'algoritmo e, infatti, vedremo che il problema della rappresentazione è il fattore critico nell'applicazione degli Algoritmi Genetici.

6.2. — A titolo illustrativo, per sostanziare con un semplice esempio l'esposizione, si supponga di dover risolvere un problema di ottimizzazione di N parametri continui della forma

$$\min c(\mathbf{z}),$$

con vincoli

$$u_i \leq z_i \leq v_i, \quad i = 1, \ldots, N.$$

In questo caso, la stringa di cifre binarie viene divisa logicamente in N segmenti, il più delle volte di uguale lunghezza l/N, e ciascun segmento viene interpretato come una codifica binaria della corrispondente variabile $z_i \in [u_i, v_i]$ del problema oggetto, per cui una funzione di decodifica del segmento i, $M_i\colon \{0,1\}^{l/N} \to [u_i, v_i]$, tipicamente sarà definita come

$$M_i(\gamma_1 \ldots \gamma_{l/N}) = u_i + \frac{v_i - u_i}{2^{l/N} - 1} \sum_{j=1}^{l/N} \gamma_j 2^{j-1},$$

dove le γ_j, per $j = 1, \ldots, l/N$, sono le cifre binarie che compongono l'i-esimo segmento di un individuo $\gamma \in \Gamma$. Un'altra possibilità, che oggi è preferita a questa semplice codifica binaria, è quella di utilizzare un codice di Gray. I codici di Gray sono codici binari caratterizzati dal fatto che le codifiche di due numeri interi successivi hanno sempre una distanza di Hamming unitaria, cioè differiscono in una sola posizione. Le prime investigazioni sull'uso di questo tipo di codici negli Algoritmi Genetici sono dovute a Hollstien [66] e Bethke [15]; Caruana e Schaffer [27] presentano e analizzano dell'evidenza sperimentale che indica una generale superiorità dei codici di Gray rispetto alla codifica binaria quando i parametri codificati sono ordinali.

La funzione di espressione genetica M si ottiene combinando le funzioni M_i di decodifica dei singoli segmenti:

$$M(\gamma) = (M_1(\gamma), \ldots, M_N(\gamma)) \in \Phi.$$

6.3. — In termini di genetica molecolare, c'è un'analogia tra l'alfabeto discreto delle basi che compongono i nucleotidi (adenina, citosina, guanina e timina) i quali, concatenati in due sequenze parallele e complementari, formano il DNA, e le cifre binarie che compongono un genotipo negli Algoritmi Genetici. Il DNA viene decodificato in varie fasi in amminoacidi, quindi in proteine e, infine, attraverso tutto l'apparato epigenetico, nel fenotipo corrispondente. Lo schema di rappresentazione utilizzato dagli Algoritmi Genetici può dunque essere considerato, tra gli Algoritmi Evolutivi, quello che più si avvicina al modello naturale che ne è la fonte di ispirazione.

Idoneità.

6.4. — Il valore di idoneità di un individuo $\gamma \in \Gamma$, in generale, si ottiene a partire dal valore della funzione obiettivo del fenotipo corrispondente, $c[M(\gamma)]$, scalato in modo opportuno da una funzione di scala F. Alcuni metodi comunemente adottati per scalare l'idoneità sono i seguenti [54]:

1. Scalatura lineare. L'idoneità scalata f' viene calcolata, per ogni $\gamma \in \Gamma$, come

   $$f'(\gamma) = af(\gamma) + b,$$

   dove $a > 0$ e b sono due costanti opportunamente scelte in modo che l'idoneità scalata sia positiva; tipicamente a e b vengono determinate ad ogni generazione in

modo tale che il valore massimo M dell'idoneità scalata sia un numero piccolo, per esempio $M = 2f'(x)$, dove $f'(x)$ è l'idoneità scalata media della popolazione. In altre parole, a e b sono scelte tra le soluzioni del sistema

$$\begin{cases} a = b\frac{(M-1)}{f^*(x)-Mf(x)} \\ b \geq -af_{\min}(x). \end{cases}$$

2. σ-Troncamento. È una modifica della scalatura lineare, per cui viene calcolata la deviazione standard σ dei valori di idoneità nella popolazione; l'idoneità scalata f' è quindi calcolata, per ogni $\gamma \in \Gamma$, come

$$f'(\gamma) = f(\gamma) - [f(x) - c\sigma],$$

dove c è una costante piccola, tipicamente con valori nell'intervallo $[1, 3]$. Gli individui la cui idoneità è inferiore alla media di almeno c deviazioni standard vengono semplicemente scartati.
3. Scalatura per classifica. Gli individui della popolazione vengono ordinati per valori di idoneità decrescente e l'idoneità scalata viene assegnata ad ogni individuo in base alla posizione occupata in classifica. Lo scopo di questa tecnica è di preservare una certa diversità evitando che un individuo molto più idoneo degli altri proliferi eccessivamente a discapito del resto della popolazione.
4. Scalatura con legge esponenziale. L'idoneità viene scalata per mezzo di una legge esponenziale

$$f'(\gamma) = f(\gamma)^{1+k},$$

con $k \ll 1$. Come è stato dimostrato in [53], la scelta di k dipende dal problema oggetto e alcuni (per esempio Michalewicz [103]) fanno variare k con l'età della popolazione.

Mutazione e ricombinazione.

6.5. — La mutazione modifica una singola cifra binaria del genotipo ed entra in azione al momento della riproduzione e della ricombinazione, cioè ogniqualvolta del materiale genetico debba essere copiato da un individuo esistente ad uno nuovo.

La mutazione, negli Algoritmi Genetici, è vista tradizionalmente come un operatore di secondo piano e applicata con probabilità molto basse, dell'ordine di una cifra binaria modificata su mille copiate.

6.6. — L'enfasi è piuttosto centrata sull'incrocio (*crossover*), l'operatore di ricombinazione degli Algoritmi Genetici, come il principale strumento per l'esplorazione dello spazio di ricerca.

Esistono diverse varianti dell'incrocio; quella proposta da Holland [65], nota con il nome di "incrocio a un punto" (*one-point crossover*), crea un nuovo individuo χ a partire da due individui genitori $\gamma = (\gamma_1 \dots \gamma_l)$ e $\kappa = (\kappa_1 \dots \kappa_l)$, estraendo un punto di

incrocio I a caso con probabilità uniforme su $\{1, \ldots, l-1\}$, quindi prendendo le prime I cifre binarie da γ e le restanti $l - I$ da κ, cosicché l'individuo figlio risulta

$$\chi = (\gamma_1 \ldots \gamma_I \kappa_{I+1} \kappa_l).$$

Spesso, per una questione di simmetria, vengono generati due figli γ' e κ' contemporaneamente, anziché uno, in modo speculare, cioè

$$\gamma' = (\gamma_1 \ldots \gamma_I \kappa_{I+1} \kappa_l) \quad \text{e} \quad \kappa' = (\kappa_1 \ldots \kappa_I \gamma_{I+1} \gamma_l).$$

Questo incrocio a un punto può essere esteso, in modo naturale, a quello a k punti [78], scegliendo k punti di incrocio distinti e trascrivendo in modo alternato i segmenti di genotipo così definiti; a un estremo si ha poi l'incrocio cosiddetto "uniforme" [135], in cui ciascuna cifra binaria può essere presa, indipendentemente dalle altre e con probabilità uniforme, dall'uno o dall'altro dei due individui genitori.

L'incrocio opera al livello del genotipo, cioè della codifica binaria, ignorando i confini logici tra i vari segmenti che lo compongono.

Selezione.

6.7. — Negli Algoritmi Genetici vengono estratti dalla popolazione corrente n individui genitori per la generazione successiva tramite n estrazioni indipendenti, secondo una legge probabilistica assegnata. Il concetto di base è che un individuo migliore deve avere una probabilità di sopravvivere e di riprodursi maggiore di quella di un individuo peggiore. Questo si concretizza, nella versione classica dell'algoritmo, nella legge di selezione *proporzionale all'idoneità* (*fitness-proportionate selection*)

$$P_x[\gamma] = q_x(\gamma) \frac{f(\gamma)}{f(x)},$$

dove con $P_x[\gamma]$ si è denotata la probabilità che un individuo con genotipo γ venga estratto dalla popolazione x. Holland deriva questa legge utilizzando un concetto della Teoria dei Giochi, l'allocazione ottima dei tentativi, attraverso l'analisi del problema della slot-machine a k leve (cfr. §9.3).

L'applicazione pratica degli Algoritmi Genetici ha mostrato pregi e difetti di questa strategia e ne sono state proposte e utilizzate con successo molte varianti o, addirittura, alternative, che verranno illustrate più avanti, nel §9.

Il Teorema degli Schemi.

Un concetto importante per l'analisi degli Algoritmi Genetici è quello di *schema*.

6.8. — Definizione Uno *schema* è un sottoinsieme $S \subseteq \Gamma$ rappresentato da una stringa modello formata da simboli nell'insieme $\{0, 1, \star\}$, dove "$\star$" svolge il ruolo

di simbolo "jolly": uno schema dunque contiene tutte le stringhe che concordano con la sua stringa modello in tutte le posizioni non marcate con il simbolo "$\star$".

Esistono in Γ esattamente 3^l schemi distinti. Ogni schema induce una bipartizione di Φ e $M(S) \subseteq \Phi$ è chiamato *iperpiano* definito da S.

6.9. — Definizione L'*ordine* $o(S)$ di uno schema S è definito come il numero delle posizioni fisse (0 o 1) nella stringa modello che lo rappresenta. La cardinalità di uno schema S è legata al suo ordine dalla relazione $\|S\| = 2^{l-o(S)}$.

6.10. — Idoneità di uno schema L'idoneità *assoluta* di uno schema S è

$$f(S) \equiv \frac{1}{\|S\|} \sum_{\gamma \in S} f(\gamma).$$

Questa quantità è l'idoneità attesa di un individuo estratto a caso, con probabilità uniforme, da S.

L'idoneità *relativa* di uno schema S rispetto a una popolazione $x \in \Gamma^*$ è

$$f_x(S) \equiv \frac{1}{q_x(S)} \sum_{\gamma \in S} q_x(\gamma) f(\gamma).$$

Questa quantità è l'idoneità attesa di un individuo estratto a caso dalla popolazione x, dato che appartenga allo schema S.

6.11. — Definizione La *lunghezza di definizione* $\delta(S)$ di uno schema S è la distanza tra la prima e l'ultima posizione fissa nella stringa modello che lo rappresenta. La lunghezza di definizione può essere interpretata come una misura della "compattezza" dell'informazione presente in uno schema.

6.12. — Teorema (Degli Schemi) Si denoti con $\{X_t\}_{t=0,1,\ldots}$ la successione di popolazioni generate dall'Algoritmo Genetico alle generazioni $t = 0, 1, \ldots$; supposto costante il rapporto

$$c = \frac{f_{X_t}(S) - f(X_t)}{f(X_t)}, \quad t = 0, 1, \ldots,$$

si ha

$$E[q_{X_t}(S)|X_0] \geq q_{X_0}(S)(1+c)^t \left(1 - p_c \frac{\delta(S)}{l-1} - o(S)p_m\right)^t,$$

dove p_c e p_m sono, rispettivamente, il tasso di incrocio e di mutazione.

In altre parole, ci si aspetta che uno schema corto, di ordine basso e sopra la media riceva un numero esponenzialmente crescente di istanze in generazioni successive.

6.13. — Dimostrazione Ricaviamo prima di tutto il valore atteso condizionato $E[q_{X_t}(S)|X_{t-1}]$ tenendo conto degli effetti cumulativi dell'incrocio e della mutazione. Usando la selezione proporzionale all'idoneità, abbiamo che

$$E[q_{X_t}(S)|X_{t-1}] \geq q_{X_{t-1}}(S) \frac{f_{X_{t-1}}}{f(X_{t-1})} P_{\text{surv}}[S] = q_{X_{t-1}}(S)(1+c)P_{\text{surv}}[S],$$

dove $P_{\text{surv}}[S]$ denota la probabilità che le posizioni fisse dello schema S vengano lasciate intatte dall'incrocio e dalla mutazione. Chiaramente $P_{\text{surv}}[S] \geq P_{sc}[S]P_{sm}[S]$, dove $P_{sc}[S]$ è la probabilità che S sopravviva all'incrocio e $P_{sm}[S]$ è la probabilità che S sopravviva alla mutazione, poiché potrebbe darsi che per un caso fortuito la mutazione restauri una parte di S corrotta dall'incrocio.

Calcoliamo dunque $P_{sc}[S]$ e $P_{sm}[S]$. Dovrebbe essere chiaro che la lunghezza di definizione di uno schema gioca un ruolo significativo nella probabilità della sua sopravvivenza: il punto d'incrocio viene scelto uniformemente tra $l-1$ punti possibili, quindi la probabilità che il punto scelto non divida in due la parte fissa di uno schema è data da $1 - \frac{\delta(S)}{l-1}$; ora, c'è una probabilità p_c che un individuo venga sottoposto all'incrocio, per cui possiamo scrivere

$$P_{sc}[S] = 1 - p_c \frac{\delta(S)}{l-1}.$$

Per quanto riguarda la mutazione, la probabilità che ciascuna singola posizione di un genotipo venga alterata è p_m e, dato che la mutazione o meno di ciascuna posizione è indipendente da quella delle altre, la probabilità che nessuna posizione fissa di S venga alterata è

$$P_{sm}[S] = (1-p_m)^{o(S)} \approx 1 - o(S)p_m,$$

poiché $p_m \ll 1$. Questo ci permette di scrivere

$$E[q_{X_t}(S)|X_{t-1}] \geq q_{X_{t-1}}(S)(1+c)\left(1 - p_c\frac{\delta(S)}{l-1} - o(S)p_m\right).$$

Sostituendo $E[q_{X_{t-1}}(S)|X_{t-2}]$ a $q_{X_{t-1}}(S)$ e iterando il ragionamento, si ottiene la tesi. ■

Parallelismo implicito.

Un concetto che gioca un ruolo molto importante nella comprensione del funzionamento degli Algoritmi Genetici è quello di *parallelismo intrinseco* (come lo chiama Holland in [65]) o *implicito* (come è oggi universalmente noto).

6.14. — Il parallelismo implicito si riferisce al fatto che una popolazione di individui estratti da Γ è allo stesso tempo un campione di istanze di un gran numero di schemi e dunque si può affermare che tutti questi schemi sono implicitamente elaborati dall'algoritmo "in parallelo" tramite gli individui della popolazione.

Holland ricavò in [65] una stima di un limite inferiore al numero di schemi disgiunti presenti in una popolazione di $n = c2^l$ individui dell'ordine di $\frac{n^3}{c\sqrt{l}}$, dove c è una piccola costante intera. Di solito questo risultato viene interpretato dicendo che un algoritmo genetico, elaborando solo n individui, in realtà implicitamente elabora almeno n^3 schemi disgiunti: questo sarebbe un raro, se non l'unico, caso in cui un'esplosione combinatoria congiura a nostro favore.

Un risultato più generale è stato fornito recentemente da Alberto Bertoni e Marco Dorigo:

6.15. — Teorema [14] Sia ϵ tale che, per ogni schema S, posta $P_{sc}[S]$ la probabilità che S passi indenne l'incrocio, $P_{sc}[S] \geq 1 - \epsilon$ e sia $k = \frac{l\epsilon}{2}$; si consideri una popolazione di $n = c2^k = 2^{\beta k}$ individui estratti da Γ con probabilità uniforme, con $\beta > 0$. Allora

1. il numero atteso di schemi disgiunti con lunghezza di definizione $\delta(S) \leq 2k$ presenti nella popolazione è almeno dell'ordine di $\frac{n^{K(\beta)}}{\sqrt{\log_2 n}}$, dove

$$K(\beta) = \begin{cases} 1 + \frac{2}{\beta}, & 0 < \beta < 1, \\ \frac{H(\beta/2)}{\beta}, & 1 \leq \beta \leq \frac{4}{3}, \\ \frac{2\log_2 3}{\beta}, & \beta > \frac{4}{3}, \end{cases}$$

   dove $H(\xi) = \xi \log \frac{1}{\xi} + (1 - \xi) \log \frac{1}{1-\xi}$ è la Funzione Entropia;
2. per $\beta \geq 1$, tale limite inferiore è ottimo a meno di una costante;
3. per $\beta \geq 1$, con probabilità $P \geq 1 - \frac{2}{e}$ il numero di schemi disgiunti presenti nella popolazione è maggiore della metà del limite inferiore definito al punto 1.

6.16. — Dimostrazione Si veda [14]. ∎

§ 7. — La Programmazione Genetica.

Rappresentazione e idoneità — mutazione — ricombinazione — selezione.

Rappresentazione e idoneità.

7.1. — Nella Programmazione Genetica le strutture che prendono parte alla simulazione del processo evolutivo sono programmi scritti in un linguaggio di programmazione semplificato, spesso dotato di primitive ritagliate sul problema oggetto. Storicamente, il linguaggio d'elezione per la Programmazione Genetica è un opportuno sottoinsieme del LISP, e questo per una serie di motivi (cfr. [84] §IV.3), tra i quali il principale è il fatto che in LISP dati e programmi sono rappresentati nello stesso modo, come liste di elementi che a loro volta possono essere liste, per cui risulta molto facile manipolarli preservandone la correttezza sintattica. Ciò non significa, tuttavìa, che non siano possibili esperimenti basati su altri linguaggi, che infatti non mancano in letteratura: si hanno esempi che utilizzano il C++, un linguaggio radicalmente diverso dal LISP [82], altri che sfruttano rappresentazioni più simili a quelle degli Algoritmi Genetici che vengono "compilate" in programmi al momento della valutazione [10] ed altre ancora che fanno evolvere addirittura programmi in codice macchina [113].

7.2. — Il tipo di problema che si presta ad essere trattato per mezzo della Programmazione Genetica è caratterizzato dall'induzione di una funzione. Problemi che

ricadono in questa tipologia sono, per esempio, quello del controllo ottimo, in cui deve essere determinata una strategia di controllo che determina i valori delle variabili di controllo sulla base dei valori delle variabili di stato; la pianificazione in Intelligenza Artificiale e in Robotica, in cui deve essere definito un piano d'azione sulla base di informazioni sensoriali provenienti dall'ambiente e tenendo presente un obiettivo da conseguire; l'induzione di successioni, in cui si cerca un'espressione matematica che generi tutti gli elementi di una successione della quale siano noti solo alcuni elementi iniziali; la regressione simbolica, che comporta l'identificazione di un'espressione matematica che fornisca la spiegazione, o almeno una sua buona approssimazione, dei valori assunti dalle variabili dipendenti in relazione ai valori delle variabili indipendenti in un certo campione finito di istanze.

Altri problemi di questo tipo sono la programmazione automatica, la scoperta di strategie di gioco, la previsione di serie storiche, l'integrazione e la differenziazione simbolica e i problemi inversi.

7.3. — L'idoneità di un dato programma viene calcolata eseguendolo su un certo insieme di istanze del problema oggetto, che può essere determinato a priori in modo tale da rappresentare con una certa completezza le varie possibilità che ci si può aspettare, oppure scelto a caso di volta in volta; le prestazioni del programma applicato alla singola istanza del problema vengono misurate, tipicamente per mezzo di funzioni che penalizzano la violazione di vincoli e la deviazione dalla condotta desiderata, e quindi combinate in un unico punteggio; l'idoneità complessiva del programma viene definita, in genere, come la media dei punteggi ottenuti su ciascuna singola istanza (cfr. [84], Cap. VI).

Mutazione.

7.4. — Esistono diversi operatori di mutazione utilizzati nella Programmazione Genetica. La mutazione vera e propria sceglie un nodo dell'albero sintattico che rappresenta il programma associato a un individuo, quindi sostituisce un nuovo sottoalbero generato a caso al sottoalbero avente il nodo scelto per radice.

La *permutazione* sceglie un nodo interno all'albero sintattico, quindi ne cambia in modo casuale l'ordine dei sottoalberi.

La *semplificazione* (*editing*) applica al programma associato a un individuo un certo insieme di regole di semplificazione; un esempio di regola di semplificazione universalmente valida è il seguente: se una certa funzione che non ha effetti collaterali ha come argomenti solo costanti, essa può essere sostituita da una costante, ottenuta valutando la funzione sui suoi argomenti costanti. Naturalmente, in genere, le regole di semplificazione dipendono strettamente dal problema oggetto.

Ricombinazione.

7.5. — Dati due programmi associati ad altrettanti individui, l'operatore di in-

crocio sceglie a caso, con probabilità uniforme, due nodi nei rispettivi alberi sintattici, quindi scambia i due sottoalberi aventi come radici i nodi scelti. Per come è definito, l'operatore di incrocio produce sempre due nuovi programmi sintatticamente validi.

Va notato che l'incrocio nella Programmazione Genetica differisce dall'incrocio in Natura e negli altri Algoritmi Evolutivi in un importante riguardo: incrociandosi, due individui identici, contrariamente a quello che saremmo abituati ad aspettarci, in generale producono figli diversi, sia tra loro che dai genitori. Questa caratteristica viene valutata positivamente da Koza e colleghi, in quanto riduce la tendenza alla convergenza prematura, causata cioè dalla presenza di ottimi locali e da fluttuazioni statistiche dovute all'esiguità delle popolazioni che in pratica possono essere simulate sulle macchine disponibili, che, effettivamente, è un problema che affligge gli Algoritmi Evolutivi.

Selezione.

La Programmazione Genetica parte con la creazione di una popolazione iniziale di programmi generati a caso e composti da primitive, variabili e costanti appropriate al problema trattato.

Per la selezione, la Programmazione Genetica adotta senza variazioni il modello degli Algoritmi Genetici convenzionali (cfr. §6.7), con la selezione proporzionale all'idoneità.

§ 8. — **Sintesi.**

Gli Algoritmi Evolutivi come sintesi unificante e generalizzazione dei modelli sopra presentati — il problema della rappresentazione — ottimizzazione numerica — ottimizzazione su grafi e reti — ottimizzazione di permutazioni — operatori avanzati — Algoritmi Evolutivi distribuiti e paralleli.

8.1. — Ciascuno dei modelli illustrati nei paragrafi precedenti nasce in un particolare contesto di ricerca ed è più o meno influenzato dal tipo di problema che i suoi ideatori avevano in mente di affrontare: perciò succede che le Strategie Evolutive siano adatte all'ottimizzazione di funzioni continue, la Programmazione Evolutiva a compiti di previsione, gli Algoritmi Genetici all'evoluzione di classificatori e la Programmazione Genetica alla regressione simbolica.

Chiunque si sia occupato di Algoritmi Evolutivi conosce molto bene l'importanza dello scambio di informazioni tra soluzioni diverse come principale responsabile della rapida convergenza verso l'ottimo, grazie al fatto che innovazioni vantaggiose scoperte indipendentemente e casualmente da individui diversi vengono combinate in un nuovo individuo e diffuse in tutta la popolazione, invece di attendere che il gioco del caso le

riproponga e le sovrapponga in un unico individuo.

In modo analogo, idee emerse in un contesto possono essere prese in prestito ed applicate in un altro e viceversa, a volte con successo, contribuendo così all'avanzamento della conoscenza e al miglioramento delle tecniche utilizzate.

Questo è esattamente il processo che è alla base dell'ideazione di ciascuno dei modelli illustrati nei paragrafi precedenti, cioè l'incrocio di concetti presi in prestito dalla Biologia da una parte e dalla Matematica dall'altra, e, ad un livello più basso, quello che, a partire dagli anni ottanta, ha portato diverse correnti di ricerca a convergere nello studio di quelli che oggi sono chiamati Algoritmi Evolutivi, cioè la mutua ibridazione dei vari modelli proposti.

Il problema della rappresentazione.

8.2. — Ortogonalità ed epistasi Il concetto chiave che emerge dalla ricerca di questi ultimi anni è che un genotipo può essere una qualsiasi struttura dati, possibilmente una che riesca a combinare le seguenti due caratteristiche, a volte in conflitto tra di loro:

1. rappresenti una soluzione in modo naturale, minimizzi cioè la complessità computazionale dell'algoritmo che realizza la funzione di decodifica M;
2. sia scomponibile in blocchi costitutivi, ciascuno dei quali abbia significato rispetto al problema, rispecchi cioè la struttura semantica delle soluzioni.

Il secondo requisito, in particolare, è importante in relazione agli operatori genetici, come la mutazione e la ricombinazione: se il genotipo è scomponibile in blocchi "ortogonali", cioè mutuamente indipendenti, la ricombinazione di due buone soluzioni darà, in media, un'altrettanto buona soluzione e un'ottima soluzione qualora i blocchi ereditati da ciascuno dei genitori siano quelli "giusti." La situazione contraria viene chiamata *epistasi* con un termine preso in prestito dalla Genetica, dove significa interazione tra geni, nel senso che il tipo di influenza che un gene esercita sul fenotipo dipende dalla configurazione di un insieme di altri geni.

8.3. — Nei paragrafi 4–7 sono state già presentate quattro semplici strutture dati usate in ciascuno degli algoritmi evolutivi tradizionali per rappresentare soluzioni del problema oggetto: vettori di numeri reali nelle Strategie Evolutive, matrici bidimensionali di simboli e stati nella Programmazione Evolutiva, vettori di cifre binarie negli Algoritmi Genetici ed alberi sintattici nella Programmazione Genetica. Nel seguito di questo paragrafo si illustrano alcune delle strutture dati usate negli Algoritmi Evolutivi in relazione alla tipologia del problema oggetto.

Ottimizzazione numerica.

8.4. — La rappresentazione binaria usata negli Algoritmi Genetici reca degli svantaggi se applicata a problemi di ottimizzazione funzionale nei quali sia richiesta una precisione elevata e aventi spazi di ricerca con un grande numero di dimensioni.

La ragione per la quale in passato questo tipo di rappresentazione ha goduto di ampia popolarità è che l'alfabeto binario fornisce il massimo numero di schemi per unità d'informazione [54], facilita l'analisi teorica e consente di definire gli operatori genetici in modo conciso, elegante e, soprattutto, indipendente dal problema oggetto.

Tuttavia sviluppi recenti nella Teoria degli Algoritmi Evolutivi hanno dimostrato che i risultati sul parallelismo implicito non dipendono dall'uso di stringhe binarie (cfr. [2] e [55]), per cui non c'è più motivo di escludere a priori alfabeti di cardinalità maggiore, eventualmente combinati con nuovi operatori.

Infatti, si può considerare ormai empiricamente provato (cfr. [103], Cap. V) che una codifica a vettore di numeri reali unita a operatori genetici appropriatamente definiti raggiunge prestazioni migliori sia in termini di rapidità, sia di consistenza dei risultati da esecuzione a esecuzione (minore varianza), sia di precisione (numero di cifre decimali significative). Inoltre la rappresentazione reale ha il pregio, essendo intuitivamente più vicina allo spazio del problema, di facilitare l'incapsulamento di conoscenza specifica negli operatori genetici, il che è essenziale qualora si desiderino trattare problemi con vincoli non banali.

Ottimizzazione su grafi e reti.

Esiste una vasta classe di problemi su grafi e reti in cui una soluzione è un assegnamento di quantità numeriche ad archi o vertici di un grafo $G = (V, E)$ tale da soddisfare una serie di vincoli. Un problema esemplare di questa classe è il Problema del Trasporto (cfr. [103], Cap. IX).

8.5. — Un approccio tradizionale codificherebbe un assegnamento di etichette numeriche v_i, $i = 1, \ldots, \|V\|$ e w_j, $j = 1, \ldots, \|E\|$, in un vettore di $k\|V\|\|E\|$ cifre binarie, scomponibile in $\|V\|\|E\|$ stringhe di k cifre binarie da interpretare come rappresentazioni in virgola fissa dei numeri v_i e w_j.

In generale, tuttavia, un vettore arbitrario di $k\|V\|\|E\|$ cifre binarie non corrisponde necessariamente a una soluzione ammissibile; inoltre, per preservare l'ammissibilità di una soluzione, la mutazione di uno qualsiasi degli elementi del vettore dovrà essere seguita da appropriati aggiustamenti in altre posizioni. Se il problema è fortemente vincolato, la probabilità che l'incrocio di due genotipi associati a soluzioni ammissibili generi un individuo corrispondente a una soluzione non ammissibile sarà tutt'altro che trascurabile. Queste osservazioni suggeriscono chiaramente che una rappresentazione binaria di questo genere mal si adatta a problemi di questo tipo.

8.6. — Una tecnica efficace e concettualmente semplice per superare le difficoltà di rappresentazione per questa classe di problemi fortemente vincolati richiede la disponibilità di un algoritmo a bassa complessità computazionale (al massimo polinomiale di grado basso) e parametrizzabile, che calcoli un assegnamento ammissibile dei valori v_i e w_j [36], da utilizzare come funzione di decodifica M. Un semplice algoritmo *greedy* (cioè "ingordo", poiché cerca di costruire una soluzione compiendo sempre la scelta localmente migliore tra le alternative disponibili, anche quando questo può si-

gnificare precludersi in seguito scelte globalmente migliori), o simile, spesso soddisfa questi requisiti; ci si riduce quindi a un problema di ottimizzazione funzionale nello spazio "trasformato" dei parametri di questo algoritmo. In altre parole, invece di cercare l'assegnamento ottimale, si cerca di determinare i valori dei parametri che fanno sì che l'algoritmo *greedy* lo generi.

Ottimizzazione di permutazioni.

Molte potenziali applicazioni degli Algoritmi Evolutivi sono caratterizzate da complesse dipendenze di ordinamento, sul tipo di quelle che si incontrano nel Problema del Commesso Viaggiatore, nel Job-Shop Scheduling e simili. Generalizzando, si può affermare che una soluzione a un problema di questa classe è una permutazione, o una classe di equivalenza di permutazioni (come per esempio un circuito hamiltoniano in un grafo), di oggetti.

8.7. — Una permutazione di N oggetti potrebbe essere rappresentata dal suo indice, cioè un intero in $\{1, \ldots, N!\}$, a sua volta codificato in una stringa binaria. Sebbene questa sia senza dubbio la rappresentazione più compatta, essa ha il difetto che l'alterazione di un suo elemento (una cifra binaria) produce una nuova soluzione che non ha niente a che vedere con quella di partenza e lo stesso discorso vale per l'incrocio.

8.8. — Una rappresentazione più intuitiva (cfr. per esempio [36]) consiste nel codificare una permutazione di N elementi $e_1, \ldots, e_N$ come una lista o vettore di N interi in $\{1, \ldots, N\}$ interpretati come indici: così il vettore $(i_1, \ldots, i_N)$, con $1 \leq i_k \leq N$ e $i_j \neq i_k$ per ogni $k = 1, \ldots, N$, $j \neq k$, rappresenta la permutazione $(e_{i_1}, \ldots, e_{i_N})$. Chiaramente una rappresentazione siffatta richiede la ridefinizione degli operatori genetici in modo da assicularne la "chiusura": infatti, alterando casualmente uno degli N interi i_k, si otterrà un nuovo vettoreche non rappresenta più una permutazione, essendo un elemento ripetuto e un altro assente. Un'altra soluzione consiste nell'usare una funzione di penalizzazione implicita, come in [91].

8.9. — Un'ulteriore possibilità, che per giunta garantisce la chiusura rispetto a qualsiasi operatore, consiste nel rappresentare una permutazione associando a ciascuno degli N elementi un peso reale $w_i \in \mathbb{R}$, $i = 1, \ldots, N$, in modo tale che sia $w_i < w_j$ se e solo se e_i precede e_j nella permutazione. La funzione di decodifica M dovrà quindi semplicemente ordinare N numeri per ottenere la permutazione corrispondente, compito che ha complessità computazionele $O(N \log N)$ [33].

Una quarta alternativa, proposta in [67] per il Problema del Commesso Viaggiatore, utilizza la matrice binaria di adiacenza per rappresentare direttamente gli archi di un circuito, il che consente l'utilizzo di operatori genetici solo leggermente modificati rispetto a quelli tradizionali per garantire la chiusura.

Operatori avanzati.

Nei paragrafi 4–7 sono già stati menzionati e sommariamente illustrati alcuni schemi di operatori genetici. Questa sezione fornisce un breve inventario di questi ed altri schemi, soprattutto di ricombinazione, con cenni sulle motivazioni che hanno suggerito la loro introduzione. Gli operatori genetici si possono dividere in generali e specializzati, a seconda che siano, almeno in linea di principio, indipendenti dal problema oggetto ovvero siano definiti per una classe specifica di problemi e facciano uso di concetti ed euristiche tipiche di quella classe.

8.10. — Il più semplice modello di incrocio è quello "a un punto" (cfr. §6.6). Nel contesto più generale degli Algoritmi Evolutivi, l'incrocio a un punto consiste nello spezzare in due parti la struttura dati dei genotipi dei genitori e nel formare uno o due individui figli prendendo una parte della loro struttura dati da un genitore e una parte dall'altro. Questo meccanismo può essere esteso spezzando la struttura dati in un numero arbitrario di parti. Quando ogni singolo gene può provenire, con probabilità uniforme e indipendente, dall'uno o dall'altro genitore, si ha il cosiddetto incrocio uniforme. Per esempio, la ricombinazione descreta usata nelle Strategie Evolutive (cfr. §4.7) è un incrocio uniforme su vettori di numeri reali.

8.11. — Incrocio ordinato uniforme L'incrocio ordinato uniforme (*Uniform Order-Based Crossover*), introdotto da Lawrence Davis in [35], si basa su un'idea semplice ed efficace: scegliere a caso un insieme di posizioni da uno dei due individui genitori e riordinare gli altri elementi secondo l'ordine in cui gli elementi corrispondenti compaiono nell'altro. Siano $\gamma = (\gamma_1, \ldots, \gamma_l)$ e $\kappa = (\kappa_1, \ldots, \kappa_l)$ i genotipi dei due individui genitori. Il genotipo $\chi = (\chi_1, \ldots, \chi_l)$ di un individuo figlio è generato come segue:

1. sia $\mathbf{b} = (b_1, \ldots, b_l)$ un vettore casuale di l cifre binarie e sia $k = \sum_{i=0}^{l} b_i$;
2. $\chi_i := \gamma_i$ per ogni $1 \leq i \leq l$ tale che $b_i = 1$;
3. sia ℓ la lista degli elementi γ_i tali che $b_i = 0$;
4. sia ℓ' la permutazione di ℓ tale che gli stessi elementi vi compaiano nello stesso ordine in cui compaiono in κ;
5. si riempiano le posizioni χ_i tali che $b_i = 0$ utilizzando in ordine gli elementi di ℓ'.

8.12. — Incrocio parzialmente mappato L'incrocio parzialmente mappato per problemi di ottimizzazione di circuiti su grafi (*Partially Mapped Crossover,* PMX), introdotto in [56], cerca di preservare l'informazione posizionale presente nei due individui genitori. Due punti di incrocio vengono scelti a caso e gli elementi di uno dei genitori tra questi compresi vengono ereditati, senza modifiche, dall'individuo figlio; gli elementi nella stessa posizione dell'altro genitore vengono mappati nella posizione occupata dall'elemento corrispondente nel primo genitore e gli elementi restanti vengono passati al figlio. Siano $\gamma = (\gamma_1, \ldots, \gamma_l)$ e $\kappa = (\kappa_1, \ldots, \kappa_l)$ i genotipi dei due individui genitori. Il genotipo $\chi = (\chi_1, \ldots, \chi_l)$ di un individuo figlio è generato come segue:

1. siano $1 \leq p \leq q \leq l$ due posizioni generate a caso con probabilità uniforme su $\{1, \ldots, l\}$;
2. $i := p$

4. se $i = q$ si vada al punto 9;
5. $\chi_i = \gamma_i$;
6. sia j tale che $\kappa_j = \gamma_i$;
7. se $j < p$ o $j > q$, $\chi_j := \kappa_i$;
8. $i := i + 1$, si torni al punto 4;
9. si riempiano le posizioni di χ rimaste vacanti con gli elementi che occupano le posizioni corrispondenti in κ;
10. a questo punto ci possono essere alcuni elementi ripetuti in χ ed altri mancanti: si sostituisca ogni doppione χ_i con $i < p$ o $i > q$ con uno degli elementi mancanti, ottenendo così un circuito hamiltoniano.

8.13. — Ricombinazione su archi Un operatore di ricombinazione su archi per problemi di ottimizzazione di circuiti su grafi descritto in [146] è il seguente. Sia $G = (V, E)$ un grafo con N vertici e siano γ e $\kappa \in \Gamma$ due circuiti hamiltoniani su G; un circuito figlio $\chi \in \Gamma$ viene prodotto come segue:

1. $\chi := \emptyset$; per ogni $v \in V$, sia $u \in A(v)$ se e solo se $(u, v) \in \gamma$ o $(u, v) \in \kappa$;
2. $s :=$ un vertice qualsiasi di γ o κ;
3. $d := \arg\min_{v \in A(s)} \|A(v)\|$;
4. $\chi := \chi \cup (s, d)$;
5. $A(s) := A(s) \setminus \{d\}$, $A(d) := A(d) \setminus \{s\}$;
6. s := d;
7. se $A(s) \neq \emptyset$, si torni al punto 3.

8.14. — Altri operatori Vale infine la pena di menzionare il fatto che spesso gli Algoritmi Evolutivi più avanzati adottano diversi operatori addizionali "ad hoc", costituiti tipicamente da semplici euristiche del problema (per esempio il k-scambio per il Problema del Commesso Viaggiatore) o da algoritmi cosiddetti di *hill-climbing*, cioè di ottimizzazione locale. Negli algoritmi evolutivi che fanno uso di una rappresentazione ad albero sintattico, come la Programmazione Genetica, un operatore notevole è la semplificazione (si veda §7.4).

Un altro "operatore" può essere considerato il metodo di generazione della popolazione iniziale. Diversi autori (cfr. per esempio [36]) suggeriscono di utilizzare algoritmi classici per generare un certo numero di soluzioni già di buon livello da immettere nella popolazione iniziale.

Algoritmi Evolutivi distribuiti e paralleli.

8.15. — Una popolazione in cui ciascun individuo, ad ogni generazione, può incrociarsi con qualunque altro in essa presente è chiamata *panmittica,* vale a dire completamente mescolata. Questo è certamente il caso in tutti gli algoritmi evolutivi classici illustrati nei paragrafi precedenti.

Essendo gli Algoritmi Evolutivi in generale impegnativi dal punto di vista computazionale e, nello stesso tempo, facilmente parallelizzabili, si è cominciato abbastanza

presto ad implementarli su reti di elaboratori o su elaboratori paralleli [59, 30, 117, 107].

8.16. — Poiché la comunicazione tra due elaboratori o tra processori diversi dello stesso elaboratore parallelo è in genere costosa in termini computazionali, si pone il problema di imporre una struttura *apomittica* (cioè che si discosta dal mescolamento completo) alla popolazione, cioè di scomporla in un certo numero di *demi,* o sottopopolazioni, collegati tra di loro in qualche modo. Esistono tre modelli apomittici già investigati dai genetisti (una buona rassegna sui vari modelli di popolazioni si trova in [43]):

1. *a isole,* in cui la migrazione può avvenire tra qualunque coppia di demi;
2. *stepping-stone,* in cui la migrazione può avvenire solo tra demi adiacenti;
3. *continuo,* in cui la selezione e l'incrocio sono locali a un intorno di raggio fissato e l'isolamento tra i demi avviene per la distanza.

8.17. — L'ultimo di questi modelli dà luogo ad algoritmi evolutivi cosiddetti "a grana fine" [98], in cui tipicamente gli individui sono disposti su una griglia planare, con intorno fissato da un parametro. Ad ogni generazione vengono eseguiti i seguenti passi in parallelo:

1. sostituire ogni individuo con uno selezionato nel suo intorno;
2. incrociare ogni individuo con uno scelto a caso nel suo intorno;
3. applicare la mutazione con probabilità assegnata.

I modelli a isole e stepping-stone si adattano a un tipo di algoritmo evolutivo distribuito [137], come per esempio quello investigato in [117], che mantiene un certo numero di popolazioni comunicanti, disposte in una struttura a ipercubo.

§ 9. — Modelli di selezione.

Esplorazione e sfruttamento — selezione proporzionale all'idoneità — selezione per ordinamento lineare — selezione a torneo locale — selezione per troncamento — classificazione.

Esplorazione e sfruttamento.

9.1. — Lo scopo della selezione negli Algoritmi Evolutivi è concentrare l'utilizzo delle risorse computazionali disponibili in regioni promettenti dello spazio di ricerca.

Esiste un rapporto di reciprocità tra i due aspetti di *esplorazione* e *sfruttamento* dello spazio di ricerca e, chiaramente, più forte è la pressione esercitata dalla selezione nella direzione della concentrazione dello sforzo computazionale, minore è la frazione di risorse utilizzata per esplorare altre possibilità; a un estremo, al diminuire della pressione selettiva e, di conseguenza, all'aumentare delle risorse utilizzate per

l'esplorazione, l'Algoritmo Evolutivo tende a comportarsi come un metodo Monte Carlo crudo, campionando casualmente lo spazio delle soluzioni ammissibili; all'altro estremo, all'aumentare della pressione selettiva, l'Algoritmo Evolutivo degenera in un metodo di ricerca locale a "discesa del gradiente".

9.2. — Questi due estremi hanno una precisa corrispondenza con due tipologie di funzione obiettivo: da una parte il metodo del gradiente assicura la convergenza (rapida) all'ottimo globale per funzioni obiettivo unimodali; dall'altra, l'unico algoritmo che assicura la convergenza quasi certa all'ottimo globale per funzioni obiettivo fortemente irregolari (quasi ovunque discontinue, multimodali, affette da rumore, ecc.) è, a parte la ricerca esaustiva, che comunque non è fattibile in pratica per quasi tutti i problemi, il campionamento casuale.

Gli Algoritmi Evolutivi possono quindi essere visti come un compromesso tra questi due estremi e la selezione è lo strumento che lo regola: una pressione selettiva troppo elevata può risultare nella convergenza a un ottimo locale, mentre il contrario può rallentare inutilmente la convergenza all'ottimo globale.

Selezione proporzionale all'idoneità.

9.3. — Il modello di selezione proporzionale all'idoneità fu ricavato da Holland come quello che ottimizza il compromesso tra esplorazione e sfruttamento, utilizzando il modello astratto della slot-machine a più leve (cfr. [65], Cap. V).

Siano $\xi_1, \ldots, \xi_k$ k variabili aleatorie reali, i cui valori rappresentano possibili ricompense ottenute giocando con la leva corrispondente della slot-machine, ciascuna con una sua distribuzione di probabilità nota al giocatore solo attraverso una stima π_i del valore atteso μ_i, con $i = 1, \ldots, k$, ottenuta osservando un certo numero di giocate precedenti.

Il giocatore ha a disposizione in tutto N tentativi e deve decidere come distribuirli tra le k leve in modo tale da massimizzare il valore atteso del proprio capitale, cioè della somma delle ricompense ricevute,

$$C = \sum_{i=1}^{k} n_i \xi_i,$$

dove $\mathbf{n} = (n_1, \ldots, n_k)$ è un assegnamento del numero di tentativi allocati a ciascuna leva, tale che $\sum_{i=1}^{k} n_i = N$.

In altri termini, si cerca l'assegnamento che massimizzi il valore atteso

$$E[C] = \sum_{i=1}^{k} n_i \mu_i.$$

Si dimostra che, date queste ipotesi, l'assegnamento ottimo $\mathbf{n}^*$ è quello per cui, per ogni $i \in \{1, \ldots, k\}$,

$$n_i^* = \frac{\pi_i N}{\sum_{j=1}^{k} \pi_j},$$

che corrisponde a un'allocazione dei tentativi proporzionale alle stime delle ricompense attese.

9.4. — Pur essendo stato ricavato in maniera rigorosa ed avendo una giustificazione profonda radicata nella Teoria delle Decisioni, il modello di selezione proporzionale all'idoneità presenta degli inconvenienti.

Per esempio, si considerino due funzioni di idoneità $f_1(\gamma)$ e $f_2(\gamma) = f_1(\gamma) + c$, per ogni $\gamma \in \Gamma$. Dal momento che esse appaiono sostanzialmente equivalenti, ci si aspetterebbe che il comportamento di un Algoritmo Evolutivo non cambi sostituendo l'una all'altra; tuttavia, se il modello di selezione è quello proporzionale all'idoneità ciò è ovviamente falso.

Un altro problema della selezione proporzionale all'idoneità è costituito dai cosiddetti *superindividui*. Un superindividuo in una popolazione x è un individuo il cui genotipo $\gamma \in \Gamma$ è tale che $f(\gamma) \gg f(x)$. Il genotipo di un superindividuo si vede allocare dalla selezione proporzionale all'idoneità una fetta consistente di istanze nella generazione successiva e, nel giro di poche generazioni, sommerge qualsiasi altro genotipo inizialmente presente nella popolazione, causando una situazione di convergenza. Se il superindividuo è associato all'ottimo globale del problema oggetto, questo è esattamente ciò che si desidera, ma se, d'altronde, esso corrisponde a un ottimo locale, questo determina un fallimento dell'algoritmo.

La lista dei problemi non si esaurisce qui. La spinta verso il miglioramento fornita dalla selezione proporzionale all'idoneità tende asintoticamente a zero a mano a mano che gli individui di una popolazione si avvicinano collettivamente all'ottimo. Questo effettivamente rispecchia ciò che si osserva in Natura, ma nell'ambito dell'ottimizzazione rappresenta un indubbio inconveniente.

Per affrontare e risolvere queste difficoltà sono possibili due approcci: uno consiste nel modificare opportunamente la funzione di idoneità (cfr. §6.4); l'altro ricorre più semplicemente alla definizione di modelli di selezione alternativi. Uno di questi modelli, proposto dall'autore, costituisce l'oggetto del Capitolo III di questa Tesi.

Selezione per ordinamento lineare.

9.5. — La selezione per ordinamento lineare [8] si basa sull'ordinamento degli individui presenti nella popolazione per valori decrescenti dell'idoneità. Si definisce perciò la funzione $\mathrm{rank}\colon \Gamma \to \{1, \ldots, n\}$, dove n è la cardinalità della popolazione, che assegna ad ogni genotipo presente nella popolazione una posizione in classifica. La probabilità che l'i-esimo individuo in classifica venga estratto è quindi definita, per $i = 1, \ldots, n$, come

$$p(i) = \frac{1}{n}\left[\beta - 2(\beta - 1)\frac{i-1}{n-1}\right],$$

dove $0 \leq \beta \leq 2$ è un parametro che può essere interpretato come il tasso atteso di campionamento dell'individuo migliore in n estrazioni indipendenti con reimbussolamento.

Considerando che individui con la stessa idoneità occuperanno posizioni contigue in classifica, la probabilità $P_x[\gamma]$ che un individuo avente genotipo γ venga estratto

dalla popolazione x può essere calcolato come segue:

$$P_x[\gamma] = q_x(\gamma)\frac{1}{n_x(\Gamma_{f(\gamma)})}\sum_{i=0}^{n_x(\Gamma_{f(\gamma)})-1}\left[\beta - 2(\beta-1)\frac{\mathrm{rank}(\gamma)+i+1}{n-1}\right] =$$
$$= q_x(\gamma)\left[\beta - 2\frac{\beta-1}{n-1}\left(\mathrm{rank}(\gamma) + \frac{n_x(\Gamma_{f(\gamma)})-1}{2} - 1\right)\right].$$

Per valori di $\beta \geq 2$, gli ultimi $n - \frac{(n-1)\beta}{2(\beta-1)}$ individui in classifica vengono esclusi dal processo selettivo; per $\beta = 2$ solo l'individuo peggiore non viene preso in considerazione per la selezione.

Selezione a torneo locale.

9.6. — La selezione a torneo locale, attribuita a un lavoro (mai pubblicato) di Wetzel e studiata per la prima volta in [22], estrae k individui dalla popolazione con probabilità uniforme ma *senza* reimbussolamento e fa loro disputare un "torneo", che viene vinto, nel caso deterministico, dall'individuo più idoneo tra i partecipanti. Il torneo può anche essere probabilistico, nel qual caso la probabilità che un individuo lo vinca è in genere proporzionale alla sua idoneità.

La pressione selettiva è direttamente proporzionale al numero k dei partecipanti al torneo: a un estremo, per $k = n$, la cardinalità della popolazione, la selezione a torneo locale deterministico degenera in una selezione per troncamento con parametro $\tau = \frac{1}{n}$ (si veda sotto) e la selezione a torneo locale probabilistico degenera nella selezione proporzionale all'idoneità.

Selezione per troncamento.

9.7. — La selezione per troncamento [108], introdotta da Heinz Mühlenbein e Dirk Schlierkamp-Voosen con il loro Algoritmo Genetico Allevatore (*Breeder Genetic Algorithm*), trova la sua fonte di ispirazione nella zootecnia, una branca della statistica applicata, e i concetti principali sui quali poggia sono la correlazione tra genitore e figlio e il coefficiente di ereditarietà.

Come osservano Mühlenbein e Schlierkamp-Voosen, non ci sono grandi differenze di principio tra l'allevamento di organismi naturali e quello di soluzioni a un problema oggetto. Una differenza secondaria è che nel secondo caso è possibile esercitare un controllo sugli operatori genetici di incrocio e mutazione e modificarli in modo da trarne il massimo vantaggio.

9.8. — L'effetto della selezione che più interessa all'allevatore è la *risposta alla selezione* R, definita come la differenza tra le idoneità medie della popolazione in due generazioni successive:

$$R_t = f(x_{t+1}) - f(x_t).$$

Gli allevatori misurano la selezione per mezzo del *differenziale selettivo* S, definito come la differenza tra l'idoneità media dei capi selezionati per la riproduzione e quella della popolazione nel suo complesso:

$$S_t = f(\bar{x}_t) - f(x_t), \quad \bar{x}_t \sqsubseteq x_t.$$

La selezione per troncamento consiste nel selezionare per la riproduzione solo gli individui migliori e scartare gli altri. Il differenziale selettivo dipende dalla proporzione τ della popolazione che è selezionata: più piccolo è τ, maggiore sarà S_t. Per formalizzare il modello, si supponga di ordinare per valori di idoneità decrescenti $f_1 > f_2 > \ldots > f_n$ gli individui della popolazione; i primi τn individui sono quelli selezionati per la riproduzione. Il differenziale selettivo è allora dato da

$$S_t = \sum_{i=1}^{\tau n} \frac{f_i}{\tau n} - f(x_t).$$

Assumendo che valori di idoneità abbiano distribuzione normale con media $f(x_t)$ e varianza σ_t^2 nella popolazione x_t, è conveniente definire l'*intensità selettiva* come la quantità standardizzata $I = S_t/\sigma_t$; il valore atteso di questa quantità è dato da (cfr. [23], Cap. IX)

$$E[I] = \frac{1}{\tau n} \sum_{i=1}^{\tau n} E\left[\frac{f_i - f(x_t)}{\sigma_t}\right].$$

Per valori sufficientemente grandi di n, la distribuzione di I approssima una distribuzione normale troncata in corrispondenza di z_t tale che $P[f(\gamma) \geq z_t | \gamma \in x_t] = \tau$; è possibile perciò approssimare la funzione di densità di I con $\phi(t)/\tau$, dove

$$\phi(t) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}$$

ottenendo così un'approssimazione per il suo valore atteso:

$$E[I] \approx \frac{1}{\tau} \int_{z_t}^{\infty} t\phi(t)dt = \frac{\phi(z_t)}{\tau}.$$

La risposta alla selezione è

$$R_t = b_t S_t,$$

dove il coefficiente b_t, che si suppone costante per un certo numero di generazioni, è chiamato *ereditarietà realizzata*.

È chiaro che, qualora si disponga di una stima di b_t e σ_t^2, è possibile dare una stima della velocità di convergenza attesa dell'algoritmo.

Classificazione dei modelli di selezione.

9.9. — Uno schema di classificazione per i modelli di selezione è stato delineato da Thomas Bäck e Frank Hoffmeister in [6]. Seguendo la loro tassonomia, un modello di selezione può essere classificato rispetto ad almeno quattro assi ortogonali:

1. *dinamico—statico,* a seconda che le probabilità di selezione dipendano o meno dai valori di idoneità effettivamente presenti nella popolazione e quindi varino o meno da una generazione all'altra: la selezione proporzionale all'idoneità è quindi statica, mentre quelle per ordinamento lineare, torneo locale e troncamento sono dinamiche;
2. *preservativo—estintivo,* a seconda che garantisca o meno ad ogni individuo una probabilità non nulla di essere selezionato: così la selezione per troncamento e per torneo locale deterministico sono estintive, la selezione proporzionale all'idoneità è preservativa e la selezione per ordinamento lineare è preservativa per $\beta < 2$ ed estintiva per $\beta \geq 2$;
3. *elitistico—puro,* a seconda che garantisca o meno il passaggio dell'individuo più idoneo immutato alla generazione successiva;
4. *generazionale—a stato stazionario,* a seconda che l'insieme degli individui genitori sia determinato in un colpo solo e rimanga fissato fintantoché una nuova popolazione di individui figli non sia stata prodotta, oppure gli individui genitori siano estratti in momenti successivi e gli individui figli siano a mano a mano introdotti nella popolazione.

§ 10. — Applicazioni ai problemi di ottimizzazione.

Applicazione a problemi NP-completi — il Commesso Viaggiatore — problemi di scheduling e routing — problemi su grafi — altri problemi — Intelligenza Artificiale — altre applicazioni.

Da quando furono introdotti ad oggi gli Algoritmi Evolutivi sono stati applicati alla risoluzione di vari problemi di ottimizzazione, spesso incontrati in situazioni d'interesse pratico. Molte di queste applicazioni sono risultate in un successo, vuoi per la qualità delle soluzioni ottenute, vuoi per lo sforzo d'implementazione relativamente ridotto richiesto per raggiungere prestazioni accettabili, contribuendo così ad alimentare l'interesse per questa classe di metodi.

La letteratura è ricca di esempi: quelli forniti qui di seguito intendono suggerire un quadro indicativo dello stato della ricerca sull'applicazione degli Algoritmi Evolutivi a diverse classi di problemi, piuttosto che fornire un elenco completo.

Applicazione a problemi NP-completi.

10.1. — In [39], Kenneth A. DeJong e William M. Spears osservano che, sebbene tutti i problemi NP-completi siano ugualmente difficili dal punto di vista computazionale, alcuni tra di essi ammettono delle rappresentazioni più facili da trattare da parte degli Algoritmi Evolutivi. Dato che per definizione [51] ogni problema NP-completo può essere mappato in ogni altro in tempo polinomiale, una strategia generale per affrontare problemi NP-completi per mezzo di Algoritmi Evolutivi può consistere nell'identificare un problema per così dire "canonico," sul quale gli Algoritmi Evolutivi funzionino

particolarmente bene e nel risolvere gli altri problemi indirettamente riconducendoli al problema canonico. DeJong e Spears propongono il problema della Soddisfacibilità booleana (SAT) come problema canonico e supportano questa loro proposta con alcuni risultati empirici.

Tuttavia non mancano in letteratura esempi di applicazioni specifiche a vari problemi NP-completi, spesso di interesse pratico.

10.2. — Il Commesso Viaggiatore Un primo algoritmo evolutivo per il Problema del Commesso Viaggiatore fu proposto da Mühlenbein e colleghi in [107]. In [146], Darrell Whitley, Timothy Starkweather e D'Ann Fuquay propongono un operatore di ricombinazione d'archi specifico per problemi di sequenziazione. Heinrich Braun descrive in [20] un'applicazione che migliora i risultati di Mühlenbein e colleghi.

Un approccio diverso, che si basa su algoritmi classici di ricerca locale è presentato in [143]; sulla stessa falsa riga è l'approccio proposto da Valenzuela e Jones in [145], che utilizzano una tecnica *divide et impera* in cui l'algoritmo genetico esplora lo spazio delle suddivisioni del problema, piuttosto che il problema stesso, e applica poi una versione adattata dell'algoritmo di Karp con le suddivisioni trovate.

Hans-Georg Beyer [16] descrive un'applicazione agli studi di progettazione di un Linear Collider per la Fisica delle alte energie; [67] introduce una rappresentazione basata sulla matrice di adiacenza del grafo delle città, che consente all'algoritmo genetico di manipolare gli archi utilizzando l'incrocio convenzionale: questa rappresentazione, combinata con l'operatore di inversione (2-opt, che è una ben nota euristica per il Problema del Commesso Viaggiatore), fa sì che l'algoritmo trovi rapidamente l'ottimo di sei su otto problemi di test ampiamente studiati in letteratura e commetta un errore entro il 2% negli altri due.

10.3. — Job-Shop Scheduling e simili Questa famiglia di problemi è strettamente imparentata con il Problema del Commesso Viaggiatore. Gary A. Cleveland e Stephen F. Smith, in [29], studiano il problema del flow shop; Philip Husbands, Frank Mill e Stephen Warrington descrivono in [72] un approccio parallelo, basato sulla coevoluzione di specie diverse, ai problemi di ottimizzazione dei piani di produzione e di scheduling.

Per quanto riguarda l'applicazione a problemi di Job-Shop Scheduling di grande dimensioni, esistono due lavori di Ryohei Nakano e Takeshi Yamada della NTT, [111] e [150]: il primo studia l'utilizzo di Algoritmi Genetici classici, cioè con codifica binaria. Il problema, con questo tipo di codifica, è che difficilmente un genotipo ottenuto per incrocio o mutazione rappresenta una soluzione ammissibile; questo problema viene risolto per mezzo di un apposito operatore di *riparazione*, che mappa una soluzione inammissibile nella più vicina soluzione ammissibile. La qualità delle soluzioni ottenute si avvicina ai metodi di *branch and bound*. Il secondo lavoro ricorre ad una rappresentazione simbolica e sfrutta un operatore di incrocio ad hoc basato sull'Algoritmo di Giffler e Thompson.

10.4. — Scheduling, Routing, ecc. Gilbert Syswerda e Jeff Palmucci riferiscono in dettaglio la costruzione di un ottimizzatore per lo scheduling di risorse che sfrutta una combinazione di ricerca locale con metodi convenzionali e globale con

un algoritmo genetico [136].

Alberto Colorni, Marco Dorigo e Vittorio Maniezzo, del Politecnico di Milano, descrivono in [32] una loro esperienza in un problema di scheduling di grande importanza pratica, il problema dell'orario, maturata sviluppando un'applicazione per la costruzione degli orari di un Istituto Tecnico italiano. Si-Eng Ling descrive in [94] un sistema per il Problema dell'Orario che fa uso di un programma di assegnamento, scritto in PROLOG, come componente che fornisce soluzioni parziali, che poi spetta all'Algoritmo Genetico completare.

Un'applicazione d'interesse pratico, sviluppata per la CSX Transportation, una compagnia statunitense di trasporto merci su rotaia, è descritta in [50].

Studiando un'applicazione a un problema di routing di più veicoli con finestre temporali e vincoli di capacità, Joe Blanton Jr. e Roger Wainwright introducono in [73] due nuovi operatori di incrocio per problemi fortemente vincolati. Un problema analogo, in cui addiritura il compito di trovare una soluzione ammissibile, ancorché ottima, è NP-completo, è affrontato in [139], dove vengono confrontate le prestazioni di un algoritmo genetico e un algoritmo basato su un'euristica *greedy* e si osserva che i due approcci sono complementari, nel senso che uno funziona meglio per quelle istanze in cui l'altro si trova in difficoltà.

Un problema che ricade nella categoria trattata in questo paragrafo è quello del carico e della sistemazione di bancali affrontato da Kate Juliff in [79] con un algoritmo che rappresenta le soluzioni per mezzo di tre cromosomi distinti, uno per ciascuna delle caratteristiche di una soluzione, un accorgimento adottato anche da Philip Husbands in [71] per un problema di Job Shop Scheduling.

10.5. — Alberi di Steiner In [64] le prestazioni di un Algoritmo Genetico per la costruzione di alberi di Steiner vengono confrontate con quelle di un algoritmo di Annealing Simulato e uno dei migliori algoritmi convenzionali, proposto da Rayward-Smith e Clare [123], e i tre algoritmi si dimostrano equivalenti nel numero di valutazioni richieste per trovale l'ottimo.

Bryant Julstrom affronta in [80] il problema di costruire alberi di Steiner rettilinei, i cui archi sono cioè orientati esclusivamente lungo i due assi del piano.

Nello studio dell'evoluzione biologica si ricostruiscono alberi filogenetici a partire da dati (come sequenze molecolari) di insiemi di organismi attuali. Ogni foglia dell'albero rappresenta i dati di un organismo. Spesso un criterio per la ricostruzione è la minimizzazione del numero di mutazioni lungo i rami dell'albero: un albero siffatto è chiamato "albero a mutazioni minime" ed è l'equivalente multidimensionale di un albero di Stainer. Un'applicazione degli Algoritmi Evolutivi a questo problema è descritta in [52].

10.6. — Altri problemi su grafi Quello della partizione di grafi o reti è un problema complesso, discreto, fortemente non lineare con importanti applicazioni pratiche per esempio nella progettazione di circuiti elettronici: Martin Hulin, della Siemens, suggerisce in [69] l'utilizzo di una rappresentazione che preserva la struttura dei circuiti, risultando in genotipi diploidi, cioè composti di due cromosomi distinti. Donald Jones e Mark Beltramo, dei Laboratori di Ricerca della General Motors, suggeriscono, dopo un confronto di varie alternative [76], di rappresentare una partizione

per mezzo di una permutazione, come già proposto da Bhuyan, Raghavan e Elayavalli [17] per la partizione di insiemi (cfr. §10.8), e di decodificarla con un'euristica additiva *greedy.* Heinz Mühlenbein e Gregor von Laszewski affrontano, in [109], lo stesso problema per mezzo di un algoritmo parallelo con selezione locale e operatori genetici sofisticati.

Anche il problema del Channel Routing ha importanti applicazioni nella progettazione di circuiti elettronici: Rahmani e Ono [122] lo affrontano incorporando della conoscenza specifica del problema in uno speciale operatore di mutazione; Jens Lienig e K. Thulasiraman invece, in [92], adottano una rappresentazione e operatori genetici *ad hoc,* ottenendo risultati qualitativamente simili o superiori ai migliori pubblicati.

Il problema più generale della sintesi di circuiti integrati a partire da una descrizione ad alto livello del comportamento desiderato per mezzo di Algoritmi Evolutivi è studiato in [102].

10.7. — Il Problema dello Zaino e simili Un'algoritmo parallelo per l'impaccamento di rettangoli, cioè una versione bidimensionale del Problema del Bin Packing è presentato in [85].

10.8. — Partizione di Insiemi Bhuyan, Raghavan e Elayavalli, in [17], affrontano il problema di trovare una partizione in M sottoinsiemi non banali e disgiunti di un insieme N oggetti. I risultati sperimentali indicano che le prestazioni dell'algoritmo proposto si pongono a metà strada tra i migliori algoritmi euristici conosciuti e le tecniche tradizionali basate su algoritmi di branch and bound.

Lo stesso problema è studiato da David M. Levine con un algoritmo evolutivo combinato con un'euristica di ricerca locale [89].

Intelligenza Artificiale.

10.9. — Pianificazione Wilfried Jakob, Martina Gorges-Schleuter e Christian Blume [74] applicano GLEAM, un algoritmo genetico specializzato per l'apprendimento "intuitivo" a due problemi di pianificazione di compiti e apprendimento per un robot industriale.

10.10. — Induzione In [83], John Koza applica la Programmazione Genetica all'apprendimento di alberi di decisione. Peter Wyard presenta in [149] un'originale applicazione all'induzione di grammatiche libere dal contesto di linguaggi con parentesi annidate e bilanciate correttamente.

10.11. — Reti Neuronali Gli Algoritmi Evolutivi hanno trovato applicazione nella sintesi ed istruzione di Reti Neuronali, come naturale alternativa agli algoritmi di "Backpropagation", che si possono caratterizzare come algoritmi di massima discesa. A differenza di questi, gli Algoritmi Evolutivi difficilmente rimangono intrappolati in ottimi locali. Applicazioni in questo senso sono descritte in [5] e [62].

Strettamanete imparentate con le Reti Neuronali sono le Mappe di Kohonen; [118]

descrive l'applicazione di un algoritmo genetico a questo tipo di rete.

Altre applicazioni.

10.12. — Analisi Numerica Gli Algoritmi Evolutivi hanno trovato applicazione anche in alcuni settori dell'Analisi Numerica, come la sintesi di funzioni spline: per esempio [100] investiga l'uso di un algoritmo genetico come parte di un algoritmo ibrido per la regressione di dati sperimentali affetti da rumore con funzioni spline.

10.13. — Ricerca Operativa Il problema del Massimo Flusso nelle reti è affrontato in [110] rappresentando le soluzioni per mezzo di matrici di flusso e definendo la funzione di idoneità sulla base di due criteri, il bilanciamento dei flussi ai vertici e il tasso di saturazione della rete.

10.14. — Progettazione ingegneristica L'applicazione di algoritmi genetici all'ottimizzazione di progetti ingegneristici caratterizzati dalla presenza di vincoli non lineari è stata studiata da David J. Powell, Siu Shing Tong e Michael M. Skolnick in [120] e poi in [119]. Un problema di progettazione aeronautica è affrontato da Mark F. Bramlette e Rod Cusic in [19].

10.15. — Identificazione di sistemi Molte applicazioni ingegneristiche, tra le quali il rilevamento e la diagnosi di guasti, il controllo automatico e la simulazione, richiedono modelli matematici di sistemi dinamici. Esistono due approcci fondamentali alla costruzione di tali modelli: uno, analitico, si basa sulle leggi della Fisica; l'altro, sperimentale, si basa sulla regressione dei dati osservati. Quest'ultimo approccio è noto come "Identificazione di Sistemi". Timothy Johnson e Philip Husbands vi applicano in [75] gli Algoritmi Genetici.

10.16. — Controllistica Alla sintesi del controllo ottimo per mezzo di Algoritmi Evolutivi sono stati dedicati numerosi sforzi applicativi, tra i quali vale la pena citare [115], [60], [140] e [70]. Uno studio comparativo degli Algoritmi Evolutivi per il controllo di processi è sviluppato in [114].

Un discorso a parte meritano i lavori che studiano gli Algoritmi Evolutivi per la sintesi o l'ottimizzazione di sistemi di controllo basati sulla Logica Sfumata (cfr. Cap. IV, §4), che consistono essenzialmente in insiemi di regole in cui i valori di verità di antecedenti e conseguenti possono variare con continuità tra 0 (Falso) e 1 (Vero). Questo tipo di controllori ha di recente trovato un'applicazione sempre crescente nel campo dei prodotti di consumo, come automobili, macchine fotografiche, telecamere ed elettrodomestici "intelligenti". Un articolo che in un certo senso getta le basi dell'approccio evolutivo è quello di Charles Karr dello U. S. Bureau of Mines, [81]; altri lavori in questo senso sono quello di Philip Thrift della Texas Instruments [142] e due articoli di Michael Lee e Hideyuki Takagi [87, 88].

10.17. — Trattamento delle immagini Gli Algoritmi Evolutivi hanno

trovato applicazione anche nel campo del trattamento delle immagini. Un'applicazione alla Pattern Recognition è presentata in [134]; un approccio alternativo, descritto in [25], è quello di utilizzare gli Algoritmi Evolutivi per ottimizzare dei filtri per il riconoscimento di immagini.

Un'interessante applicazione alla generazione di identikit di criminali è stata sviluppata da Craig Caldwell e Victor Johnston del Dipartimento di Psicologia dell'Università Statale del Nuovo Messico [24], mentre un'applicazione al riconoscimento automatico di bersagli in campo militare è presentata in [9].

Un settore importante del Trattamento delle Immagini è costituito dalla Classificazione di Immagini o Clustering: [61] descrive un approccio alla classificazione basato sulla partizione delle caratteristiche o tratti di un'immagine per mezzo di Algoritmi Evolutivi; un altro lavoro, basato sulla selezione e classificazione di tratti, è descritto in [121].

10.18. — Elaborazione dei segnali L'applicazione degli Algoritmi Evolutivi all'Elaborazione dei Segnali si concretizza soprattutto nella sintesi ed ottimizzazione di filtri: si vedano per esempio i lavori di Chee-Hung Henry Chu [28], Schaffer ed Eshelman [129] e Flockton e White [44].

10.19. — Chimica e Biochimica Le applicazioni degli Algoritmi Evolutivi alla Chimica ed alla Biochimica vanno dall'analisi di molecole alla modellazione di reazioni.

Per l'analisi della conformazione di composti chimici (la cosiddetta "chemometrica") si veda [96], per l'analisi del DNA [97] e per l'analisi della struttura spaziale di proteine [130] e [144].

Un altro aspetto è costituito dall'esigenza di compiere ricerche su enormi basi di dati di strutture chimiche tridimensionali: un lavoro che affronta questo problema per mezzo degli Algoritmi Evolutivi è [77].

Nel campo della bioingegneria, lavorando su un'applicazione alla sintesi di proteine, Youvan e colleghi hanno sviluppato un nuovo membro della già numerosa famiglia degli Algoritmi Evolutivi: la "Mutagenesi Ricorsiva d'Insieme" (*Recursive Ensemble Mutagenesis*) [152]. Un'applicazione delle Strategie Evolutive alla determinazione degli equilibri di sistemi chimici reattivi è descritta in [125].

10.20. — Economia e Finanza Gli Algoritmi Evolutivi hanno destato l'interesse degli studiosi di Economia teorica: per esempio, Robert E. Marks [101] li utilizza per simulare e investigare il comportamento di agenti economici in un regime di oligopolio.

Gli Algoritmi Evolutivi sono anche stati applicati con successo a problemi finanziari; per esempio, un'applicazione descritta in [3] è oggi usata dal Gruppo I.M.I. per l'ottimizzazione dei portafogli di alcuni fondi d'investimento offerti al pubblico.

10.21. — Basi di Dati Gli Algoritmi Evolutivi hanno trovato applicazione anche nel campo della gestione delle Basi di Dati e dell'Information Retrieval, soprattutto per quanto riguarda l'ottimizzazione di query: si vedano per esempio [11] e [151].

Un'applicazione degli Algoritmi Evolutivi alla ricerca di basi di dati di strutture chimiche è descritta in [77].

Un approccio che combina gli Algoritmi Genetici con la Logica Sfumata è proposto in [128].

Capitolo II

ANALISI STOCASTICA

Nell'ambito dello studio dei fondamenti teorici degli Algoritmi Evolutivi, un argomento di grande importanza è rappresentato dall'analisi delle proprietà di convergenza del processo evolutivo che è alla base del loro funzionamento. Purtroppo questa analisi è molto complicata dall'interazione tra i vari operatori stocastici che lo definiscono ed esistono in letteratura relativamente pochi lavori che la affrontano, tra i quali si possono citare [57], [41], [93], [112] e [127], che modellano il processo evolutivo con le catene di Markov, e tutti i risultati hanno a che fare essenzialmente con la convergenza in probabilità. Sebbene questa sia una proprietà richiesta ad ogni metodo stocastico di ottimizzazione, la sua presenza da sola non dice granché sull'utilità pratica del metodo. Infatti la convergenza in probabilità all'ottimo vale addirittura per la ricerca casuale, che è il corrispettivo stocastico della ricerca esaustiva, il metodo di ottimizzazione in assoluto meno efficiente.

In primo luogo questo capitolo introduce alcuni concetti e risultati di Teoria delle Probabilità relativi a popolazioni e individui di un algoritmo evolutivo (§1), quindi si introduce un modello astratto di algoritmo evolutivo (§2), definendo i vari operatori stocastici che lo compongono; il modello viene caratterizzato come processo stocastico (§3) e vengono introdotte alcune sue statistiche d'interesse (§4). Viene infine dimostrata la convergenza in probabilità ad almeno una soluzione ottimale (§5). Dopo una rivisitazione del Teorema degli Schemi di Holland (§6), si collegano gli Algoritmi Evolutivi al concetto di martingala, dimostrandone perciò la convergenza in probabilità all'ottimo (§7). Il capitolo si conclude con alcuni cenni sul problema della velocità di convergenza, illustrando alcuni risultati preliminari (§8).

§ 1. — Misure su popolazioni e individui.

1.1. — Si può pensare a una popolazione $x \in \Gamma^{(n)}$ come a un campione di lunghezza n sullo spazio di probabilità $(\Gamma, 2^{\Gamma}, \mu)$, dove μ è una misura di probabilità su Γ.

Dato lo spazio probabilizzabile $(\Gamma^{(n)}, 2^{\Gamma^{(n)}})$, si consideri ora il problema di ricavare la misura di probabilità $\mu^{(n)}$ su $\Gamma^{(n)}$ definita come estensione della misura di probabilità μ e viceversa.

1.2. — Teorema Supposta nota $\mu(\gamma)$ per ogni $\gamma \in \Gamma$, per ogni popolazione

$x \in \Gamma^{(n)}$,

$$\mu^{(n)}(x) = n! \prod_{\gamma\in\Gamma} \frac{\mu(\gamma)^{n_x(\gamma)}}{n_x(\gamma)!}.$$

1.3. — Dimostrazione Come già detto, si può pensare a una popolazione $x \in \Gamma^{(n)}$ come la classe di equivalenza delle n-uple $\hat{\mathbf{x}} \in \Gamma^n$ di elementi di Γ identiche a meno di permutazioni; la probabilità di estrarre una particolare n-upla $\hat{\mathbf{x}}$ nella classe di equivalenza x è data da

$$\mu(\hat{\mathbf{x}}) = \mu(\hat{x}_1)\mu(\hat{x}_2)\cdots\mu(\hat{x}_n) = \prod_{\gamma\in\Gamma} \mu(\gamma)^{n_x(\gamma)}.$$

$\mu^{(n)}(x)$ è la probabilità di estrarre una qualsiasi delle n-uple $\hat{\mathbf{x}}$ nella classe di equivalenza x, quindi è data dalla somma delle singole probabilità di estrarre ciascuna n-upla $\hat{\mathbf{x}}$ nella classe di equivalenza x; poiché tutte queste probabilità sono identiche, $\mu^{(n)}(x)$ è il prodotto di $\mu(\hat{\mathbf{x}})$ per il numero di permutazioni distinte di $\hat{\mathbf{x}}$.

Se tutti gli n elementi di $\hat{\mathbf{x}}$ fossero distinti, le permutazioni di $\hat{\mathbf{x}}$ sarebbero banalmente $n!$. Invece ogni elemento $\gamma \in \Gamma$ compare in $\hat{\mathbf{x}}$ esattamente $n_x(\gamma)$ volte: queste $n_x(\gamma)$ copie di γ sono tra loro indistinguibili e così le loro $n_x(\gamma)!$ permutazioni. Iterando il ragionamento per ogni $\gamma \in \Gamma$ e ricordando che $0! = 1$, il numero di permutazioni distinte di $\hat{\mathbf{x}}$ è

$$\frac{n!}{\prod_{\gamma\in\Gamma} n_x(\gamma)!}.$$

Da ciò segue

$$\mu^{(n)}(x) = \mu(\hat{\mathbf{x}}) \frac{n!}{\prod_{\gamma\in\Gamma} n_x(\gamma)!} = n! \prod_{\gamma\in\Gamma} \frac{\mu(\gamma)^{n_x(\gamma)}}{n_x(\gamma)!},$$

come volevasi dimostrare. ∎

1.4. — In particolare, nel caso in cui μ è uniforme, $\mu(\gamma) = 1/r$ per ogni $\gamma \in \Gamma$ e, di conseguenza, si ottiene la distribuzione di Maxwell-Boltzmann

$$\mu^{(n)}(x) = \frac{n!}{r^n} \prod_{\gamma\in\Gamma} \frac{1}{n_x(\gamma)!}.$$

1.5. — Teorema Supposta nota $\mu^{(n)}(x)$ per ogni popolazione $x \in \Gamma^{(n)}$, per ogni individuo $\gamma \in \Gamma$,

$$\mu(\gamma) = \frac{1}{n} \sum_{x\in\Gamma^{(n)}} n_x(\gamma)\mu^{(n)}(x).$$

1.6. — Dimostrazione Una popolazione $x \in \Gamma^{(n)}$ è un evento elementare sullo spazio di probabilità $(\Gamma^{(n)}, 2^{\Gamma^{(n)}}, \mu^{(n)})$, ma può anche essere espresso (cfr. §1.3) come una disgiunzione di congiunzioni di eventi elementari sullo spazio di probabilità $(\Gamma, 2^\Gamma, \mu)$, quindi, in definitiva, come un elemento dell'algebra degli eventi 2^Γ. Questo

ci autorizza a scrivere la probabilità condizionata $\mu(\gamma|x)$, che esprime la probabilità di estrarre γ in una qualsiasi delle n estrazioni che danno il campione x.

Questa probabilità è data dalla frequenza di γ in x, cioè

$$\mu(\gamma|x) = q_x(\gamma) = \frac{n_x(\gamma)}{n}.$$

Allora, per il Teorema delle probabilità totali,

$$\mu(\gamma) = \sum_{x\in\Gamma^{(n)}} \mu(\gamma|x)\mu^{(n)}(x) = \sum_{x\in\Gamma^{(n)}} \frac{n_x(\gamma)}{n}\mu^{(n)}(x),$$

da cui la tesi. ∎

1.7. — In particolare, nel caso in cui sia

$$\mu^{(n)}(x) = \begin{cases} 1, & \text{se } x = \bar{x}; \\ 0, & \text{altrimenti;} \end{cases}$$

dove $\bar{x}$ è quella popolazione per cui $n_{\bar{x}}(\bar{\gamma}) = n$, cioè la popolazione in cui tutti gli individui hanno lo stesso identico genotipo $\bar{\gamma}$, si ha

$$\mu(\gamma) = \begin{cases} 1, & \text{se } \gamma = \bar{\gamma}; \\ 0, & \text{altrimenti.} \end{cases}$$

1.8. — Corollario Dai Teoremi 1.2 e 1.5 discendono le seguenti due uguaglianze:

$$\mu(\gamma) = (n-1)! \sum_{x\in\Gamma^{(n)}} n_x(\gamma) \prod_{\kappa\in\Gamma} \frac{\mu(\kappa)^{n_x(\kappa)}}{n_x(\kappa)!}$$

e

$$\mu^{(n)}(x) = n! \prod_{\gamma\in\Gamma} \frac{[\sum_{y\in\Gamma^{(n)}} n_y(\gamma)\mu^{(n)}(y)]^{n_x(\gamma)}}{n^{n_x(\gamma)} n_x(\gamma)!}$$

1.9. — Dimostrazione Per la prima uguaglianza si sostituisce il membro destro dell'equazione del Teorema 1.2 nell'equazione del Teorema 1.5; per la seconda uguaglianza si opera in maniera inversa, sostituendo il membro destro dell'equazione del Teorema 1.5 nell'equazione del Teorema 1.2. ∎

1.10. — Distribuzione dell'Idoneità La funzione di idoneità mappa genotipi in numeri reali positivi; nello stesso modo ogni misura di probabilità μ definita sullo spazio dei genotipi Γ si riflette in una corrispondente funzione di probabilità ϕ dell'idoneità di ogni singolo individuo in una popolazione.

La probabilità di un valore v di idoneità è la probabilità che un individuo γ estratto da Γ secondo la misura μ abbia idoneità $f(\gamma) = v$: cioè, posto $\Gamma_v \equiv \{\gamma \in \Gamma : f(\gamma) = v\}$, $\Gamma_v \subseteq \Gamma$,

$$\phi(v) = \mu(\Gamma_v).$$

Poiché la caratteristica di un individuo che ne determina la futura storia evolutiva è l'idoneità, un modo conveniente di osservare il processo evolutivo consiste nello studiare l'andamento della distribuzione ϕ: essendo definita su un dominio numerico, ϕ è inoltre facile da descrivere graficamente ed ha un valore atteso, una varianza e così via.

§ 2. — Un Algoritmo Evolutivo Astratto.

L'algoritmo — Gli operatori genetici — La funzione di transizione.

Definiamo una versione la più generale possibile di Algoritmo Evolutivo, di cui qualsiasi Algoritmo Evolutivo possa essere considerato una specializzazione. Un modello di questo tipo è presentato in [41], ma quello che segue, più in linea con la tradizione, è basato essenzialmente su [54] e [103].

2.1. — L'ALGORITMO Sia $(\Omega, \mathcal{F}, P)$ uno spazio di probabilità, dove Ω è interpretabile come l'insieme delle possibili traiettorie evolutive di ogni possibile popolazione iniziale; definiremo l'Algoritmo Evolutivo Astratto sulla base di funzioni aleatorie su tale spazio. Supponiamo inoltre definita in Γ una funzione di intorno $N\colon \Gamma \to 2^{\Gamma}$, dove con la scrittura 2^{Γ} si intende l'insieme delle parti di Γ. L'Algoritmo Evolutivo Astratto può essere schematizzato come segue:

1. Creare una popolazione iniziale;
2. Selezionare nella popolazione corrente un individuo per la riproduzione;
3. (Con probabilità p_{cross}) Selezionare un altro individuo nella popolazione corrente, incrociare i due individui così ottenuti e tenere l'individuo figlio risultante;
4. (Con probabilità p_{mut}) Effettuare una mutazione sull'individuo ottenuto;
5. Inserire l'individuo così ottenuto nella nuova popolazione;
6. Se la nuova popolazione non è ancora completa, ritornare al passo 2;
7. Sostituire la popolazione corrente con la nuova e ritornare al passo 2.

I passi 3 e 4 sono "casuali", nel senso che ogni volta si estrae a sorte, per ciascuno indipendentemente e con la probabilità assegnata, se eseguirli oppure no.

Sebbene al passo 3 la limitazione dell'incrocio a coppie di individui e non a gruppi più ampi, come in [41], possa apparire restrittiva, in pratica essa è usata da pressoché tutti gli autori.

A volte si richiede (per semplicità) che la consistenza numerica della popolazione sia fissata a priori e rimanga costante durante tutto lo svolgimento dell'algoritmo.

Gli operatori genetici.

2.2. — SELEZIONE Definiamo la funzione aleatoria di selezione

$$\mathbf{select}\colon \Gamma^* \times \Omega \to \Gamma$$

che estrae da una popolazione un individuo, tipicamente con una probabilità che dipende in qualche modo dalla sua idoneità; quindi, per ogni $x \in \Gamma^*$, $\omega \in \Omega$:

$$\mathbf{select}(x, \omega) \in x.$$

2.3. — Incrocio Definiamo la funzione aleatoria di incrocio

$$\mathbf{cross}\colon \Gamma \times \Gamma \times \Omega \to \Gamma$$

che da una coppia di individui genitori ricava un individuo figlio combinando in qualche modo i loro geni; il fatto che l'individuo risultante contenga informazione proveniente da entrambi i genitori viene espresso dicendo che appartiene al loro intorno, cioè, per ogni $\gamma_1, \gamma_2 \in \Gamma$, $\omega \in \Omega$:

$$\mathbf{cross}(\gamma_1, \gamma_2, \omega) \in N(\gamma_1) \cup N(\gamma_2).$$

2.4. — Mutazione La mutazione è una funzione aleatoria

$$\mathbf{mutate}\colon \Gamma \times \Omega \to \Gamma$$

che, dato un individuo, in qualche modo ne perturba il genotipo, producendo un nuovo individuo che giace ancora in un suo intorno: per ogni $\gamma \in \Gamma$, $\omega \in \Omega$,

$$\mathbf{mutate}(\gamma, \omega) \in N(\gamma).$$

2.5. — Per modellare il fatto che incrocio e mutazione vengono eseguiti in seguito a una decisione casuale con una certa probabilità, definiamo altre due funzioni aleatorie di accoppiamento e di inserimento.

La funzione aleatoria di accoppiamento

$$\mathbf{mate}\colon \Gamma \times \Gamma \times \Omega \to \Gamma,$$

dati due individui, decide, con probabilità p_{cross} di incrociarli e, in tal caso, restituisce il prodotto dell'incrocio, o, con probabilità $1 - p_{\text{cross}}$, di restituire semplicemente il primo dei due individui: per ogni $\gamma_1, \gamma_2 \in \Gamma$,

$$\mathbf{mate}(\gamma_1, \gamma_2) = \begin{cases} \mathbf{cross}(\gamma_1, \gamma_2) & \text{con probabilità } p_{\text{cross}}, \\ \gamma_1 & \text{con probabilità } 1 - p_{\text{cross}}. \end{cases}$$

La funzione aleatoria di inserimento

$$\mathbf{insert}\colon \Gamma \times \Omega \to \Gamma$$

decide, dato un individuo, se restituirlo così com'è, con probabilità $1 - p_{\text{mut}}$, oppure mutarlo, con probabilità p_{mut}: per ogni $\gamma \in \Gamma$,

$$\mathbf{insert}(\gamma) = \begin{cases} \mathbf{mutate}(\gamma)] & \text{con probabilità } p_{\text{mut}}, \\ \gamma & \text{con probabilità } 1 - p_{\text{mut}}. \end{cases}$$

La funzione di transizione.

2.6. — Supposta assegnata una popolazione iniziale $X_0 \in \Gamma^*$, la popolazione della t-esima generazione si ottiene a partire da quella della $(t-1)$-esima ripetendo un certo numero di volte, diciamo m, i passi 2, 3 e 4 dell'algoritmo. Ciò significa operare:

- m selezioni indipendenti di un individuo, eventualmente seguite da altre m selezioni di un individuo da incrociare con quello appena selezionato (quindi al massimo $2m$ selezioni indipendenti);
- al più m incroci indipendenti;
- al più m mutazioni indipendenti.

2.7. — Queste applicazioni indipendenti dello stesso operatore si realizzano associando ad ogni traiettoria evolutiva $\omega \in \Omega$, oltre alla realizzazione della popolazione corrente, le realizzazioni delle funzioni aleatorie di selezione, incrocio, mutazione, accoppiamento e inserimento secondo lo schema seguente:

$$\omega \mapsto \begin{cases} X_t(\omega), \\ \mathbf{select}_1(\cdot,\omega), \dots, \mathbf{select}_{2m}(\cdot,\omega) \\ \mathbf{cross}_1(\cdot,\cdot,\omega), \dots, \mathbf{cross}_m(\cdot,\cdot,\omega), \\ \mathbf{mate}_1(\cdot,\cdot,\omega), \dots, \mathbf{mate}_m(\cdot,\cdot,\omega), \\ \mathbf{mutate}_1(\cdot,\omega), \dots, \mathbf{mutate}_m(\cdot,\omega), \\ \mathbf{insert}_1(\cdot,\omega), \dots, \mathbf{insert}_m(\cdot,\omega). \end{cases}$$

2.8. — Quindi, applicando questi strumenti come prescritto ai passi 2, 3 e 4 dell'algoritmo, si ottiene

$$X_{t+1}(\omega) = \bigsqcup_{i=1}^{m} \mathbf{insert}_i(\mathbf{mate}_i(\mathbf{select}_{2i-1}(X_t(\omega),\omega), \mathbf{select}_{2i}(X_t(\omega),\omega),\omega),\omega),$$

dove ciascuna delle $\mathbf{insert}_i(\cdot,\omega)$ e $\mathbf{mate}_i(\cdot,\cdot,\omega)$ fa uso, rispettivamente, di $\mathbf{mutate}_i(\cdot,\omega)$ e $\mathbf{cross}_i(\cdot,\cdot,\omega)$.

2.9. — Risulta dunque definita una famiglia di funzioni aleatorie di transizione

$$T_t : \Omega \to (\Gamma^* \to \Gamma^*)$$

tali che, per ogni possibile traiettoria evolutiva $\omega \in \Omega$,

$$T_{t+1}(\omega)[\cdot] \equiv \bigsqcup_{i=1}^{\|X_{t+1}(\omega)\|} \mathbf{insert}_i(\mathbf{mate}_i(\mathbf{select}_{2i-1}(\cdot,\omega), \mathbf{select}_{2i}(\cdot,\omega),\omega),\omega),$$

che producono la popolazione della t-esima generazione a partire da quella della $(t-1)$-esima.

§ 3. — L'Algoritmo Evolutivo Astratto come Processo Stocastico.

Markovianità — Le probabilità di transizione — Misure su popolazioni e individui.

3.1. — Sia $\mathcal{B}_{\Gamma^*}$ un'opportuna σ-algebra di parti di Γ^*: l'algoritmo evolutivo astratto delineato nel §2.1 può essere allora modellato come un processo stocastico a tempo discreto

$$\{X_t(\omega)\}_{t=0,1,2,\ldots},$$

dove $X_0(\omega) = x_0$ si suppone fissato, ma potrebbe essere a sua volta aleatorio, e

$$X_{t+1}(\omega) = T_t(\omega)\,[X_t(\omega)], \quad \text{per } t \geq 0,$$

con spazio di probabilità di base $(\Omega, \mathcal{F}, \mathcal{P})$, spazio degli stati $(\Gamma^*, \mathcal{B}_{\Gamma^*})$ e insieme dei tempi l'insieme dei numeri naturali, secondo la definizione di processo stocastico che si può trovare, per esempio, in [26].

Markovianità.

3.2. — Definizione Una *catena di Markov* [42] è una sequenza di esperimenti, in cui la probabilità dell'esito di un dato esperimento dipende solo dall'esito dell'esperimento precedente. Sia X_t la variabile stocastica che denota l'esito del t-esimo esperimento; le probabilità di transizione al t-esimo esperimento sono definite, per ogni coppia x, y di esiti, come

$$P_{xy}(t) = P[X_t = y | X_{t-1} = x].$$

La matrice $\mathbf{P}(t)$ con elementi dati dall'equazione precedente è detta *matrice di transizione.*

3.3. — Lemma Se le funzioni di transizione $T_t(\omega)$, con $t > 0$, sono mutuamente indipendenti, allora il processo stocastico $\{X_t(\omega)\}_{t=0,1,2,\ldots}$ è una catena di Markov; inoltre, se le $T_t(\omega)$ sono anche identicamente distribuite, allora $\{X_t(\omega)\}_{t=0,1,2,\ldots}$ è una catena di Markov omogenea.

3.4. — Dimostrazione Le probabilità di transizione alla t-esima generazione si scrivono

$$\begin{aligned} P_{xy}(t) &= P[X_t = y | X_{t-1} = x] = \\ &= P[T_{t-1}(X_{t-1}) = y | X_{t-1} = x] = \\ &= P[T_{t-1}(x) = y]. \end{aligned}$$

Quindi, se le T_t, $t > 0$, sono mutuamente indipendenti, la popolazione alla t-esima generazione dipende solo dalla popolazione alla $(t-1)$-esima generazione; inoltre, se le T_t hanno tutte identica distribuzione, questo significa che la matrice di transizione $\mathbf{P}(i)$ è costante per ogni $t \geq 0$, quindi la catena è omogenea. ∎

3.5. — Si noti che la matrice di transizione $\mathbf{P}$ di un algoritmo evolutivo astratto, con elementi P_{xy}, $x, y \in \Gamma^*$, è in generale infinita, se la dimensione della popolazione non è superiormente limitata; nel caso invece in cui la popolazione ha consistenza numerica fissata n, $\mathbf{P}$ sarà una matrice

$$\|\Gamma^{(n)}\| \times \|\Gamma^{(n)}\| = \binom{r+n-1}{n} \times \binom{r+n-1}{n}.$$

La matrice P_{xy} è ricavata esplicitamente da Nix e Vose in [112] per un caso particolare, in cui Γ è l'insieme delle stringhe binarie di lunghezza l.

Le probabilità di transizione.

3.6. — Nell'ipotesi di funzioni di transizione T_t, $t > 0$, mutuamente indipendenti e identicamente distribuite, si può scrivere $T_t = T$ per ogni t e le probabilità di transizione possono essere calcolate come segue, per ogni coppia di popolazioni $x, y \in \Gamma^*$:

$$P_{xy} = P[T(x) = y] = \prod_{\gamma \in \Gamma} P[n_{T(x)}(\gamma) = n_y(\gamma)|x],$$

questo perché due popolazioni sono identiche se e solo se le loro funzioni di occupazione hanno uguale valore per ogni genotipo.

3.7. — Le singole probabilità nella produttoria si calcolano, per ciascun genotipo, osservando che il numero di copie di un dato genotipo nella popolazione y è dato dal numero di volte che un individuo con quel genotipo vi viene inserito in $n = \|y\|$ prove di Bernoulli indipendenti, con probabilità di successo, per ogni $\gamma \in \Gamma$, $P[\mathbf{insert}(\dots x \dots) = \gamma]$:

$$P[n_{T(x)}(\gamma) = i|x] = \binom{n}{i} P[\mathbf{insert}(\dots x \dots) = \gamma]^i (1 - P[\mathbf{insert}(\dots x \dots) = \gamma])^{n-i}.$$

3.8. — A sua volta, la probabilità di successo in una singola prova è, per ogni $\gamma \in \Gamma$, la probabilità che γ sia il risultato dell'accoppiamento, se la mutazione non avviene, oppure, se la mutazione avviene, che un altro genotipo sia il risultato dell'accoppiamento e che la mutazione lo abbia trasformato in γ:

$$\begin{aligned} P[\mathbf{insert}(\dots x \dots) = \gamma] = (1 - p_{\text{mut}}) P[\mathbf{mate}(\dots x \dots) = \gamma] + \\ + p_{\text{mut}} \sum_{\kappa \in \Gamma} P[\mathbf{mate}(\dots x \dots) = \kappa] P[\mathbf{mutate}(\kappa) = \gamma]. \end{aligned}$$

3.9. — Resta da calcolare la probabilità che un dato individuo risulti dall'accoppiamento (facoltativo) di due individui selezionati nella popolazione x; questa è data, per ogni $\gamma \in \Gamma$, dalla probabilità che γ sia selezionato "per primo" e che non avvenga incrocio, oppure, se avviene l'incrocio, che i due individui selezionati, incrociandosi, diano γ come risultato:

$$\begin{aligned} P[\mathbf{mate}(\dots x \dots) = \gamma] = (1 - p_{\text{cross}}) P[\mathbf{select}(x) = \gamma] + \\ + p_{\text{cross}} \sum_{\alpha, \beta \in \Gamma} P[\mathbf{select}(x) = \alpha] P[\mathbf{select}(x) = \beta] P[\mathbf{cross}(\alpha, \beta) = \gamma]. \end{aligned}$$

Misure su popolazioni e individui.

3.10. — Supponendo per semplicità di lavorare con una popolazione di consistenza numerica n fissata, chiamiamo $\mu_t^{(n)}(x)$ la misura di probabilità della popolazione $x \in \Gamma^{(n)}$ alla t-esima generazione, cioè

$$\mu_t^{(n)}(x) \equiv P[X_t = x],$$

e analogamente chiamiamo $\mu_t(\gamma)$ la misura di probabilità di un individuo $\gamma \in \Gamma$ alla t-esima generazione.

Conveniamo inoltre che le scritture $\mu_t^{(n)}$ e μ_t denotino i vettori stocastici le cui componenti x e γ sono rispettivamente $\mu_t^{(n)}(x)$ e $\mu_t(\gamma)$.

3.11. — Conoscendo la matrice di transizione $\mathbf{P}$, le misure $\mu_t^{(n)}$ possono essere calcolate, per ogni $t \geq 0$, utilizzando l'equazione

$$\mu_t^{(n)} = {\mu_{t-1}^{(n)}}^T \cdot \mathbf{P};$$

passando dalla notazione vettoriale a quella funzionale, questo significa semplicemente che, per ogni $x \in \Gamma^{(n)}$,

$$\mu_t^{(n)}(x) = \sum_{y \in \Gamma^{(n)}} \mu_{t-1}^{(n)}(y) P_{yx}.$$

Le misure μ_t, per ogni $t \geq 0$, possono essere calcolate di conseguenza, a partire dalle misure $\mu_t^{(n)}$, applicando il Teorema 1.5; in questo modo si può ricavare la formula che esprime μ_t in termini della μ_{t-1}:

$$\mu_t(\gamma) = \frac{1}{n} \sum_{x \in \Gamma^{(n)}} n_x(\gamma) \sum_{y \in \Gamma^{(n)}} \mu_{t-1}^{(n)}(y) P_{yx} =$$

$$= (n-1)! \sum_{x,y \in \Gamma^{(n)}} P_{yx} n_x(\gamma) \prod_{\kappa \in \Gamma} \frac{\mu_{t-1}(\kappa)^{n_y(\kappa)}}{n_y(\kappa)!}.$$

3.12. — Chiamiamo ϕ_t la funzione di probabilità dell'idoneità degli individui alla t-esima generazione del processo $\{X_t(\omega)\}_{t=0,1,2,\ldots}$. Questa funzione descrive la distribuzione dell'idoneità di un individuo pescato a caso dalla popolazione $X_t(\omega)$, e si ricava dalla misura μ_t secondo la Definizione 1.10: per ogni $v \in \mathbb{R}^+$,

$$\phi_t(v) = \mu_t(\Gamma_v).$$

§ 4. — Statistiche di Algoritmi Evolutivi.

Statistiche fenotipiche — Statistiche genotipiche — Indici di diversità.

Si possono distinguere due classi di statistiche relative a una particolare realizzazione dell'Algoritmo Evolutivo Astratto $\{X_t(\omega)\}_{t=0,1,2,\ldots}$:

a. statistiche fenotipiche, cioè che prendono in considerazione i fenotipi degli individui, e quindi la loro idoneità;
b. statistiche genotipiche, cioè che prendono in considerazione soltanto i genotipi degli individui.

Statistiche fenotipiche.

4.1. — Idoneità media L'*idoneità media* $f(x)$ di una popolazione $x \in \Gamma^*$ è l'idoneità della popolazione x definita in §I.3.14.

Data una popolazione, in generale gli individui che la compongono non saranno tra loro statisticamente indipendenti, per via degli operatori di incrocio e selezione utilizzati dall'algoritmo; tuttavia, possiamo sempre considerare una popolazione di n individui come un campione estratto da una misura di probabilità ad essa associata con n estrazioni indipendenti; è come dire che si parte dalle popolazioni e si costruisce la misura che le genera, cosicché tutta la dipendenza è incapsulata a priori nella misura.

Con questa premessa, possiamo invocare il Teorema del Limite Centrale, dal quale sappiamo che, ad ogni istante $t \geq 0$, la variabile aleatoria $f(X_t(\omega))$ ha distribuzione normale con valore atteso $E[f(X_t)] = E_\Gamma[\phi_t]$ e varianza $\mathrm{var}[f(X_t)] = \mathrm{var}_\Gamma[\phi_t]/\|X_t)\|$.

Questo significa che è possibile rendere la varianza piccola a piacere, a patto di aumentare a sufficienza la dimensione della popolazione.

4.2. — Idoneità massima Ad ogni istante $t \geq 0$ sappiamo (cfr. [105], §V.3.2) che la variabile aleatoria $f^*(X_t(\omega))$, che rappresenta l'idoneità massima, definita in §I.3.14, degli individui nella popolazione $X_t(\omega)$, è distribuita secondo la funzione di densità

$$\phi_t^*(v) = \|X_t\|[\Phi_t(v)]^{\|X_t\|-1}\phi_t(v),$$

dove

$$\Phi_t(v) = \int_0^v \phi_t(u)du$$

è la funzione cumulativa di distribuzione dell'idoneità all'istante t.

Il valore atteso dell'idoneità massima è quindi

$$E[f^*(X_t)] = \|X_t\| \int_0^\infty [\Phi_t(v)]^{\|X_t\|-1}\phi_t(v)vdv$$

e la varianza è

$$\mathrm{var}[f^*(X_t)] = \|X_t\| \int_0^\infty \left(v - \|X_t\| \int_0^\infty [\Phi_t(u)]^{\|X_t\|-1}\phi_t(u)udu\right)^2 [\Phi_t(v)]^{\|X_t\|-1}\phi_t(v)dv.$$

4.3. — Diametro dell'idoneità Il *diametro dell'idoneità* (*range* in inglese) $R(x)$ di una popolazione $x \in \Gamma^*$ è definito come la differenza tra le idoneità del migliore e del peggiore individuo in x:

$$R(x) = f^*(x) - \min_{\gamma \in x} f(\gamma).$$

Ad ogni istante $t \geq 0$ sappiamo (cfr. [105], §VI.5.2) che la variabile aleatoria $R(X_t(\omega))$ è distribuita con funzione di densità

$$\rho(r) = \|X_t\|(\|X_t\| - 1) \int_0^\infty [\Phi_t(z + r) - \Phi_t(z)]^{\|X_t\|-2} \phi_t(z + r)\phi_t(z)dz.$$

Osserviamo che, per valori di r sufficientemente piccoli, $\Phi_t(z + r) \approx \Phi_t(z) + r\phi_t(z)$ e $\phi_t(z + r) \approx \phi_t(z)$, per cui possiamo scrivere

$$\rho(r) \approx \|X_t\|(\|X_t\| - 1)r^{\|X_t\|-2} \int_0^\infty [\phi_t(z)]^{\|X_t\|}dz.$$

Statistiche genotipiche.

Un'importante gruppo di statistiche genotipiche è rappresentato da vari indici di diversità di popolazioni, le cui definizioni sono basate su altre interessanti statistiche, che sono il numero e la quota di occupazione di un genotipo in una popolazione.

4.4. — Il numero di occupazione Ad ogni istante $t \geq 0$ possiamo considerare, per ogni $\gamma \in \Gamma$, i valori $n_{X_t(\omega)}(\gamma)$ come altrettante variabili aleatorie con distribuzione binomiale

$$P[n_{X_t}(\gamma) = k] = \binom{\|X_t\|}{k} \mu_t(\gamma)^k [1 - \mu_t(\gamma)]^{\|X_t\|-k},$$

perciò con valore atteso $E[n_{X_t}(\gamma)] = \|X_t\|\mu_t(\gamma)$ e varianza $\mathrm{var}[n_{X_t}(\gamma)] = \|X_t\|\mu_t(\gamma)[1 - \mu_t(\gamma)]$.

4.5. — La quota di occupazione Analogamente, per ogni $\gamma \in \Gamma$, i valori $q_{X_t(\omega)}(\gamma)$ sono altrettante variabili aleatorie con distribuzione che può essere ricavata ricordando che $q_x(\gamma) \equiv n_x(\gamma)/\|x\|$, con valore atteso $E[q_{X_t}(\gamma)] = \mu_t(\gamma)$ e varianza

$$\mathrm{var}[q_{X_t}(\gamma)] = \frac{1}{\|X_t\|^2}\mathrm{var}[n_{X_t}(\gamma)] = \frac{\mu_t(\gamma)[1 - \mu_t(\gamma)]}{\|X_t\|}.$$

In altre parole, $q_{X_t(\omega)}(\gamma)$ è uno stimatore non distorto di $\mu_t(\gamma)$; inoltre sappiamo dal Teorema limite di De Moivre-Laplace che, per ogni coppia di numeri $a, b \in [0, 1]$, $a < b$,

$$P[a \leq q_{X_t}(\gamma) \leq b] \approx \mathrm{erf}\left(\frac{\sqrt{\|X_t\|}[a - \mu_t(\gamma)]}{\sqrt{\mu_t(\gamma)[1 - \mu_t(\gamma)]}}\right) - \mathrm{erf}\left(\frac{\sqrt{\|X_t\|}[b - \mu_t(\gamma)]}{\sqrt{\mu_t(\gamma)[1 - \mu_t(\gamma)]}}\right),$$

dove

$$\mathrm{erf}(t) = \int_{-\infty}^t \frac{e^{u^2/2}}{\sqrt{2\pi}}du.$$

Se operiamo un cambiamento di coordinate, fissando $q_{X_t}(\gamma)$, la variabile aleatoria diventa $\mu_t(\gamma)$, con la stessa distribuzione ma con valore atteso $q_{X_t}(\gamma)$. Quindi, per ogni $\epsilon > 0$,

$$P[|\mu_t(\gamma) - q_{X_t}(\gamma)| \leq \epsilon] \approx \operatorname{erf}\left(-\epsilon \frac{\sqrt{\|X_t\|}}{\sqrt{q_{X_t}(\gamma)[1 - q_{X_t}(\gamma)]}}\right) - \operatorname{erf}\left(\epsilon \frac{\sqrt{\|X_t\|}}{\sqrt{q_{X_t}(\gamma)[1 - q_{X_t}(\gamma)]}}\right).$$

Cioè, osservando le quote di occupazione in una popolazione sufficientemente numerosa, siamo in grado di calcolare intervalli di confidenza per la misura di probabilità μ_t.

Indici di diversità.

4.6. — Un indice di diversità che ha una suggestiva interpretazione probabilistica è il seguente:

$$D(X_t(\omega)) = 1 - \sum_{\gamma \in \Gamma} \frac{n_{X_t(\omega)}(\gamma)[n_{X_t(\omega)}(\gamma) - 1]}{\|X_t(\omega)\|(\|X_t(\omega)\| - 1)}.$$

La quantità $D(X_t(\omega))$ rappresenta la probabilità che due individui estratti a caso con distribuzione uniforme dalla popolazione $X_t(\omega)$ abbiano genotipi differenti. Questo fatto suggerisce un comodo algoritmo per stimare $D(X_t(\omega))$, estraendo a caso un certo numero di coppie di individui dalla popolazione $X_t(\omega)$ e contando la frequenza su questo campione delle coppie eterogenee.

L'indice $D(X_t(\omega))$ è uno stimatore non distorto della quantità

$$1 - \sum_{\gamma \in \Gamma} \mu_t(\gamma)^2,$$

che misura in un certo senso l'"ampiezza", o la dispersione, della misura μ_t.

4.7. — Un altro indice di diversità, più agevole da calcolare, è il seguente:

$$\bar{D}(X_t(\omega)) = \frac{1}{\|X_t(\omega)\| - 1} \left(\sum_{\gamma \in \Gamma} [n_{X_t(\omega)}(\gamma) > 0] - 1 \right),$$

dove con la scrittura [predicato] si intende 1 laddove il predicato è vero, e 0 altrimenti. L'indice $\bar{D}(X_t(\omega))$ stima la quantità

$$\frac{\|\Gamma\|}{\|\Gamma\| - 1} \cdot \frac{\|\operatorname{supp}(\mu_t)\| - 1}{\|\operatorname{supp}(\mu_t)\|},$$

dove $\operatorname{supp}(\mu_t) \equiv \{\gamma \in \Gamma : \mu_t(\gamma) > 0\}$ è il sottoinsieme di supporto della misura μ_t. Anche questa quantità misura la dispersione della misura μ_t, ma senza prenderne in considerazione la "sagoma".

4.8. — Entropia L'entropia di una popolazione $X_t(\omega)$ può essere definita, se si adotta per comodità la convenzione che $0 \log_2 \frac{1}{0} = 0$, come

$$H(X_t(\omega)) = \sum_{\gamma \in \Gamma} q_{X_t(\omega)} \log_2 \frac{1}{q_{X_t(\omega)}}.$$

Questa quantità fornisce una stima dell'entropia della misura μ_t, che a sua volta può essere definita come

$$H(\mu_t) = \sum_{\gamma \in \Gamma} \mu_t(\gamma) \log_2 \frac{1}{\mu_t(\gamma)}.$$

§ 5. — Convergenza ad almeno un genotipo ottimale.

Teoremi di convergenza — Condizioni per la convergenza

5.1. — Definizione Si denoti con $\Gamma_O \equiv \{\gamma \in \Gamma : \forall \kappa \in \Gamma, f(\gamma) \geq f(\kappa)\}$ il sottospazio dei genotipi corrispondenti ai fenotipi ottimi, cioè alle migliori soluzioni del problema oggetto codificabili per mezzo dello schema prescelto; analogamente, si denoti con $\Gamma^*_O \equiv \{x \in \Gamma^* : x \sqcap \Gamma_O \neq \emptyset\}$ il sottospazio delle popolazioni contenenti almeno un fenotipo ottimo.

5.2. — Per dimostrare che $\{X_t(\omega)\}_{t=0,1,2,\ldots}$ converge ad almeno un genotipo ottimale, occorre mostrare che, per ogni traiettoria evolutiva $\omega \in \Omega$, la probabilità che $X_t(\omega)$ contenga un ottimo, cioè che $X_t(\omega) \in \Gamma^*_O$, tende a 1 per t tendente all'infinito.

5.3. — Definizione $\{X_t(\omega)\}_{t=0,1,2,\ldots}$ è *monotono* se, per ogni $\omega \in \Omega$, $t > 0$, $f^*(X_t(\omega)) \geq f^*(X_{t-1}(\omega))$.

5.4. — Definizione Si denoti con

$$\mathbf{reach}(x) \equiv \{y \in \Gamma^* : \exists k \geq 0 : \left[\mathbf{P}^k\right]_{xy} > 0\}$$

l'insieme delle popolazioni, ovvero gli stati del processo, raggiungibili a partire dalla popolazione $x \in \Gamma^*$ in un numero finito di generazioni.

5.5. — Definizione In maniera analoga, diciamo che un genotipo γ è *raggiungibile* dal processo $\{X_t(\omega)\}_{t=0,1,2,\ldots}$ se esiste un tempo $0 \leq t < \infty$ tale che $P[\gamma \in X_t] > 0$.

Teoremi di convergenza.

Il teorema seguente, dovuto ad Aarts, Eiben e van Hee, fornisce lo strumento tecnico per dimostrare i successivi risultati generali sulla convergenza ad almeno un genotipo ottimale dell'algoritmo evolutivo astratto.

5.6. — Teorema Sia $\{X_t(\omega)\}_{t=0,1,2,\ldots}$ monotono, con $X_0 = x_0 \in \Gamma^*$; se esistono una successione di indici $\{t_k\}_{k\geq 0}$ ed una successione reale $\{\epsilon_k \in (0,1]\}_{k\geq 0}$ tali che

1. $\lim_{k\to\infty} t_k = \infty$;
2. $\lim_{k\to\infty} \prod_{l=0}^{k} \epsilon_l = 0$;
3. per ogni popolazione $y \in \mathbf{reach}(x_0)$ e per ogni $k \geq 0$,

$$P[X_{t_{k+1}} \notin \Gamma_O^* | X_{t_k} = y] \leq \epsilon_k;$$

allora $\{X_t(\omega)\}_{t=0,1,2,\ldots}$ raggiunge un ottimo q.c., cioè $P[\lim_{t\to\infty} X_t \in \Gamma_O^* | X_0 = x_0] = 1$.

5.7. — Dimostrazione [1] L'idea principale di questa dimostrazione consiste nel calcolare una maggiorazione per la probabilità di prendere la strada sbagliata, cioè di effettuare una transizione che non conduce ad alcun ottimo. La probabilità di raggiungere un ottimo si può scrivere, in virtù dell'ipotesi 1:

$$\begin{aligned} P[\lim_{t\to\infty} X_t \in \Gamma_O^* | X_0 = x_0] &= \lim_{k\to\infty} P[X_{t_k} \in \Gamma_O^* | X_0 = x_0] = \\ &= 1 - \lim_{k\to\infty} P[X_{t_k} \notin \Gamma_O^* | X_0 = x_0]. \end{aligned}$$

Ora, $P[X_{t_k} \notin \Gamma_O^* | X_0 = x_0]$ rappresenta la probabilità di seguire la strada sbagliata fino alla t_k-esima generazione e, ricordando che il processo $\{X_t(\omega)\}_{t=0,1,2,\ldots}$ è monotono e che quindi una volta raggiunto un ottimo non lo può più abbandonare, si calcola come segue:

$$\begin{aligned} &P[X_{t_k} \notin \Gamma_O^* | X_0 = x_0] = \\ &= P[X_{t_0} \notin \Gamma_O^* | X_0 = x_0] P[X_{t_1} \notin \Gamma_O^* | X_{t_0}] \ldots P[X_{t_k} \notin \Gamma_O^* | X_{t_{k-1}}] = \\ &= P[X_{t_0} \notin \Gamma_O^* | X_0 = x_0] \prod_{l=0}^{k} P[X_{t_{l+1}} \notin \Gamma_O^* | X_{t_l}] \leq \\ &\text{(per l'ipotesi 3)} \\ &\leq P[X_{t_0} \notin \Gamma_O^* | X_0 = x_0] \prod_{l=0}^{k} \epsilon_l \leq \prod_{l=0}^{k} \epsilon_l; \end{aligned}$$

donde, per l'ipotesi 2,

$$\lim_{k\to\infty} P[X_{t_k} \notin \Gamma_O^* | X_0 = x_0] \leq \lim_{k\to\infty} \prod_{l=0}^{k} \epsilon_l = 0,$$

da cui discende la tesi. ∎

Il teorema che segue invece collega la convergenza ad almeno un genotipo ottimale alla scelta di una popolazione iniziale.

5.8. — Teorema Sia data una popolazione iniziale $X_0 = x_0 \in \Gamma^*$ per la quale il processo $\{X_t(\omega)\}_{t=0,1,2,\ldots}$ è monotono ed omogeneo e per ciascuna popolazione $y \in \mathbf{reach}(x_0)$ esiste almeno un genotipo ottimale raggiungibile; allora il processo $\{X_t(\omega)\}_{t=0,1,2,\ldots}$ raggiunge almeno un genotipo ottimo q.c., cioè $\lim_{t\to\infty} P[X_t \in \Gamma_O^* | X_0] = 1$.

5.9. — Dimostrazione Si applica il Teorema 5.6 costruendo le due successioni $\{t_k\}$ ed $\{\epsilon_k\}$ in modo che ne soddisfino le condizioni.

Sia $T_y = \inf\{t : P[X_t \in \Gamma_O^* | X_0 = y] > 0\}$ il numero minimo di generazioni necessarie perché ci sia una probabilità non nulla che nella popolazione sia presente almeno un genotipo ottimale. Poiché per ogni $y \in \mathbf{reach}(x_0)$ esiste almeno un genotipo ottimale raggiungibile, esiste $t < \infty$ tale che $P[X_t \in \Gamma_O^* | X_0 = y] > 0$, e quindi, per ogni $y \in \mathbf{reach}(x_0)$, T_y è finito e così anche

$$T \equiv \sup_{y \in \mathbf{reach}(x_0)} T_y < \infty.$$

La monotonia di $\{X_t(\omega)\}_{t=0,1,2,\ldots}$ implica che, per ogni $y \in \mathbf{reach}(x_0)$,

$$P[X_T \in \Gamma_O^* | X_0 = y] > 0$$

e, di conseguenza,

$$p_y \equiv P[X_T \notin \Gamma_O^* | X_0 = y] < 1.$$

Si definisca $p = \sup_{y \in \mathbf{reach}(x_0)} p_y$; ovviamente, per ogni $y \in \mathbf{reach}(x_0)$,

$$P[X_T \notin \Gamma_O^* | X_0 = y] \leq p < 1.$$

Definiamo allora $t_k = kT$ ed $\epsilon_k = p$ per ogni $k \geq 0$; osserviamo che, come richiesto, $\lim_{k\to\infty} t_k = \infty$ e che $\lim_{k\to\infty} \prod_{l=0}^{k} \epsilon_l = \lim_{k\to\infty} p^{k+1} = 0$, dato che $p < 1$. Resta da dimostrare che la Condizione 3 del Teorema 5.6 sia soddisfatta. A questo proposito, per l'omogeneità di $\{X_t(\omega)\}_{t=0,1,2,\ldots}$, si ha che, per ogni $y \in \mathbf{reach}(x_0)$,

$$P[X_{t_{k+1}} \notin \Gamma_O^* | X_{t_k} = y] = P[X_{(k+1)T} \notin \Gamma_O^* | X_{kT} = y] = P[X_T \notin \Gamma_O^* | X_0 = y] \leq p = \epsilon_k,$$

e questo completa la dimostrazione. ∎

Condizioni per la convergenza.

5.10. — Definizione Data una popolazione $x \in \Gamma^*$, denotiamo con

$$\mathbf{élite}(x) \equiv \{\gamma \in x : \forall \kappa \in x, f(\gamma) \geq f(\kappa)\}$$

l'insieme dei genotipi più idonei tra quelli degli individui che fanno parte della popolazione x.

5.11. — Definizione Diciamo che una funzione di transizione $T(\omega)$ è *elitista*, se conserva sempre almeno uno dei genotipi più idonei di ogni popolazione, cioè se, per ogni $x \in \Gamma^*$,

$$\mathbf{élite}(x) \sqcap T(\omega)(x) \neq \emptyset.$$

Aarts, Eiben e van Hee in [41] usano il termine *conservativa* per questa proprietà; tuttavia il termine normalmente impiegato nella comunità degli Algoritmi Evolutivi per

designare quelle strategie che assicurano che l'individuo migliore di una popolazione passi sempre alla popolazione successiva è "elitista".

È facile dimostrare che se, per ogni $\omega \in \Omega$, la funzione di transizione $T(\omega)$ è elitista, allora il processo $\{X_t(\omega)\}_{t=0,1,2,\ldots}$ è monotono.

5.12. — NOTAZIONE Denotiamo con $\hookrightarrow \subseteq \Gamma^2$ la chiusura transitiva della relazione $\{(\gamma,\delta) \in \Gamma^2 : \delta \in N(\gamma)\}$: $\gamma \hookrightarrow \delta$ significa che si può andare da γ a δ passando da un genotipo a un altro nel suo intorno.

5.13. — DEFINIZIONE La struttura degli intorni è *connettiva* se saltando da un intorno all'altro è possibile arrivare a qualsiasi individuo a partire da qualsiasi altro, cioè se, per ogni $\gamma, \delta \in \Gamma$, $\gamma \hookrightarrow \delta$.

Un indebolimento di questo concetto è dato dalla seguente

5.14. — DEFINIZIONE La struttura degli intorni è *quasi connettiva* se esiste almeno un genotipo $\sigma \in \Gamma$ da cui è possibile arrivare a qualsiasi altro genotipo saltando da un intorno a un altro parzialmente sovrapposto al precedente, cioè se, per ogni $\gamma \in \Gamma$, $\sigma \hookrightarrow \gamma$.

5.15. — DEFINIZIONE Se la struttura degli intorni è quasi connettiva e $\sigma \in \Gamma$ è un individuo tale che, per ogni $\gamma \in \Gamma$, $\sigma \hookrightarrow \gamma$, una funzione di transizione $T(\omega)$ è *σ-elitista* se, per ogni popolazione $x \in \Gamma^*$,

$$\sigma \in x \quad \text{implica} \quad \sigma \in T(\omega)(x).$$

5.16. — DEFINIZIONE Diciamo che **select** è *generosa* se, per ogni $x \in \Gamma^*$ e per ogni $\gamma \in \Gamma$,

$$n_x(\gamma) > 0 \quad \text{implica} \quad P[\mathbf{select}(x) = \gamma] > 0.$$

5.17. — DEFINIZIONE Diciamo che **mutate** è *generosa* se, per ogni $\gamma, \kappa \in \Gamma$,

$$\kappa \in N(\gamma) \quad \text{implica} \quad P[\mathbf{mutate}(\gamma) = \kappa] > 0.$$

5.18. — TEOREMA Siano **select** e **mutate** generose; la struttura degli intorni sia connettiva e le funzioni di transizione $T_t(\omega)$, per $t > 0$, siano indipendenti, identicamente distribuite e, per ogni $\omega \in \Omega$, elitiste. Allora l'algoritmo evolutivo astratto trova sempre almeno un genotipo ottimale q.c. per ogni popolazione iniziale x_0, cioè:

$$P[\lim_{t\to\infty} X_t \in \Gamma^*_O] = 1.$$

5.19. — DIMOSTRAZIONE Si invoca il Teorema 5.8, mostrando che le sue condizioni valgono per ogni scelta della popolazione iniziale $x_0 \in \Gamma^*$.

Che $\{X_t(\omega)\}_{t=0,1,2,\ldots}$ è monotono segue banalmente dal fatto che le funzioni di transizione sono elitiste.

Che $\{X_t(\omega)\}_{t=0,1,2,\ldots}$ è omogeneo segue dal Lemma 3.3, essendo per ipotesi le funzioni di transizione T_t, per $t > 0$, indipendenti e identicamente distribuite.

Resta da dimostrare che, per ogni popolazione iniziale x_0 e per ogni popolazione $y \in \mathbf{reach}(x_0)$ esiste almeno un ottimo raggiungibile, cioè che esiste un genotipo ottimale $\tilde{\gamma} \in \Gamma_O$ tale che esiste un indice $t' \geq t$ per cui $P[\tilde{\gamma} \in X_{t'} | X_t = y] > 0$; ma in realtà, per la generosità di **select** e **mutate**, unitamente al fatto che la struttura degli intorni è connettiva, per ogni $\gamma \in \Gamma$ esiste un $t \geq t'$ tale che $P[\gamma \in X_{t'} | X_t = y] > 0$, e questo conclude la dimostrazione. ∎

Si può rilassare l'ipotesi sulla struttura connettiva degli intorni al prezzo di un'ulteriore restrizione sulle funzioni di transizione. Si può dimostrare il seguente teorema:

5.20. — Teorema Fatte salve le altre ipotesi del Teorema 5.18, siano le funzioni di transizione $T_t(\omega)$, per $t > 0$, σ-elitiste per ogni $\omega \in \Omega$, e la struttura degli intorni sia quasi connettiva. Allora l'algoritmo evolutivo astratto trova almeno un genotipo ottimale q.c. per ogni popolazione iniziale x_0 che contenga almeno un genotipo σ definito come nella Definizione 5.14.

5.21. — Dimostrazione Anche qui, come per il Teorema 5.18, si invoca il Teorema 5.8. Il processo $\{X_t(\omega)\}_{t=0,1,2,\ldots}$ è monotono ed omogeneo per lo stesso ragionamento fatto in §5.19. Mostriamo ora che, per ogni $y \in \mathbf{reach}(x_0)$ ed ogni $\tilde{\gamma} \in \Gamma_O$, esiste una generazione t in cui $P[\tilde{\gamma} \in X_t] > 0$. Essendo $\sigma \in x_0$, per il σ-elitismo della funzione di transizione $\sigma \in y$; dunque qualsiasi $\gamma \in \Gamma$ può essere raggiunto da y, compreso $\tilde{\gamma} \in \Gamma_O$. Il resto della dimostrazione è identico a quella del Teorema 5.18. ∎

§ 6. — Il Teorema degli Schemi rivisitato.

Il Teorema degli Schemi, dovuto a John H. Holland [65], (cfr. Cap. I, §6.12) è stato a lungo considerato come la giustificazione teorica fondamentale degli Algoritmi Genetici; sebbene la dimostrazione di Holland sia ristretta agli Algoritmi Genetici propriamente detti, è possibile estenderla agli Algoritmi Evolutivi in generale.

Recentemente tuttavia alcuni studiosi, primo fra tutti Heinz Mühlenbein [106], hanno messo in dubbio che il Teorema degli Schemi sia di qualche utilità, obiettando che, nella migliore delle ipotesi, si tratta in pratica di una tautologia.

Quella esposta qui di seguito è una nuova versione del Teorema degli Schemi, più generale di quella di Holland e adattata alla successione di misure di probabilità $\{\mu_t\}_{t=0,1,2,\ldots}$, invece che ad una particolare realizzazione del processo evolutivo: questo dovrebbe superare le obiezioni di cui sopra.

6.1. — Definizione Uno *schema* è un qualsiasi sottoinsieme non vuoto $S \subseteq \Gamma$. [1]

[1]) Nella definizione di Holland, che lavora con genotipi costituiti da stringhe di cifre binarie di lunghezza fissata l, per cui $\Gamma = \{0,1\}^l$, uno schema è una stringa modello formata da simboli nell'insieme $\{0, 1, \star\}$, dove "$\star$" svolge il ruolo di simbolo "jolly": uno schema dunque rappresenta tutte le stringhe che concordano con esso in tutte le posizioni non marcate con il simbolo "$\star$".

Se Γ è un insieme finito, ci sono $2^{\|\Gamma\|} - 1$ schemi distinti. Ogni schema induce una bipartizione di Φ e $M(S) \subseteq \Phi$ è chiamato *iperpiano* definito da S.

6.2. — Definizione L'*ordine* $o(S)$ di uno schema S è definito come

$$o(S) \equiv \log_2 \frac{\|\Gamma\|}{\|S\|}.$$

Questa quantità può essere interpretata come l'informazione di S, cioè il minimo numero di bit necessari per specificare S. (²)

6.3. — Idoneità di uno schema L'idoneità *assoluta* di uno schema S è

$$f(S) \equiv \frac{1}{\|S\|} \sum_{\gamma \in S} f(\gamma).$$

Questa quantità è l'idoneità attesa di un individuo estratto a caso, con probabilità uniforme, da S.

L'idoneità *relativa* di uno schema S rispetto a una popolazione $x \in \Gamma^*$ è

$$f_x(S) \equiv \frac{1}{q_x(S)} \sum_{\gamma \in S} q_x(\gamma) f(\gamma).$$

Questa quantità è l'idoneità attesa di un individuo estratto a caso dalla popolazione x, dato che appartenga allo schema S.

6.4. — Questo ci porta a definire l'idoneità relativa attesa di uno schema S all'istante t del processo evolutivo come

$$E[f_{X_t}(S)] = \frac{1}{\mu_t(S)} \sum_{\gamma \in S} \mu_t(\gamma) f(\gamma) = \frac{1}{\mu_t(S)} E_S[\phi_t].$$

All'idoneità relativa attesa di uno schema si contrappone l'idoneità attesa della popolazione allo stesso istante:

$$E[f(X_t)] = \sum_{\gamma \in \Gamma} \mu_t(\gamma) f(\gamma) = E_\Gamma[\phi_t].$$

Si osservi che $E_\Gamma[\phi_t] = E_S[\phi_t] + E_{\bar{S}}[\phi_t]$, quindi $E_\Gamma[\phi_t] \geq E_S[\phi_t]$.

6.5. — Definizione Diciamo che la selezione è *proporzionale all'idoneità* se l'operatore di selezione **select** è tale che, per ogni $\gamma \in \Gamma$ e $x \in \Gamma^*$,

$$P[\mathbf{select}(x) = \gamma] = \frac{n_x(\gamma) f(\gamma)}{\|x\| f(x)}.$$

²) Nella definizione di Holland, l'ordine di uno schema è dato dal numero di posizioni fisse (0 o 1) nella stringa modello che lo rappresenta.

A questo punto abbiamo tutti gli elementi necessari per enunciare la seguente versione del Teorema degli Schemi.

6.6. — Teorema (Degli Schemi) Se la selezione è proporzionale all'idoneità, per ogni schema $S \subseteq \Gamma$ e ad ogni istante $t = 0, 1, \ldots$,

$$o(S) < \log_2 E\left[\frac{f_{X_t}(S)}{f(X_t)}\right] \quad \text{implica} \quad \mu_{t+1}(S) > \mu_t(S).$$

6.7. — Dimostrazione Per dimostrare questo teorema, calcoliamo $\mu_{t+1}(S)$ tenendo conto degli effetti cumulativi della selezione, dell'incrocio e della mutazione. La probabilità che un individuo selezionato dalla popolazione X_t con probabilità proporzionale alla sua idoneità appartenga allo schema S è data da

$$P[\mathbf{select}(X_t) \in S] = q_{X_t}(S)\frac{f_{X_t}(S)}{f(X_t)}.$$

Si osservi che questa probabilità dipende da X_t e se vogliamo svincolarla dalla particolare realizzazione della popolazione, dobbiamo calcolarne il valore atteso. Ricordiamo (cfr. §4.5) che $\mu_t(\cdot) = E[q_{X_t}(\cdot)]$; allora

$$\mu_{t+1}(S) = E[q_{X_{t+1}}(S)] = E[P[\mathbf{select}(X_t) \in S]]P_{\text{surv}}[S] = \mu_t(S)E\left[\frac{f_{X_t}(S)}{f(X_t)}\right]P_{\text{surv}}[S],$$

dove $P_{\text{surv}}[S]$ denota la probabilità che lo schema S sopravviva all'incrocio e alla mutazione, cioè

$$P_{\text{surv}}[S] = P[\mathbf{insert}(\mathbf{mate}(\gamma, \mathbf{select}(X_t))) \in S | \gamma \in S].$$

Ora, non abbiamo fatto alcuna assunzione su **mutate** e **cross**, per cui diremo che, se l'individuo selezionato viene toccato dall'incrocio o dalla mutazione o da entrambe le cose, il nuovo individuo che si ottiene può essere considerato come estratto a caso da Γ. Ma la probabilità che un tale individuo appartenga allo schema S è $\frac{\|S\|}{\|\Gamma\|}$, per la Probabilità Classica, ovvero $2^{-o(S)}$, in virtù della Definizione 6.2. Ovviamente questo è solo un limite inferiore, perché è possibile che lo schema sopravviva per altri motivi che non per il puro caso, per cui, ponendo $\alpha = (1 - p_{\text{cross}})(1 - p_{\text{mut}})$, possiamo scrivere

$$P_{\text{surv}}[S] \geq \alpha + (1 - \alpha)2^{-o(S)} = 2^{-o(S)} + \alpha(1 - 2^{-o(S)}) \geq 2^{-o(S)}.$$

Riassumendo, abbiamo stabilito che

$$\mu_{t+1}(S) \geq \mu_t(S)E\left[\frac{f_{X_t}(S)}{f(X_t)}\right]2^{-o(S)},$$

quindi siamo sicuri che $\mu_{t+1}(S) > \mu_t(S)$ solo quando

$$E\left[\frac{f_{X_t}(S)}{f(X_t)}\right]2^{-o(S)} > 1,$$

cioè quando

$$2^{-o(S)} > \frac{1}{E\left[\frac{f_{X_t}(S)}{f(X_t)}\right]},$$

ovvero, facendo il logaritmo di ambo i membri e cambiando di segno,

$$o(S) < \log_2 E\left[\frac{f_{X_t}(S)}{f(X_t)}\right]$$

e questo conclude la dimostrazione. ∎

6.8. — In parole povere, il Teorema degli Schemi ci dice che la misura di probabilità di uno schema di ordine sufficientemente basso o con idoneità relativa sufficientemente superiore alla media aumenta passando da una generazione alla successiva.

Questo risultato è stato interpretato in passato dicendo che "generazioni successive allocano un numero esponenzialmente crescente di tentativi (o individui) agli schemi di ordine basso e sopra la media"; chiaramente, però, questo è vero solo a livello locale, perché con il procedere dell'evoluzione la media si alza e un individuo che era sopra la media prima o poi diviene un individuo medio o, magari, inferiore alla media.

Anche ammettendo che quest'interpretazione non fosse molto significativa, resta tuttavia il fatto che il Teorema degli Schemi, almeno in questa versione di tipo probabilistico, può essere d'aiuto per dimostrare risultati di convergenza.

§ 7. — Algoritmi Evolutivi come Martingale.

In questo paragrafo si dimostra che, date alcune ipotesi molto ragionevoli, l'Algoritmo Evolutivo Astratto può essere collegato al concetto di Martingala: si caratterizzerà infatti il processo $\{f(X_t)\}_{t=0,1,\ldots}$ dell'idoneità della popolazione come una submartingala stretta e uniformemente integrabile.

Una martingala è, in origine, una correggia attaccata ai finimenti della bocca e fermata al pettorale del cavallo, per impedirgli di alzare troppo la testa. In Matematica, una martingala è un processo stocastico reale il cui valore atteso in un tempo futuro data la sua storia passata è dato semplicemente dal valore presente.

Il fatto che un Algoritmo Evolutivo sia caratterizzabile come una submartingala rende possibile dimostrarne la convergenza in modo semplice ed elegante.

7.1. — Definizione (cfr. [95], §4, [147], §10.3, [26], §1.6) Un processo stocastico $\{X_t\}_{t=0,1,\ldots}$ sullo spazio di probabilità $(\Omega, \mathcal{F}, P)$, adattato alla filtrazione $\{\mathcal{F}_t\}_{t=0,1,\ldots}$ è una *martingala* se, per ogni $t \in \mathbb{N}$, $E[|X_t|] < \infty$ e

$$E[X_{t+1}|\mathcal{F}_t] = X_t.$$

$\{X_t\}_{t=0,1,\ldots}$ è una *submartingala* se

$$E[X_{t+1}|\mathcal{F}_t] \geq X_t;$$

è una submartingala *stretta* se non vale l'uguaglianza.

Una martingala $\{X_t\}_{t=0,1,\ldots}$ è *uniformemente integrabile* se

$$\lim_{N\to\infty} \sup_{t\geq 0} E[|X_t| \cdot I_{|X_t|>N}] = 0.$$

I due teoremi che seguono sono in realtà due versioni leggermente differenti dello stesso risultato: il primo asserisce che, di due genotipi, il migliore tende in media a espandersi più dell'altro nella stessa popolazione; il secondo asserisce la stessa cosa, ma in maniera condizionata.

7.2. — Teorema Se nell'Algoritmo Evolutivo Astratto la selezione è proporzionale all'idoneità (cfr. §6.5), allora la funzione di transizione T è tale che, per ogni $\gamma, \kappa \in \Gamma$,

$$f(\gamma) > f(\kappa) \quad \text{implica} \quad \frac{\mu_{t+1}(\gamma)}{\mu_t(\gamma)} > \frac{\mu_{t+1}(\kappa)}{\mu_t(\kappa)}.$$

7.3. — Dimostrazione Per come è definita la selezione, possiamo scrivere, per ogni $\gamma \in \Gamma$,

$$\mu_{t+1}(\gamma) = \mu_t(\gamma)\frac{f(\gamma)}{E[f(X_t)]}P_{\text{surv}},$$

dove P_{surv} è la probabilità che un individuo passi immutato attraverso le fasi di incrocio e mutazione, $P_{\text{surv}} \geq (1 - p_{\text{cross}})(1 - p_{\text{mut}})$, ed è uguale per tutti gli individui di una popolazione fissata, anche se, in generale, può variare tra popolazioni diverse. Allora possiamo riscrivere la disuguaglianza

$$\frac{\mu_{t+1}(\gamma)}{\mu_t(\gamma)} > \frac{\mu_{t+1}(\kappa)}{\mu_t(\kappa)}$$

come

$$\frac{f(\gamma)}{E[f(X_t)]}P_{\text{surv}} > \frac{f(\kappa)}{E[f(X_t)]}P_{\text{surv}}$$

ovvero, dividendo ambo i membri per $P_{\text{surv}}/E[f(X_t)]$,

$$f(\gamma) > f(\kappa).$$

■

7.4. — Teorema Se nell'Algoritmo Evolutivo Astratto la selezione è proporzionale all'idoneità (cfr. §6.5), allora la funzione di transizione T è tale che, per ogni $\gamma, \kappa \in \Gamma$ tali che $0 < q_{X_t}(\gamma) < 1$ e $0 < q_{X_t}(\kappa) < 1$,

$$f(\gamma) > f(\kappa) \quad \text{implica} \quad \frac{E[q_{X_{t+1}}(\gamma)|X_t]}{q_{X_t}(\gamma)} > \frac{E[q_{X_{t+1}}(\kappa)|X_t]}{q_{X_t}(\kappa)}.$$

7.5. — Dimostrazione Per come è definita la selezione, possiamo scrivere, per ogni $\gamma \in \Gamma$,

$$E[q_{X_{t+1}}(\gamma)|X_t] = q_{X_t}(\gamma)\frac{f(\gamma)}{f(X_t)}P_{\text{surv}}.$$

Allora possiamo riscrivere la disuguaglianza della tesi come

$$\frac{f(\gamma)}{f(X_t)}P_{\text{surv}} > \frac{f(\kappa)}{f(X_t)}P_{\text{surv}}$$

ovvero, dividendo ambo i membri per $P_{\text{surv}}/f(X_t)$,

$$f(\gamma) > f(\kappa).$$

■

Ora, per dimostrare che da questa condizione segue la proprietà di submartingala, abbiamo bisogno del seguente

7.6. — LEMMA Siano f e g due funzioni di probabilità su $\mathbb{R}$; se, per ogni $x, y \in \text{supp}(f) \cup \text{supp}(g)$,

$$x > y \quad \text{implica} \quad \begin{cases} g(y) = 0, & \text{se } g(x) = 0; \\ g(x)/f(x) > g(y)/f(y), & \text{se } f(x) > 0 \text{ e } g(x) > 0; \\ f(x) = 0, & \text{se } f(y) = 0; \end{cases}$$

allora $E[g] > E[f]$.

7.7. — DIMOSTRAZIONE Osserviamo che f e g possono assumere lo stesso valore al più in un solo punto e, perciò, esiste almeno un punto $a \in \mathbb{R}$ tale che

$$\begin{aligned} g(x) < f(x) \quad & \text{se } x < a; \\ g(x) > f(x) \quad & \text{se } x > a. \end{aligned}$$

Ora, si costruisca una nuova funzione h, definita come segue, per ogni $x \in \mathbb{R}$:

$$h(x) = \begin{cases} g(x) & \text{se } x < a; \\ f(x) & \text{se } x \geq a. \end{cases}$$

Si verifica che

$$\int_{-\infty}^{+\infty} xh(x)dx > \int_{-\infty}^{+\infty} xf(x)dx = E[f];$$

infatti,

$$\int_{-\infty}^{+\infty} xh(x)dx = \int_{-\infty}^{a} xg(x)dx + \int_{a}^{+\infty} xf(x)dx$$

e

$$\int_{-\infty}^{a} xg(x)dx > \int_{-\infty}^{a} xf(x)dx.$$

Analogamente,

$$\int_{-\infty}^{+\infty} xh(x)dx < \int_{-\infty}^{+\infty} xg(x)dx = E[g];$$

quindi $E[g] > \int_{-\infty}^{+\infty} xh(x)dx > E[f]$. ■

I due teoremi che seguono danno due risultati che derivano, rispettivamente, dai Teoremi 7.2 e 7.4 tramite l'applicazione del lemma appena dimostrato.

7.8. — Teorema Se la funzione di transizione T è tale che, per ogni $\gamma, \kappa \in \Gamma$,

$$f(\gamma) > f(\kappa) \quad \text{implica} \quad \frac{\mu_{t+1}(\gamma)}{\mu_t(\gamma)} > \frac{\mu_{t+1}(\kappa)}{\mu_t(\kappa)},$$

allora $E[f(X_{t+1})] > E[f(X_t)]$.

7.9. — Dimostrazione Dall'Ipotesi segue banalmente che, se $u > v$,

$$\frac{\mu_{t+1}(\Gamma_u)}{\mu_t(\Gamma_u)} > \frac{\mu_{t+1}(\Gamma_v)}{\mu_t(\Gamma_v)};$$

ricordando che, per ogni $t \geq 0$, $\mu_t(\Gamma_u) = \phi_t(u)$, la stessa cosa può essere scritta come

$$\frac{\phi_{t+1}(u)}{\phi_t(u)} > \frac{\phi_{t+1}(v)}{\phi_t(v)}.$$

Le due misure ϕ_t e ϕ_{t+1} soddisfano quindi le ipotesi del Lemma 7.6, invocando il quale si dimostra che $E_\Gamma[\phi_{t+1}] > E_\Gamma[\phi_t]$. Ma $E_\Gamma[\phi_t] = E[f(X_t)]$ e $E_\Gamma[\phi_{t+1}] = E[f(X_{t+1})]$, per cui la tesi è dimostrata. ∎

7.10. — Teorema Se la funzione di transizione T è tale che, per ogni $\gamma, \kappa \in \Gamma$,

$$f(\gamma) > f(\kappa) \quad \text{implica} \quad \frac{E[q_{X_{t+1}}(\gamma)|X_t]}{q_{X_t}(\gamma)} > \frac{E[q_{X_{t+1}}(\kappa)|X_t]}{q_{X_t}(\kappa)}.$$

allora $E[f(X_{t+1})] > f(X_t)$.

7.11. — Dimostrazione Si convenga di denotare

$$q_{X_t}(u) \equiv \sum_{\substack{\gamma \in \Gamma \\ f(\gamma)=u}} q_{X_t}(\gamma).$$

Allora dall'Ipotesi segue banalmente che, se $u > v$,

$$\frac{E[q_{X_{t+1}}(u)|X_t]}{q_{X_t}(u)} > \frac{E[q_{X_{t+1}}(v)|X_t]}{q_{X_t}(v)};$$

le funzioni $q_{X_t}(u)$ e $E[q_{X_{t+1}}(u)|X_t]$, con $u \in \mathbb{R}^+$, sono due misure (sono non negative ed hanno integrale unitario su $\mathbb{R}^+$) con valore atteso rispettivamente $f(X_t)$ e $E[f(X_{t+1}|X_t]$: dunque possiamo invocare il Lemma 7.6, ottenendo $E[f(X_{t+1}|X_t] > f(X_t)$, che è la nostra tesi. ∎

7.12. — Osservazione Un corollario di questo teorema è che il processo derivato $\{f(X_t)\}_{t=0,1,\ldots}$, che assume il valore dell'idoneità media della popolazione X_t dell'Algoritmo Evolutivo Astratto ad ogni istante $t \geq 0$, è una submartingala.

Il Teorema 7.8 ci dà una caratterizzazione ancora più stringente, in quanto afferma che il valore atteso *non condizionato,* cioè calcolato su tutte le traiettorie $\omega \in \Omega$, dell'idoneità media della popolazione X_t cresce monotonamente.

7.13. — Allora, per il Teorema di Convergenza "in avanti" di Doob (cfr. [147], §11.5) esiste quasi certamente una variabile aleatoria $f_\infty \equiv \lim_{t\to\infty} f(X_t) < \infty$. Precisamente, per ogni traiettoria $\omega \in \Omega$,

$$f_\infty(\omega) \equiv \limsup f(X_t(\omega)) \equiv \inf_{s\in\mathbb{N}} \left\{ \sup_{t\geq s} f(X_t(\omega)) \right\}.$$

Se il genotipo ottimo ha una probabilità non nulla di essere generata, allora, per il Lemma "aspettarsi ciò che è quasi inevitabile" (cfr. [147], §10.11), $P[f_\infty = f_O] = 1$.

§ 8. — **Velocità di convergenza.**

8.1. — Dalla Definizione 6.5 di selezione proporzionale all'idoneità, trascurando per il momento l'effetto dell'incrocio e della mutazione, segue che

$$\mu_{t+1}(\Gamma_v) = \mu_t(\Gamma_v)\frac{v}{E[f(X_t)]};$$

passando dalla notazione genotipica a quella fenotipica, si ottiene l'equazione riproduttiva

$$\phi_{t+1}(v) = \phi_t(v)\frac{v}{\int_0^\infty u d\phi_t(u)}. \qquad (ER)$$

Questa equazione può essere risolta in modo abbastanza semplice facendo ricorso a un espediente suggerito in [38]: si supponga che la popolazione cresca secondo l'equazione di crescita esponenziale disaccoppiata, per ogni $\gamma \in \Gamma$,

$$E[n_{X_{t+1}}(\gamma)] = E[n_{X_t}(\gamma)]f(\gamma).$$

Dividendo per la cardinalità attesa della popolazione ad ogni generazione e ricordando che $\mu_t(\gamma) = E[q_{X_t}(\gamma)]$, si ottiene, per ogni $\gamma \in \Gamma$,

$$\mu_{t+1}(\gamma) = \frac{E[n_{X_t}(\gamma)]f(\gamma)}{\sum_{\kappa\in\Gamma} E[n_{X_{t+1}}(\kappa)]} = \frac{E[n_{X_t}(\gamma)]f(\gamma)}{\sum_{\kappa\in\Gamma} E[n_{X_t}(\kappa)]f(\kappa)} = \frac{\mu_t(\gamma)f(\gamma)}{\sum_{\kappa\in\Gamma} \mu_t(\kappa)f(\kappa)},$$

dividendo numeratore e denominatore per $\|X_t\|$.

Si noti che un altro modo di scrivere il denominatore è ora

$$\sum_{\kappa\in\Gamma} \mu_t(\kappa)f(\kappa) = \int_0^\infty u d\phi_t(u);$$

inoltre, per l'additività delle misure μ possiamo scrivere

$$\mu_{t+1}(\Gamma_v) = \frac{v\mu_t(\Gamma_v)}{\int_0^\infty u d\phi_t(u)},$$

che è identica all'Equazione riproduttiva che assume una popolazione di cardinalità fissata.

8.2. — Ora, l'equazione di crescita disaccoppiata può essere risolta direttamente, permettendo così di esprimere il numero atteso di copie di un genotipo γ alla t-esima generazione come

$$E[n_{X_t}(\gamma)] = f(\gamma)^t E[n_{X_0}(\gamma)].$$

Dividendo ambo i membri per la cardinalità attesa della popolazione alla t-esima generazione

$$E[\|X_t\|] = \sum_{\kappa\in\Gamma} E[n_{X_t}(\kappa)] = \sum_{\kappa\in\Gamma} f(\kappa)^t E[n_{X_0}(\kappa)],$$

si ottiene

$$\mu_t(\gamma) = \frac{f(\gamma)^t E[n_{X_0}(\gamma)]}{\sum_{\kappa\in\Gamma} f(\kappa)^t E[n_{X_0}(\kappa)]}.$$

Questa equazione può essere riscritta, ricordando che $E[n_{X_0}(\gamma)] = \mu_0(\gamma)E[\|X_0\|]$, come

$$\mu_t(\gamma) = \frac{f(\gamma)^t \mu_0(\gamma)E[\|X_0\|]}{\sum_{\kappa\in\Gamma} f(\kappa)^t \mu_0(\kappa)E[\|X_0\|]} = \frac{f(\gamma)^t \mu_0(\gamma)}{\sum_{\kappa\in\Gamma} f(\kappa)^t \mu_0(\kappa)},$$

ovvero, passando alla notazione fenotipica,

$$\phi_t(v) = \frac{v^t \phi_0(v)}{\int_0^\infty u^t \phi_0(u)du}.$$

8.3. — Supponendo che la popolazione iniziale sia caratterizzata da una distribuzione Gamma dell'idoneità con parametri r e λ

$$\phi_0(v) = \frac{\lambda}{\Gamma(r)}(\lambda v)^{r-1}e^{-\lambda v},$$

si ottiene

$$\phi_t(v) = \frac{v^{t+r-1}e^{-\lambda v}}{\int_0^\infty u^{t+r-1}e^{-\lambda u}du}.$$

Ponendo invece $\phi_0(v) = \lambda e^{-\lambda v}$, cioè una distribuzione esponenziale, che è un caso particolare della distribuzione Gamma con $r = 1$, la soluzione sarebbe

$$\phi_t(v) = \frac{v^t e^{-\lambda v}}{\int_0^\infty u^t e^{-\lambda u}du} = \frac{\lambda}{\Gamma(t+1)}(\lambda v)^t e^{-\lambda v},$$

il che permette di concludere che, in generale, con distribuzione Gamma iniziale con parametri r e λ,

$$\phi_t(v) = \frac{\lambda}{\Gamma(r+t)}(\lambda v)^{t+r-1}e^{-\lambda v}.$$

8.4. — Questo risultato ci permette di calcolare l'andamento dell'idoneità attesa della popolazione al passare delle generazioni:

$$E[\phi_t] = \frac{r+t}{\lambda},$$

che cresce linearmente col tempo. Tenendo presente che $f(X_0) = \frac{r}{\lambda}$, è possibile, data una realizzazione della popolazione iniziale X_0, stimare i parametri r e λ cercando la coppia di valori che meglio approssima la distribuzione osservata, per esempio con il metodo dei momenti, e utilizzare questa stima per ricavare un'indicazione del numero di generazioni necessarie perché la popolazione raggiunga un'idoneità media desiderata.

8.5. — Chiaramente l'ipotesi di una distribuzione Gamma può essere ragionevole qualora non si abbia idea del valore massimo che l'idoneità può assumere; tuttavia, se esiste, l'ottimo globale avrà un'idoneità precisa, sebbene sconosciuta, che costituisce un limite superiore ai valori di idoneità osservabili. Quando la distribuzione dell'idoneità osservata nella popolazione giace ben al di sotto di questo limite, il ragionamento fatto sopra fornirà una buona approssimazione del comportamento della popolazione; per un modello più fedele è però necessario esaminare il caso con idoneità superiormente limitata.

8.6. — Per quanto riguarda il caso con idoneità superiormente limitata, ci si può sempre ridurre, con opportune trasformazioni, al caso in cui $0 \leq f(\gamma) \leq 1$ per ogni $\gamma \in \Gamma$. Supponendo che la popolazione iniziale sia caratterizzata da una distribuzione Beta con parametri $a, b > 0$

$$\phi_0(v) = \frac{v^{a-1}(1-v)^{b-1}}{B(a,b)},$$

dove

$$B(a,b) = \int_0^1 u^{a-1}(1-u)^{b-1}du,$$

si può scrivere, sostituendo nella soluzione dell'Equazione riproduttiva,

$$\phi_t(v) = \frac{v^t\phi_0(v)}{\int_0^1 u^t\phi_0(u)du} = \frac{v^{t+a-1}(1-v)^{b-1}}{\int_0^1 u^{t+a-1}(1-u)^{b-1}du} = \frac{v^{t+a-1}(1-v)^{b-1}}{B(t+a,b)}.$$

Si noti che un caso particolare della distribuzione Beta, quando $a = b = 1$, è la distribuzione uniforme nell'intervallo $[0, 1]$.

8.7. — Il valore atteso dell'idoneità della popolazione è dato da

$$E[\phi_t] = \frac{t+a}{t+a+b},$$

che cresce come un iperbole equilatera con asintoto a 1 per t tendente all'infinito. Anche qui siamo in presenza di una convergenza lineare: infatti, il numero atteso di generazioni $T(\epsilon)$ necessario perché lo scostamento medio della popolazione dall'ottimo globale si di ϵ unità di idoneità, è dato da

$$T(\epsilon) = \frac{b}{\epsilon} - a - b.$$

Per esempio, con distribuzione iniziale uniforme, il numero atteso di generazioni per ottenere un'idoneità media pari a 0.99 è $T(0.01) = 98$.

Data una popolazione iniziale con una certa distribuzione, è possibile trovare i parametri a e b che meglio la spiegano in base all'ipotesi di una distribuzione Beta e sfruttare i parametri così calcolati per dare una stima approssimativa della velocità di convergenza.

Un esempio: il problema MAXONE.

8.8. — MAXONE è un problema giocattolo molto semplice: i genotipi sono stringhe binarie di lunghezza l, $\Gamma = \{0,1\}^l$ e l'obiettivo è massimizzare il numero degli 1 nella stringa; la funzione di idoneità è definita come la frequenza del simbolo 1 nella stringa: per ogni $\gamma \in \Gamma$,

$$f(\gamma) = \frac{1}{l}\sum_{i=1}^{l} \gamma_i.$$

Si supponga di generare la popolazione iniziale decidendo il valore di ciascuna polizione del genotipo di ogni individuo con il lancio di una monetina bilanciata; allora la distribuzione iniziale dell'idoneità è

$$\phi_0(v) = \binom{l}{vl} 2^{-l} \quad v = 0, \frac{1}{l}, \frac{2}{l}, \ldots, 1.$$

8.9. — Con il metodo dei momenti centrali stimiamo i parametri a e b della distribuzione Beta che meglio spiegano questa distribuzione binomiale:

$$E[\phi_0] = \tfrac{a}{a+b} \qquad E[\phi_0] = \tfrac{1}{2}$$

$$\mathrm{var}[\phi_0] = \tfrac{ab}{(a+b+1)(a+b)^2} \quad \mathrm{var}[\phi_0] = \tfrac{1}{4l}$$

Risolvendo si ottiene $a = b$ e $4la^2 = (2a+1)4a^2$, per cui

$$a = b = \frac{l-1}{2}.$$

Ne segue che

$$E[\phi_t] = \frac{t + \frac{l-1}{2}}{t + l - 1}$$

e quindi

$$T(\epsilon) = \frac{l-1}{2\epsilon} - l + 1.$$

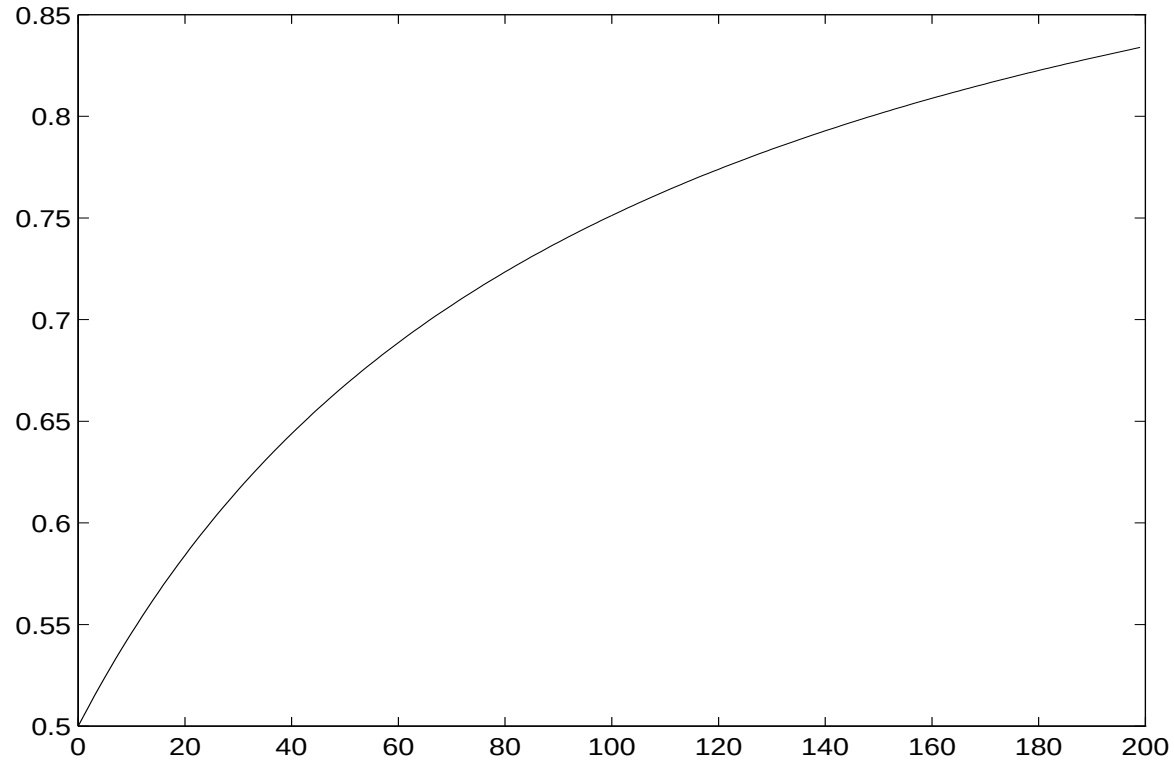

FIGURA II.1. — Grafico teorico dell'idoneità media $E[\phi_t]$ per MAXONE, con $l = 100$ ed $n = 1000$.

Per esempio, con $l = 100$ ed $\epsilon = 0.01$ ci si aspetterebbe, in media,

$$T(0.01) = \frac{99}{0.02} - 99 = 4851$$

generazioni; l'idoneità media dovrebbe crescere come

$$E[\phi_t] = \frac{49.5 + t}{99 + t},$$

il cui grafico è mostrato in Figura II.1.

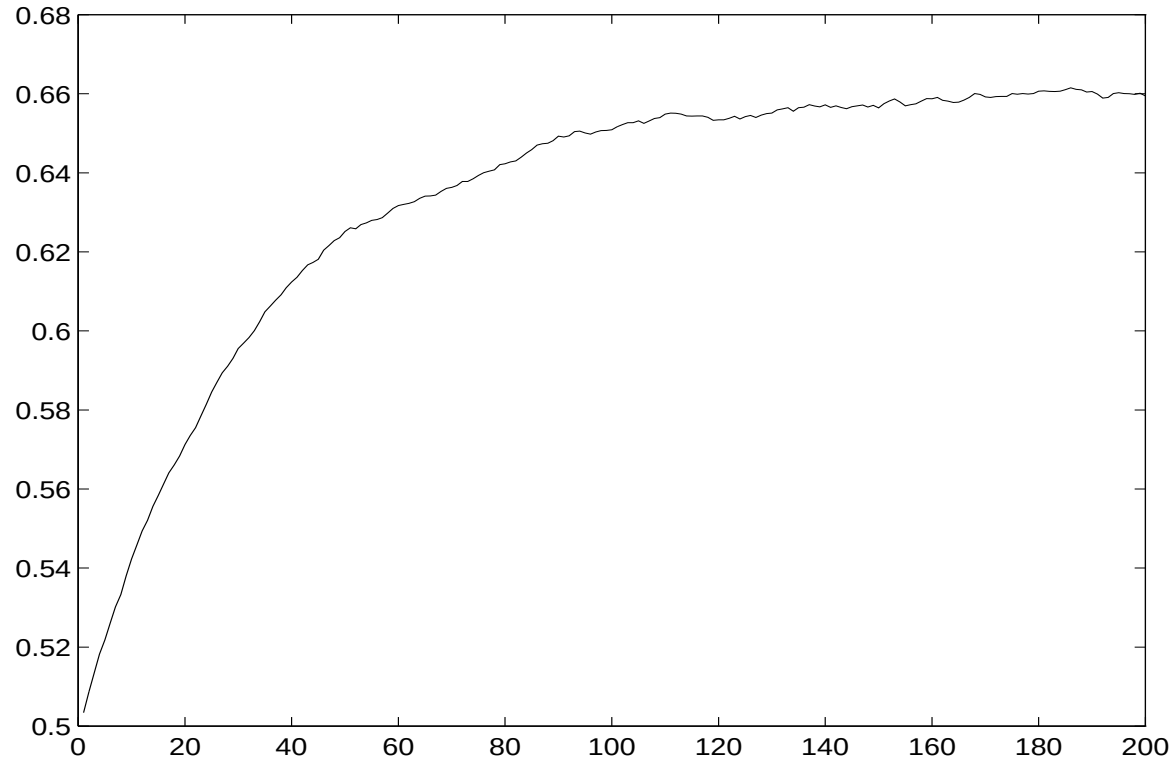

FIGURA II.2. — Grafico sperimentale dell'idoneità media per MAXONE, con $l = 100$, $n = 1000$, $p_{\text{mut}} = 0.01$ e $p_{\text{cross}} = 0.6$.

8.10. — In Figura II.2 è riportato un grafico dell'andamento dell'idoneità ottenuto sperimentalmente facendo la media di dieci esecuzioni di un algoritmo genetico canonico per il problema MAXONE con $l = 100$, tasso di mutazione $p_{\text{mut}} = 0.01$, tasso di incrocio $p_{\text{cross}} = 0.6$ e popolazione di $n = 1000$ individui.

Si può osservare che, sebbene per le prime quaranta generazioni il grafico segua con pochi scostamenti l'andamento teorico calcolato in §8.9, nelle generazioni successive l'effetto di disturbo causato dall'incrocio e dalla mutazione si fa sentire in misura sempre crescente, bloccando addirittura la convergenza a un livello di idoneità media intorno allo 0.66.

Ciò è imputabile in larga misura all'effetto della mutazione. Infatti, ripetendo l'esperimento con $p_{\text{mut}} = 0$, a parità degli altri parametri, si ottiene il grafico in Figura II.3, che, sorprendentemente, esibisce una convergenza più rapida di quella prevista, anche se l'assenza della mutazione non assicura la convergenza all'ottimo globale.

Il dato sperimentale dimostra chiaramente che, per studiare la velocità di convergenza di un algoritmo evolutivo, non basta tenere conto della sola selezione, ma è necessario modellare anche gli effetti cumulati della mutazione e, sebbene in misura minore, dell'incrocio.

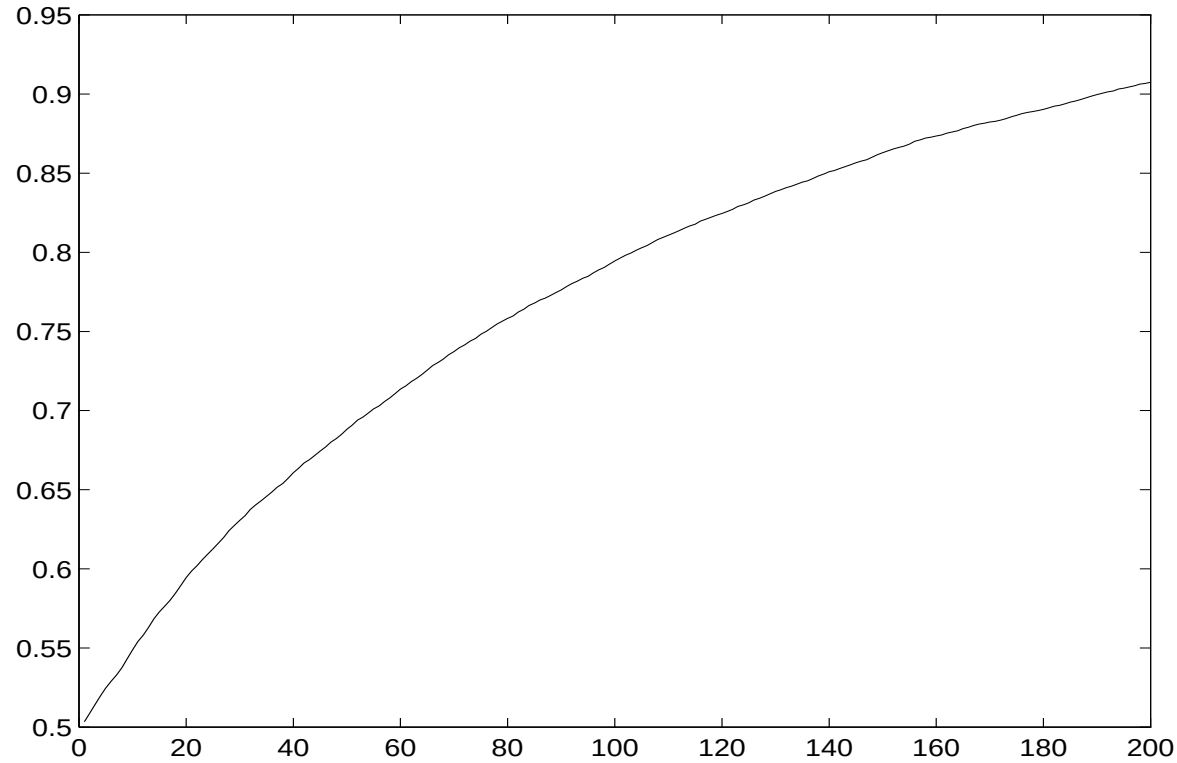

FIGURA II.3. — Grafico sperimentale dell'idoneità media per MAXONE, con $l = 100$, $n = 1000$, $p_{\text{mut}} = 0$ e $p_{\text{cross}} = 0.6$.

Sono noti altri risultati parziali come quello ricavato sopra:

8.11. — TEOREMA [4] In un algoritmo evolutivo con popolazione di cardinalità n senza mutazione né incrocio ($p_{\text{mut}} = p_{\text{cross}} = 0$) e con selezione casuale, il valore atteso del numero di generazioni T per la convergenza ad una popolazione stazionaria, composta di n individui identici, è

$$E[T] = O(n).$$

8.12. — TEOREMA [4] Sia $\Gamma = \{0,1\}^l$; il valore atteso del tempo di convergenza T di un algoritmo evolutivo con popolazione iniziale generata a caso, con incrocio uniforme e con selezione casuale, è

$$E[T] = O(n \log l).$$

Questo risultato può forse essere estrapolato per Γ qualsiasi ponendo $l = \log_2 \Gamma$.

Conclusione.

8.13. — Come si può osservare da quanto illustrato sopra, si è ancora lontani da una teoria soddisfacente della velocità di convergenza per gli Algoritmi Evolutivi e il

lavoro in questa direzione è estremamente complesso ed irto di ostacoli, sebbene sia già stato possibile compiere qualche passo significativo.

Capitolo III

UN MODELLO DISTRIBUITO DI SELEZIONE

Questo capitolo introduce ed analizza un modello di selezione locale per gli Algoritmi Evolutivi basato sul concetto di competizione, particolarmente adatto all'implementazione su elaboratori ad architettura massicciamente parallela.

In molte applicazioni, come l'evoluzione di trategie di gioco o la sintesi di procedure di controllo, la competizione tra due individui su un'istanza del problema oggetto scelta secondo qualche distribuzione di probabilità può costituire una valida alternativa rispetto alla definizione di una funzione di idoneità che inglobi conoscenza a priori del problema, cosa che non è nelle possibilità di chiunque, richiedendo una comprensione approfondita del problema unitamente a una certa quale destrezza di tipo matematico.

Il modello viene definito e ne vengono fornite le principali proprietà (§2); alcuni esempi applicativi illustrano come si può sfruttare il concetto di competizione così definito per l'implementazione di un algoritmo evolutivo. Si investigano poi (§3) le connessioni tra i concetti di competizione e di idoneità: in particolare si definisce un'idoneità *a posteriori* basata sul risultato atteso delle competizioni e si mostra come la competizione può essere definita a partire da una data idoneità *a priori* in modo consistente, cioè tale per cui la corrispondente idoneità a posteriori sia eguale a quella a priori a meno di una costante moltiplicativa, e questo nei due casi in cui l'idoneità sia limitata o illimitata superiormente.

Il capitolo si conclude (§4) con un'analisi del comportamento locale di una popolazione soggetta alla selezione competitiva, effettuata attraverso lo studio dell'andamento del tasso riproduttivo di un individuo rispetto alla sua idoneità (a priori o a posteriori) con diverse ipotesi sulla distribuzione dell'idoneità nella popolazione.

§ 1. — **Motivazioni.**

1.1. — Nel Capitolo I, §9, sono stati illustrati alcuni dei modelli classici di selezione normalmente impiegati negli Algoritmi Evolutivi e largamente esplorati in letteratura; l'esigenza di sfruttare elaboratori a parallelismo massiccio [31, 40], costringe l'implementatore di un Algoritmo Evolutivo a ricorrere a modelli di selezione locali, come la Selezione a Torneo Locale, con tutte le possibili varianti che si trovano nei modelli globali.

D'altra parte, uno dei passaggi critici nella progettazione e implementazione di un Algoritmo Evolutivo efficace consiste tradizionalmente nel definire una funzione di idoneità che incapsuli, nella maggior quantità possibile, ogni conoscenza o caratteristica

del problema che possa essere utile per la sua risoluzione. Questo implica ovviamente la disponibilità di una qualche stima significativa della distanza di una soluzione arbitraria dalla soluzione ottima. In generale, purtroppo, questo compito è tutt'altro che banale e spesso difficile, in quanto richiede una conoscenza profonda del problema, cioè della topologia dello spazio delle soluzioni ammissibili, e la capacità di trasformare questa conoscenza in espressioni matematiche.

1.2. — Lo scopo del lavoro di ricerca illustrato in questo capitolo è sostanzialmente di chiarire che la definizione esplicita di una funzione di idoneità, sebbene sicuramente sufficiente, non costituisce una condizione necessaria per costruire un Algoritmo Evolutivo in grado di ottenere risultati utili. L'alternativa proposta è un nuovo modello di selezione basato sul concetto di *competizione*: in sintesi, si introduce un operatore stocastico che determina, secondo una qualche procedura che dipenderà dal problema oggetto ma sarà caratterizzata tipicamente da un basso costo computazionale, un vincitore tra due individui dati.

La motivazione intuitiva per l'introduzione di questo modello deriva dall'osservazione che la competizione tra individui per assicurarsi qualche risorsa vitale la cui disponibilità sia limitata gioca un ruolo chiave nella Selezione Naturale. Tra gli esempi naturali di questo tipo di competizione si possono citare i combattimenti per l'accaparramento di cibo e i duelli tra maschi di alcune specie animali per guadagnare il diritto ad accoppiarsi con una femmina.

Inoltre, il processo stesso della Selezione Naturale appare intrinsecamente locale, distribuito ed asincrono.

§ 2. — Un modello basato sulla competizione.

2.1. — Definizione Introduciamo una funzione stocastica

$$\mathbf{winner} : \Gamma \times \Gamma \to \Gamma$$

che, dati due individui, che chiameremo *avversari,* ne sceglie uno, che chiameremo *vincitore,* secondo una certa probabilità; si denoterà con $P[\mathbf{winner}(\gamma, \kappa) = \gamma]$ la probabilità che l'individuo γ vinca la competizione contro l'avversario κ.

Da questa definizione segue banalmente:

1. $P[\mathbf{winner}(\gamma, \kappa) = \gamma] + P[\mathbf{winner}(\gamma, \kappa) = \kappa] = 1$;
2. $P[\mathbf{winner}(\gamma, \kappa) = \gamma] = P[\mathbf{winner}(\kappa, \gamma) = \gamma]$.

2.2. — Esempio Un semplice esempio dovrebbe chiarire come la funzione stocastica **winner** possa essere suggerita in modo naturale, per alcune classi di problemi, dalla natura stessa del problema oggetto (questo esempio sarà ripreso e sviluppato a fondo nel Capitolo IV).

Si supponga che Γ sia una collezione finita di formule, ciascuna delle quali rappresenta una funzione analitica reale di una variabile reale; con un leggero abuso di notazione possiamo scrivere $\gamma(x) \in \mathbb{R}$ per il valore della funzione rappresentata da γ calcolata nel punto $x \in \mathbb{R}$. Si desidera trovare la formula in Γ che approssima meglio

una funzione bersaglio $\clubsuit\colon \mathbb{R} \to \mathbb{R}$, che si suppone di essere in grado di calcolare in ogni punto $x \in \mathbb{R}$, per esempio compiendo un qualche esperimento, pur ignorando la sua espressione analitica.

In questo contesto, si può definire $\mathbf{winner}(\gamma, \kappa)$ immaginando che i due avversari γ e κ provino quanto vicino riescono ad andare al valore $\clubsuit(x)$ per un $x \in \mathbb{R}$ estratto a caso: a questo scopo sia per esempio $X \in \mathbb{R}$ una variabile aleatoria con distribuzione normale standard $\mathcal{N}(0, 1)$; allora

$$\mathbf{winner}(\gamma, \kappa) = \begin{cases} \gamma, & \text{se } |\gamma(X) - g(X)| < |\kappa(X) - g(X)|; \\ \kappa, & \text{se } |\gamma(X) - g(X)| > |\kappa(X) - g(X)|. \end{cases}$$

In caso di pareggio, cioè quando $|\gamma(X) - g(X)| = |\kappa(X) - g(X)|$, si può semplicemente lanciare una moneta equa e decretare vincitore l'individuo γ se l'esito del lancio è testa e κ se è croce. Si ha quindi

$$\begin{aligned} P[\mathbf{winner}(\gamma, \kappa) = \gamma] = {} & P[|\gamma(X) - g(X)| < |\kappa(X) - g(X)|] + \\ & + \frac{1}{2} P[|\gamma(X) - g(X)| = |\kappa(X) - g(X)|]. \end{aligned}$$

Si ponga $A = \{x : |\gamma(x) - g(x)| < |\kappa(x) - g(x)|\}$ e $B = \{x : |\gamma(x) - g(x)| = |\kappa(x) - g(x)|\}$; allora

$$P[\mathbf{winner}(\gamma, \kappa) = \gamma] = \int_A \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2} dx + \frac{1}{2} \int_B \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2} dx.$$

2.3. — Esempio Questo esempio suggerisce un modo per risparmiare una quantità importante di sforzo computazionale in applicazioni che sfruttano il modello della Programmazione Genetica (cfr. I.§7).

Nella Programmazione Genetica il calcolo dell'idoneità viene effettuato testando il programma corrispondente a ciascun individuo presente nella popolazione su un certo insieme predeterminato di dati di ingresso, o casi, e misurando l'errore cumulativo rispetto alle specifiche. Anche per problemi relativamente semplici, l'assegnamento di un valore di idoneità rappresenta un compito computazionalmente pesante, in particolare perché lo stesso programma, corrispondente a un individuo, deve essere eseguito un certo numero di volte con dati d'ingresso differenti al fine di ottenere una stima significativa della sua bontà. Questa è la ragione per la quale l'adozione della selezione competitiva potrebbe condurre a un miglioramento sostanziale delle prestazioni in molte applicazioni della Programmazione Genetica..

Si supponga dunque che Γ sia l'insieme dei programmi scritti in un dato linguaggio di programmazione; siano I l'insieme delle possibili configurazioni dei dati d'ingresso ed O quello dei possibili valori di uscita di un programma in Γ; per mantenere un alto livello d'astrazione, si assuma che le specifiche siano tali da rendere possibile, dati due valori $o_1, o_2 \in O$ per la stessa identica configurazione $i \in I$ dei dati di ingresso, decidere in genere, anche se non necessariamente in tutti i casi, quale si avvicini maggiormente al valore desiderato.

La competizione può allora essere eseguita come segue: si estrae una configurazione $i \in I$ dei dati di ingresso secondo una qualche misura di probabilità p su I, che può dipendere, per esempio, dalla frequenza osservata delle configurazioni nel contesto applicativo per cui si cerca di evolvere un programma; si eseguono entrambi i programmi

avversari $\gamma, \kappa \in \Gamma$ con lo stesso ingresso i, ottenendo due valori di uscita $\gamma(i), \kappa(i) \in O$; quindi

$$\mathbf{winner}(\gamma, \kappa) = \begin{cases} \gamma & \text{se } \gamma(i) \text{ è migliore di } \kappa(i) \\ \kappa & \text{se } \gamma(i) \text{ è peggiore di } \kappa(i) \end{cases}$$

Qui, ancora, se non è possibile decidere quale dei due valori sia migliore, cioè più vicino al valore desiderato, il vincitore può essere determinato a caso con probabilità $\frac{1}{2}$.

Se $A \subseteq I$ è l'insieme delle istanze sulle quali il programma γ restituisce un risultato migliore di quello restituito dal programma κ, e $B \subseteq I$ è quello delle istanze per le quali i risultati dei due programmi non sono confrontabili,

$$P[\mathbf{winner}(\gamma, \kappa) = \gamma] = p(A) + \frac{1}{2}p(B).$$

Si noti che lo stesso schema si applica con poche modifiche anche a quegli algoritmi evolutivi che evolvono, per esempio, reti neuronali o controllori.

2.4. — Esempio L'intento di quest'ultimo esempio è di illustrare come una funzione stocastica **winner** appropriata possa essere costruita per aggirare l'effettivo calcolo di una funzione di idoneità particolarmente pesante. Il problema oggetto è quello finanziario della Selezione di Portafoglio, che sorge quando un investitore deve decidere come dividere una certa somma tra un certo numero di possibili investimenti. Dettagli su come questo problema è stato affrontato per mezzo di un algoritmo evolutivo si trovano in [3]. Qui Γ è l'insieme di tutti i portafogli ammissibili e si desidera massimizzare il valore atteso del rendimento cercando nello stesso tempo di tenere il rischio quanto più possibile basso; è oltre lo scopo di questo esempio entrare nei dettagli: basti sapere che il rischio $R(\gamma)$ e il rendimento atteso $E(\gamma)$ associati al portafoglio $\gamma \in \Gamma$ sono calcolati sulla base di un grande numero di osservazioni storiche H_{it} dei rendimenti di ciascuno degli investimenti $i = 1, \ldots, N$ nei periodi $t = 1, \ldots, T$. Le dimensioni tipiche di questo problema sono un centinaio di possibili forme di investimento, ciascuna con qualche centinaio di rendimenti storici osservati.

Ora, il rendimento atteso di un portafoglio $\gamma \in \Gamma$ è normalmente calcolato dalla formula

$$E(\gamma) = \frac{1}{NT} \sum_{i=1}^{N} \sum_{t=1}^{T} \gamma_i H_{it},$$

mentre $R(\gamma)$ è un qualche indicatore di rischio, eventualmente non lineare e discontinuo; l'idoneità di γ è ottenuta come combinazione convessa del rendimento atteso e del rischio $z(R(\gamma), E(\gamma))$. Invece di calcolare le idoneità di due individui e quindi confrontarle, è possibile definire $\mathbf{winner}(\gamma, \kappa)$ come segue: si scelga un periodo $0 < t \leq T$ e un investimento $0 < i \leq N$ a caso con probabilità uniforme; allora

$$\mathbf{winner}(\gamma, \kappa) = \begin{cases} \gamma & \text{se } \gamma_i H_{it} > \kappa_i H_{it} \\ \kappa & \text{se } \gamma_i H_{it} < \kappa_i H_{it} \end{cases}$$

Questo richiede $\frac{1}{NT}$ del tempo necessario per calcolare i due valori di idoneità; la probabilità che γ vinca una competizione contro κ è data da

$$P[\mathbf{winner}(\gamma, \kappa) = \gamma] = P[H_{it} > 0]P[\gamma_i > \kappa_i] + P[H_{it} < 0]P[\gamma_i < \kappa_i].$$

Non c'è dubbio che, con una competizione definita in questo modo, in molti casi possa avvenire che il portafoglio peggiore riesca a vincere sul migliore, ma va tenuto in considerazione che, nello stesso tempo in cui verrebbe calcolata l'idoneità di un singolo individuo, può essere effettuato un numero di competizioni dell'ordine delle 100.000.

2.5. — Il modello Si denoti con n la cardinalità della popolazione; è possibile sfruttare la funzione stocastica **winner** per costruire un modello generale di selezione come segue:

for $i = 1$ **to** n **do**
begin
$\gamma :=$ *old-pop*$[\mathcal{U}(1,n)]$;
$\kappa :=$ *old-pop*$[\mathcal{U}(1,n)]$;
new-pop$[i] :=$ **winner**(γ, κ);
end;

dove $\mathcal{U}(1,n)$ denota una variabile aleatoria intera con distribuzione uniforme sull'insieme $\{1,\ldots,n\}$.

Questo è quasi esattamente l'algoritmo della selezione a torneo locale binario. L'unica ma sostanziale differenza è che qui viene usato un operatore stocastico **winner** per determinare quale individuo possa riprodursi, invece di ricorrere a un criterio basato sull'idoneità.

2.6. — Questo modello di selezione ben si presta a un'implementazione parallela: per esempio in un'architettura ad elevato parallelismo a ciascun elemento di calcolo può essere assegnato un genotipo γ; ad ogni passo dell'algoritmo, ciascun elemento di calcolo può scegliere a caso uno tra i suoi vicini immediati rispetto alla topologia delle interconnessioni, leggere il suo genotipo κ, eseguire una competizione e quindi sostituire il vincitore **winner**(γ, κ) a γ.

In un modello di calcolo a sciame [104, 13], in cui gli elementi di calcolo possono essere immaginati come particelle di un gas, che fluttuano liberamente e occasionalmente interagiscono venendo a contatto l'una con l'altra, l'interazione tra due elementi di calcolo con associati i genotipi γ e κ consisterebbe nella disputa di una competizione e nella sostituzione, su entrambi, del vecchio individuo col vincitore **winner**(γ, κ).

2.7. — Il numero atteso di occorrenze di un genotypo γ nella popolazione X_{t+1} ottenuta dalla popolazione X_t tramite l'applicazione della selezione competitiva illustrata sopra può essere calcolato in

$$E_c[n_{X_{t+1}}(\gamma)] = \frac{2n_{X_t}(\gamma)}{n} \sum_{\kappa \in \Gamma} n_{X_t}(\kappa) P[\mathbf{winner}(\gamma, \kappa) = \gamma]. \qquad (1)$$

§ 3. — Competizione e idoneità.

Idoneità a priori e a posteriori — Idoneità illimitata — Idoneità superiormente limitata.

La selezione competitiva non richiede la definizione esplicita di una funzione di idoneità. Tuttavia, si può immaginare che, pur senza essere mai calcolata completamente dall'algoritmo, l'idoneità sia presente e contribuisca a determinare gli esiti delle competizioni. Questo concetto può essere esplicitato in due modi complementari.

Idoneità a priori e a posteriori.

3.1. — Da un lato, si può partire con una definizione dell'operatore **winner** e definire l'idoneità di un individuo sulla base della sua probabilità attesa di vittoria in una competizione contro un altro individuo qualsiasi. Tale misura di idoneità, ottenuta osservando gli esiti delle competizioni, potrebbe essere chiamata *a posteriori.*

Dall'altro lato, ci si può immaginare di essere già in grado di calcolare esattamente l'idoneità di ogni individuo dato. Questa misura di idoneità, data sin dall'inizio, potrebbe essere chiamata *a priori.* Con queste premesse, si può partire dall'idoneità a priori e definire l'operatore **winner** sulla base di questa.

3.2. — Ora, si pone la questione di stabilire se queste due interpretazioni possano essere conciliate in un unico quadro consistente. In altre parole, si vuole rispondere alla domanda se esista un modo di assegnare le probabilità agli esiti delle competizioni tra due individui sulla base della loro idoneità a priori, tale che l'idoneità a posteriori, ottenuta tramite l'osservazione degli esiti, risulti proporzionale all'idoneità a priori.

Se questo è, come si vedrà, il caso, allora sarà possibile, dato un algoritmo evolutivo che faccia uso della selezione competitiva, scriverne un altro equivalente che utilizzi un'idoneità corrispondente e una forma di selezione a torneo locale probabilistico. Dal punto di vista pratico, se l'algoritmo che fa uso della selezione competitiva è più veloce (poiché non esegue che una valutazione parziale, veloce ed economica, dell'idoneità), è possibile sfruttarlo ed ottenere gli stessi risultati con un minore dispendio di risorse computazionali.

3.3. — Si assegni un valore di idoneità ad ogni genotipo in Γ come segue: si immagini di far diputare a tutti i genotipi in Γ un torneo all'italiana([3]) con il solo girone di andata; si denoti con $s(\gamma)$ il numero di vittorie ottenute dal genotipo γ dopo aver disputato tutti i suoi incontri, ovvero il suo punteggio (non si ammettono pareggi):

$$s(\gamma) = \sum_{\kappa \in \Gamma} I_{[\mathbf{winner}(\gamma,\kappa)=\gamma]}.$$

Naturalmente $s(\gamma)$ è una variabile aleatoria e dunque è possibile misurare l'idoneità di ogni $\gamma \in \Gamma$ con il valore atteso del suo punteggio, dato da:

$$f(\gamma) = E[s(\gamma)] = \sum_{\kappa \in \Gamma} P[\mathbf{winner}(\gamma, \kappa) = \gamma].$$

[3]) Un *torneo all'italiana* è caratterizzato dal fatto che ogni partecipante incontra tutti gli altri; qui, per semplificare i calcoli, si suppone che ogni partecipante incontri anche se stesso.

Chiameremo *a posteriori* questo tipo di idoneità assegnata sulla base dell'esito atteso delle competizioni.

Nel seguito si analizzano due casi, quello con idoneità non limitata superiormente e quello con indoneità compresa in un intervallo.

Idoneità illimitata.

3.4. — Si supponga che sia definita l'idoneità a priori e che possa essere calcolata per ogni individuo in Γ. In generale, è possibile che non si conosca il valore massimo da essa raggiungibile. Ciò equivale a dire che l'idoneità è finita ma superiormente illimitata.

Si desidera trovare una legge che esprima le probabilità degli esiti di una competizione in funzione delle idoneità a priori dei due avversari; inoltre, si richiede che questa legge sia *consistente* con la definizione di idoneità a posteriori data in §3.3.

In altri termini, si desidera trovare una funzione $p\colon \mathbb{R}^+ \times \mathbb{R}^+ \to [0,1]$ tale che, posto $P[\mathbf{winner}(\gamma,\kappa)=\gamma] = p(f(\gamma),f(\kappa))$, verifichi le seguenti condizioni:

1. $p(x,y) = 1 - p(y,x)$;
2. $p(0,y) = 0$, for $y > 0$;
3. $x < z$ implica $p(x,y) < p(z,y)$;
4. esiste una costante C tale che, per ogni $\gamma \in \Gamma$,

$$\sum_{\kappa\in\Gamma} p(f(\gamma),f(\kappa)) = Cf(\gamma).$$

3.5. — La prima condizione deriva dal fatto che esistono solo due esiti possibili per una competizione, e le probabilità che uno di essi si verifichi deve essere uguale alla probabilità che l'altro non si verifichi.

La seconda condizione stabilisce che un individuo con idoneità nulla non può concepibilmente vincere una competizione con un altro individuo avente idoneità non nulla. Quest'ipotesi è piuttosto intuitiva e non richiede ulteriori giustificazioni.

Anche la terza condizione è abbastanza intuitiva, dato che afferma che, di due individui, quello con idoneità più alta avrà più probabilità di vincere contro un terzo individuo fissato.

La Condizione 4 è la più importante e corrisponde al requisito di consistenza: le idoneità a priori e a posteriori devono essere proporzionali, cioè equivalenti a meno di una costante moltiplicativa.

Un'ipotesi ragionevole che ci permette di semplificare notevolmente il problema è che la funzione p dipenda in realtà dal rapporto tra i valori di idoneità dei due avversari, e non dal loro valore assoluto; si può allora riscrivere $p(x,y)$ come $p(x/y)$, ottenendo l'equazione funzionale

$$\begin{cases} p(x) = 1 - p(x^{-1}) \\ \qquad p(0) = 0. \end{cases} \tag{2}$$

3.6. — Soluzioni su $\mathbb{R}$ Questa equazione funzionale ha almeno una soluzione

monotona, continua e derivabile sull'intero asse reale; questa soluzione è

$$p(x) = \frac{2}{\pi} \arctan x.$$

3.7. — Soluzioni su $\mathbb{R}^+$. Restringendo però l'attenzione al semiasse positivo, che poi è quello che ci interessa dato che i valori di idoneità sono positivi per definizione, troviamo una famiglia di soluzioni monotone continue e derivabili, e precisamente

$$p(x) = 1 - \frac{1}{x^k + 1}, \tag{3}$$

per $k = 1, 2, \ldots$; le differenze di comporamento che queste soluzioni determinano se inserite nel modello di selezione basato sulla competizione saranno esaminate più avanti in §4. Un grafico di questa famiglia di soluzioni è dato in Figura III.1.

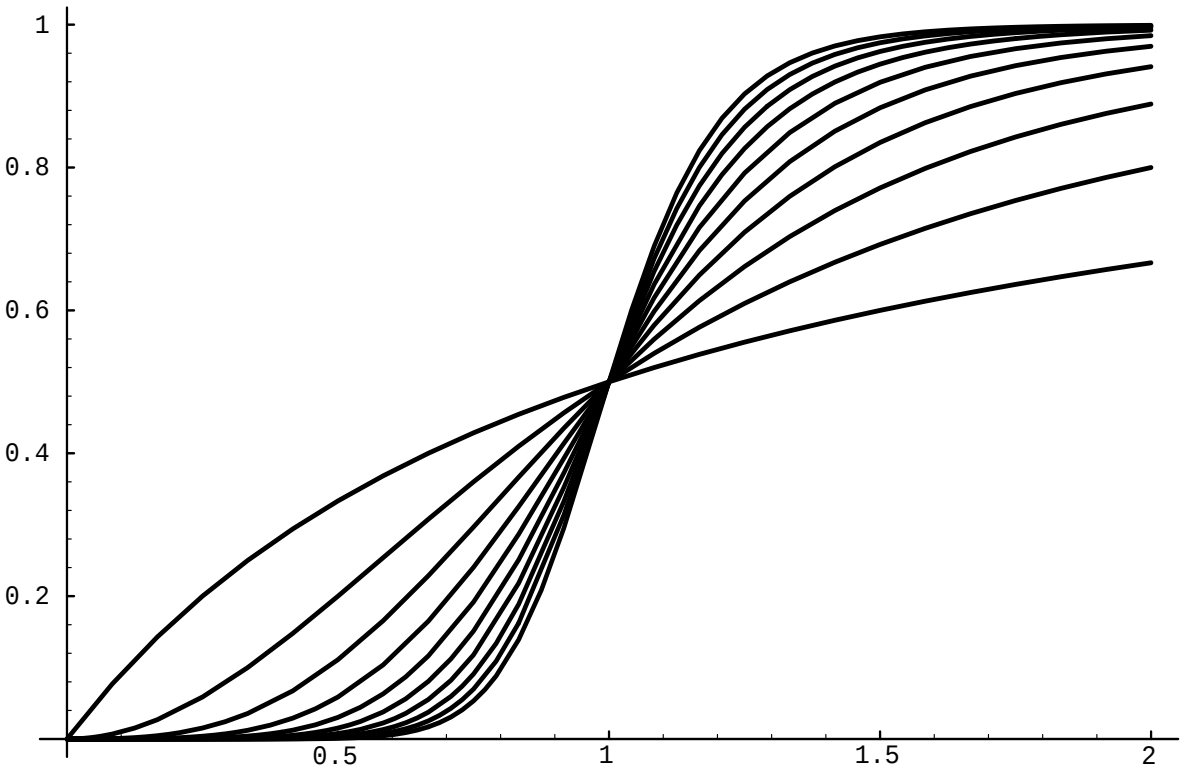

Figura III.1. — La famiglia di funzioni $p(x) = 1 - \frac{1}{x^k+1}$ per $k = 1, \ldots, 10$.

Si osservi che, per $k = 1$, l'Equazione (3) può essere riscritta come

$$p(x) = 1 - \frac{1}{x+1} = \frac{f(\gamma)}{f(\gamma) + f(\kappa)}.$$

Questa è una forma di selezione a torneo locale (cfr. Cap. II §10.7) in cui la selezione proporzionale all'idoneità è applicata a una popolazione di due individui γ e κ.

Al crescere di k, $p(x)$ concede progressivamente meno chances di vittoria all'avversario peggiore; nel caso limite, per $k = \infty$, la competizione diviene deterministica e l'avversario migliore vince sempre:

$$\lim_{k\to\infty} p(x) = I_{[x>1]} + \frac{1}{2} I_{[x=1]}. \tag{4}$$

Questo caso corrisponde alla selezione a torneo locale deterministica con due partecipanti.

Si verifica che ogni funzione p che sia soluzione dell'Equazione funzionale (2) soddisfa la Condizione 4 in 3.4. Per dimostrarlo, è necessario predisporre il seguente

3.8. — LEMMA Per ogni insieme di numeri reali positivi $x_1, x_2, \ldots, x_r$, $0 < x_i < +\infty$, $i = 1, 2, \ldots, r$, esiste una costante $0 < C < +\infty$ tale che, per tutti gli interi $1 \le n \le r$,

$$\sum_{i=1}^{r} p(x_n/x_i) = Cx_n,$$

dove p è una qualsiasi funzione che soddisfi l'Equazione funzionale (2).

3.9. — DIMOSTRAZIONE Per dimostrare che una tale costante C esiste, la calcoleremo esplicitamente. Scriviamo p_{ij} invece che $p(x_i/x_j)$ per alleggerire la notazione: abbiamo un sistema di r equazioni

$$\begin{aligned} Cx_1 &= p_{11} + p_{12} + \ldots + p_{1r} \\ Cx_2 &= p_{21} + p_{22} + \ldots + p_{2r} \\ &\vdots\vdots \\ Cx_r &= p_{r1} + p_{r2} + \ldots + p_{rr}. \end{aligned}$$

Prima di tutto osserviamo che $p_{ii} = 1/2$, per $i = 1, 2, \ldots, r$; inoltre sappiamo che $p_{ij} = 1 - p_{ji}$, per cui possiamo riscrivere tutti i p_{ij}, con $i > j$, come $1 - p_{ji}$, ottenendo

$$\begin{aligned} Cx_1 &= \frac{1}{2} + p_{12} + \ldots + p_{1r} \\ Cx_2 &= 1 - p_{12} + \frac{1}{2} + \ldots + p_{2r} \\ &\vdots\vdots \\ Cx_r &= 1 - p_{1r} + 1 - p_{2r} + \ldots + \frac{1}{2}. \end{aligned}$$

In generale,

$$Cx_n = (n-1) + \frac{1}{2} + p_{n,n+1} + \ldots + p_{nr} - p_{1n} - \ldots - p_{n-1,n}.$$

Ora, sommando tutte le r equazioni i termini di segno opposto si annullano e ne risulta

$$C(x_1 + x_2 + \ldots + x_r) = \sum_{n=2}^{r} (n-1) + \frac{1}{2} r = \frac{1}{2} r^2,$$

e perciò

$$C = \frac{\frac{1}{2}r^2}{x_1 + x_2 + \ldots + x_r},$$

che è la costante cercata. ∎

Sostituendo $f(\gamma)$, per ogni $\gamma \in \Gamma$, alle x_i si ottiene il seguente

3.10. — Teorema Sia $f\colon \Gamma \to \mathbb{R}^+$ tale che, per ogni $\gamma \in \Gamma$, $0 \leq f(\gamma) \leq +\infty$; sia $p\colon \mathbb{R}^+ \to [0,1]$ una qualsiasi soluzione dell' Equazione funzionale (2); allora esiste una costante $0 < C < +\infty$ tale che, per ogni $\gamma \in \Gamma$,

$$\sum_{\kappa \in \Gamma} p(f(\gamma)/f(\kappa)) = Cf(\gamma).$$

Idoneità superiormente limitata.

3.11. — A volte la natura del problema permette di sapere in anticipo il valore massimo che l'idoneità può raggiungere; in questo caso, anzi, ci si può sempre ridurre a valori dell'idoneità compresi nell'intervallo $[0,1]$. Con queste ipotesi, si arriva a condizioni più stringenti per la funzione p:

1. $p(x,y) = 1 - p(y,x)$;
2. $p(0,y) = 0$, for $y > 0$;
3. $p(1,y) = 1$, for $y < 1$;
4. esiste una costante C tale che, per ogni $\gamma \in \Gamma$,

$$\sum_{\kappa \in \Gamma} p(f(\gamma), f(\kappa)) = Cf(\gamma).$$

Occorre dunque risolvere la nuova equazione funzionale

$$\begin{cases} p(x,y) = 1 - p(y,x) & \\ p(0,y) = 0 & y > 0 \\ p(1,y) = 1 & y < 1 \end{cases} \tag{5}$$

3.12. — Si può sfruttare il fatto che sono già note soluzioni per la più semplice Equazione funzionale (2); ci si limiterà a cercare soluzioni su $\mathbb{R}^+$, dato che sono queste che interessano. Ci si vuole ricondurre a $p(x,y) = p(z(x,y))$, dove la p unaria è una soluzione all'Equazione funzionale (2): affinché $p(x,y)$ soddisfi l'Equazione funzionale (5), z deve essere una soluzione a

$$\begin{cases} z(x,y) = z(y,x)^{-1} & \\ z(0,y) = 0 & y > 0 \\ z(1,y) = +\infty & y < 1 \end{cases} \tag{6}$$

Porre semplicemente $z(x,y) = x/y$ non funzionerebbe, poiché violerebbe la condizione $z(1,y) = +\infty$ per $y < 1$, quindi occorre una trasformazione strettamente non decrescente t che mappi l'intervallo $[0,1)$ nell'intervallo $[0,+\infty)$, in modo tale che si possa scrivere la soluzione come

$$z(x,y) = t(x)/t(y).$$

Ora, t può essere qualsiasi funzione che soddisfi le tre condizioni

1. $t(0) = 0$,
2. $x < y$ implica $t(x) < t(y)$,
3. $\lim_{x\to 1} t(x) = +\infty$,

per ogni $x, y \in [0, 1)$. La più semplice funzione t siffatta è

$$t(x) = 1 - \frac{1}{x+1} = \frac{x}{x+1};$$

sostituendo e semplificando si ottiene allora

$$p(x, y) = \frac{x + xy}{x + 2xy + y}.$$

Questa soluzione è mostrata in Figura III.2.

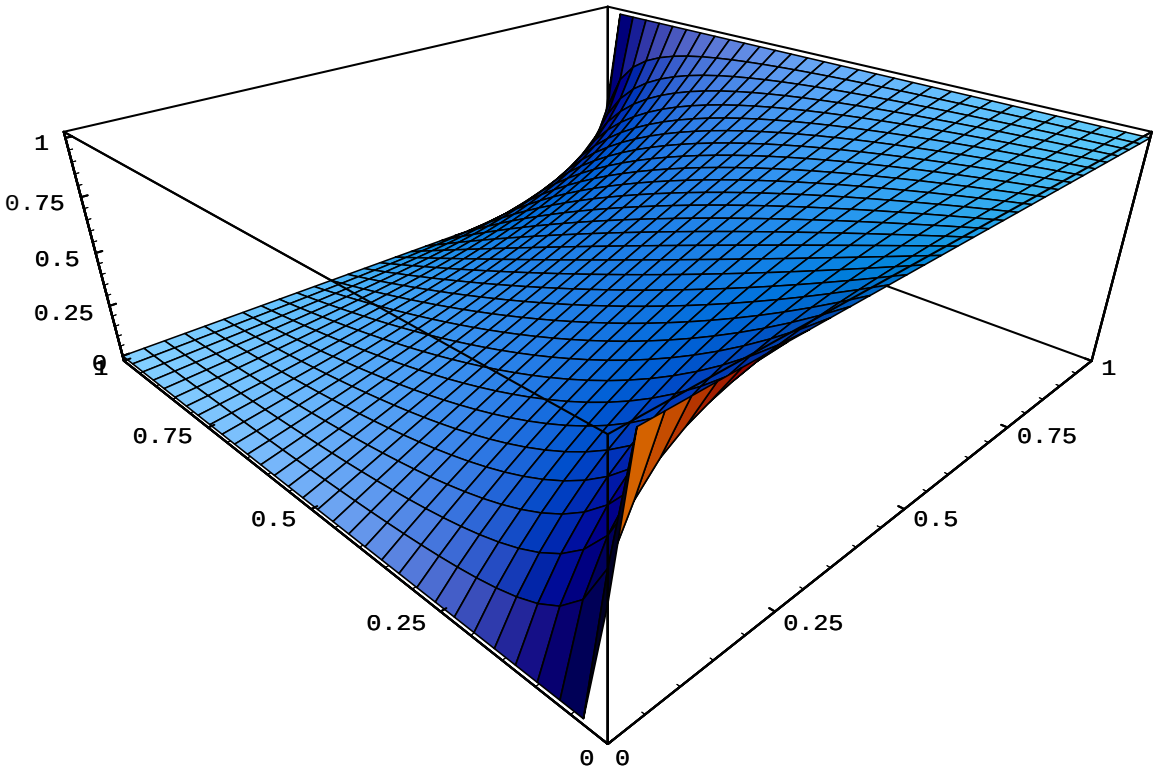

FIGURA III.2. — Un grafico tridimensionale della funzione $p(x, y) = \frac{x+xy}{x+2xy+y}$.

Per dimostrare che p soddisfa la Condizione 4, si invoca una versione leggermente modificata del Lemma 3.8:

3.13. — LEMMA Per ogni insieme di numeri reali $x_1, x_2, \ldots, x_r$, $0 \le x_i \le 1$, $i = 1, 2, \ldots, r$, esiste una costante $0 < C < +\infty$ tale che, per tutti gli interi $1 \le n \le r$,

$$\sum_{i=1}^{r} p(x_n, x_i) = Cx_n,$$

dove p è una qualsiasi funzione che soddisfi l'Equazione funzionale (5).

3.14. — Dimostrazione La dimostrazione è analoga a quella del Lemma 3.8. ■

3.15. — Teorema Sia $f\colon \Gamma \to [0,1]$ e sia $p\colon [0,1]\times[0,1] \to [0,1]$ una qualsiasi funzione che soddisfi l'Equazione funzionale (5); allora esiste una costante $0 < C < +\infty$ tale che, per ogni $\gamma \in \Gamma$,

$$\sum_{\kappa\in\Gamma} p(f(\gamma), f(\kappa)) = Cf(\gamma).$$

3.16. — Dimostrazione Si invoca il Lemma 3.13 sostituendo $f(\gamma)$, per ogni $\gamma \in \Gamma$, alle x_i. ■

§ 4. — Analisi della selezione competitiva.

Distribuzione uniforme — Distribuzione lineare — Altre distribuzioni.

Questo paragrafo studia il comportamento di una popolazione di individui soggetta alla selezione competitiva sulla base di diverse ipotesi riguardo alla distribuzione dei valori di idoneità.

4.1. — Va notato, tanto per cominciare, che la selezione competitiva si comporta in modo abbastanza diverso dalla classica selezione proporzionale all'idoneità per quanto riguarda il tasso di riproduzione degli individui. Il numero atteso di occorrenze di un genotipo γ nella popolazione X_{t+1} ottenuta dalla popolazione X_t per mezzo della selezione proporzionale all'idoneità si può scrivere

$$E_f[n_{X_{t+1}}(\gamma)] = \frac{nf(\gamma)n_{X_t}(\gamma)}{\sum_{\kappa\in\Gamma} n_{X_t}(\kappa)f(\kappa)}.$$

Si denoti questo valore atteso con l'indice f, per distinguerlo da quello relativo al modello basato sula competizione introdotto in §2.7, che è invece denotato con l'indice c.

4.2. — Osservazione È facile verificare che, data una popolazione X_{t+1} ottenuta da X_t applicando uno dei due modelli di selezione, per ogni $\gamma \in \Gamma$,

$$0 \le E_c[n_{X_{t+1}}(\gamma)] \le 2n_{X_t}(\gamma); \tag{6}$$

$$0 \le E_f[n_{X_{t+1}}(\gamma)] \le n. \tag{7}$$

Nella Disequazione (6), l'uguaglianza a destra vale quando $P[\mathbf{winner}(\gamma,\kappa) = \gamma] = 1$ per ogni $\kappa \in X_t$; questa situazione equivale, nel modello competitivo, a quando nella selezione proporzionale all'idoneità γ ha un'idoneità incomparabilmente più grande rispetto al resto degli individui della popolazione X_t, che è il caso in cui vale l'uguaglianza a destra nella Disequazione (7).

Naturalmente le uguaglianze a sinistra valgono in entrambe le disequazioni date sopra quando $P[\mathbf{winner}(\gamma, \kappa) = \gamma] = 0$ per ogni $\kappa \in X_t$ o, in modo equivalente, $f(\gamma) = 0$.

4.3. — Un indicatore statistico significativo che può essere studiato per caratterizzare il comportamento di un modello di selezione è quello che potremmo chiamare *tasso riproduttivo* di un genotipo, cioè il numero atteso di occorrenze di un genotipo γ nella generazione successiva diviso il numero di occorrenze nella popolazione corrente. Denotiamo questa quantità E_γ.

È noto (cfr. [54] per esempio) che sotto il regime di selezione proporzionale all'idoneità, per ogni popolazione $x \in \Gamma^*$,

$$E_\gamma = \frac{f(\gamma)}{f(x)} = \frac{nf(\gamma)}{\sum_{\kappa\in\Gamma} n_x(\kappa)},$$

per cui E_γ è lineare rispetto a $f(\gamma)$; concentreremo ora l'attenzione sull'analisi dell'andamento di E_γ rispetto a $f(\gamma)$ in un regime di selezione basata sulla competizione.

4.4. — Sia $\phi(v) = P_x[f(\gamma) = v]$ la funzione di probabilità dei valori di idoneità degli individui facenti parte della poplazione $x \in \Gamma^*$; da §2.7 sappiamo che

$$E_\gamma = \frac{2}{n} \sum_{\kappa\in\Gamma} n_x(\kappa) p(f(\gamma), f(\kappa)).$$

La stessa cosa può essere riscritta, utilizzando un integrale di Lebesgue, come

$$E_\gamma = 2 \int_0^\infty p(f(\gamma), v) d\phi(v). \tag{8}$$

Potremo quindi sostituire in questa espressione qualsiasi funzione p che rappresenti una soluzione alle Equazioni funzionali (2) o (5) insieme a una particolare distribuzione ϕ dei valori di idoneità e risolvere l'integrale per ottenere E_γ espresso in termini di $f(\gamma)$.

4.5. — Per esempio, scegliendo p all'interno della famiglia di soluzioni in §3.7, cioè

$$p(v, u) = 1 - \frac{1}{\frac{v^k}{u^k} + 1} = \frac{v^k}{v^k + u^k},$$

si ottiene

$$E_\gamma = 2 \int_0^\infty \frac{f(\gamma)^k d\phi(v)}{f(\gamma)^k + v^k} = 2 f(\gamma)^k \int_0^\infty \frac{d\phi(v)}{f(\gamma)^k + v^k}.$$

Il limite di questa espressione al tendere di k all'infinito, che dà il tasso riproduttivo di γ quando l'esito delle competizioni è deterministico, può essere calcolato ricordando l'Equazione (4):

$$\lim_{k\to\infty} 2 \int_0^\infty \frac{f(\gamma)^k d\phi(u)}{f(\gamma)^k + v^k} = \phi(f(\gamma)) + 2 \int_0^{f(\gamma)} d\phi(v). \tag{9}$$

Chiaramante una distribuzione dell'idoneità è sempre discreta, vuoi perché la popolazione ha cardinalità finita, vuoi perché, anche supponendo una cardinalità infinita per la popolazione, lo spazio dei genotipi Γ è finito, dato che si lavora con rappresentazioni che occupano una quantità di memoria finita su un elaboratore. Tuttavia, a patto che la cardinalità della popolazione n sia abbastanza grande, è possibile approssimare la distribuzione discreta con una funzione continua di densità di probabilità e quindi risolvere l'integrale per varie distribuzioni dell'idoneità. Nel seguito si sviluppano i calcoli necessari per alcuni semplici casi.

Distribuzione uniforme dell'idoneità.

4.6. — Si supponga che i valori dell'idoneità siano uniformemente distribuiti nell'intervallo $[0, b]$; allora

$$E_\gamma = \frac{2}{b} f(\gamma)^k \int_0^b \frac{dt}{f(\gamma)^k + t^k} = 2F\left(1, \frac{1}{k}; 1 + \frac{1}{k}; -\frac{b^k}{f(\gamma)^k}\right), \tag{10}$$

dove

$$F(a, b; c; z) = \sum_{i=0}^{\infty} \frac{(a)_i (b)_i z^i}{(c)_i i!}$$

è la funzione ipergeometrica, conosciuta anche sotto il nome di Serie di Gauss o Kummer, e $(a)_i = \Gamma(a + i)/\Gamma(a)$ è il simbolo di Pochhammer. Un grafico parametrico dell'andamento del tasso riproduttivo nel caso con distribuzione uniforme dell'idoneità è dato in Figura III.3.

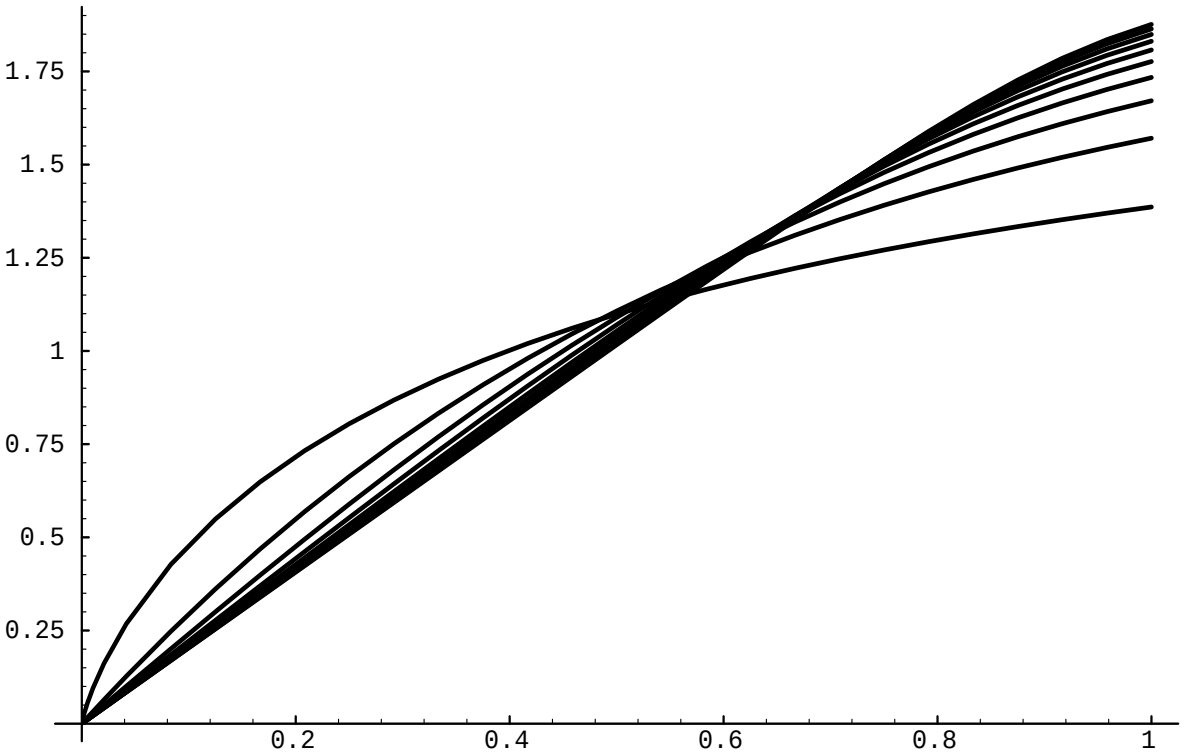

FIGURA III.3. — Grafico parametrico di E_γ nel caso con distribuzione uniforme dell'idoneità per $k = 1, \ldots, 10$.

4.7. — Per $k = 1$ l'Equazione (10) diventa

$$E_\gamma = \frac{2}{b} f(\gamma) \ln\left(1 + \frac{b}{f(\gamma)}\right).$$

Perciò $E_\gamma = 0$ per $f(\gamma) = 0$; per $f(\gamma) = b$, $E_\gamma = 2\ln 2 \sim 1.3863$; $E_\gamma = 1$, cioè ci si aspetta che il numero di occorrenze di γ rimanga stazionario, quando

$$f(\gamma) = \frac{b}{e^{\frac{b}{2}} - 1}.$$

Si osservi che questo valore è leggermente minore di $\frac{b}{2}$, il valor medio della distribuzione uniforme: la selezione competitiva appare generosa con gli individui di idoneità media e tende a favorirli un poco.

Per studiare che cosa succede a un individuo di idoneità media $\bar\gamma$, con $f(\bar\gamma) = E_\phi[f] = \frac{b}{2}$, sostituiamo $\frac{b}{2}$ a $f(\gamma)$ in (10); integrando si ottiene

$$E_{\bar\gamma}(k) = 2F\left(1, \frac{1}{k}; 1 + \frac{1}{k}; -2^k\right).$$

Analogamente siamo interessati a sapere che cosa succede al migliore individuo γ^* presente nella popolazione, con $f(\gamma^*) = b$: sostituendo in (10) e integrando, si ottiene

$$E_{\gamma^*}(k) = 2F\left(1, \frac{1}{k}; 1 + \frac{1}{k}; -1\right) = \frac{\psi\left(\frac{k+1}{2k}\right) - \psi\left(\frac{1}{2k}\right)}{k},$$

dove $\psi(z) = \Gamma'(z)/\Gamma(z)$ è la Funzione Digamma.

La tabella seguente elenca i valori di $E_{\bar{\gamma}}(k)$ e $E_{\gamma^*}(k)$ per $k = 1, \ldots, 10$:

k	$E_{\bar{\gamma}}(k)$	$E_{\gamma^*}(k)$
1	1.0986	1.3863
2	1.1071	1.5708
3	1.0900	1.6713
4	1.0702	1.7339
5	1.0536	1.7766
6	1.0410	1.8075
7	1.0318	1.8310
8	1.0250	1.8493
9	1.0202	1.8641
10	1.0164	1.8762

Si può osservare che $E_{\bar{\gamma}}(k)$ raggiunge un massimo a $k = 2$, per poi decrescere asintoticamente a 1 al crescere di k.

4.8. — Per $k = 2$,

$$E_\gamma = \frac{2}{b} f(\gamma) \arctan \frac{b}{f(\gamma)}.$$

Qui ancora $E_\gamma = 0$ for $f(\gamma) = 0$; per $f(\gamma) = b$, $E_\gamma = \frac{\pi}{2} \sim 1.5708$.

Al crescere di k, il grafico di E_γ rispetto a $f(\gamma)$ si appiattisce gradualmente e converge a una retta la cui equazione può essere calcolata dall'Equazione (9):

$$E_\gamma = 2 \int_0^{f(\gamma)} \frac{dt}{b} = \frac{2}{b} f(\gamma).$$

Distribuzione lineare dell'idoneità.

4.9. — Si supponga che $\phi(v) = \frac{2}{b}(1 - \frac{v}{b})$ per $v \in [0, b]$ (si può considerare questa come una approssimazione abbastanza rozza di una distribuzione esponenziale): allora

$$\begin{aligned} E_\gamma &= \frac{4}{b} f(\gamma)^k \int_0^b \frac{1 - t/b}{f(\gamma)^k + t^k} dt = \\ &= 4F\left(1, \frac{1}{k}; 1 + \frac{1}{k}; -\frac{b^k}{f(\gamma)^k}\right) - 2F\left(1, \frac{2}{k}; 1 + \frac{2}{k}; -\frac{b^k}{f(\gamma)^k}\right). \end{aligned} \tag{11}$$

Un grafico parametrico di questo andamento è mostrato in Figura III.4.

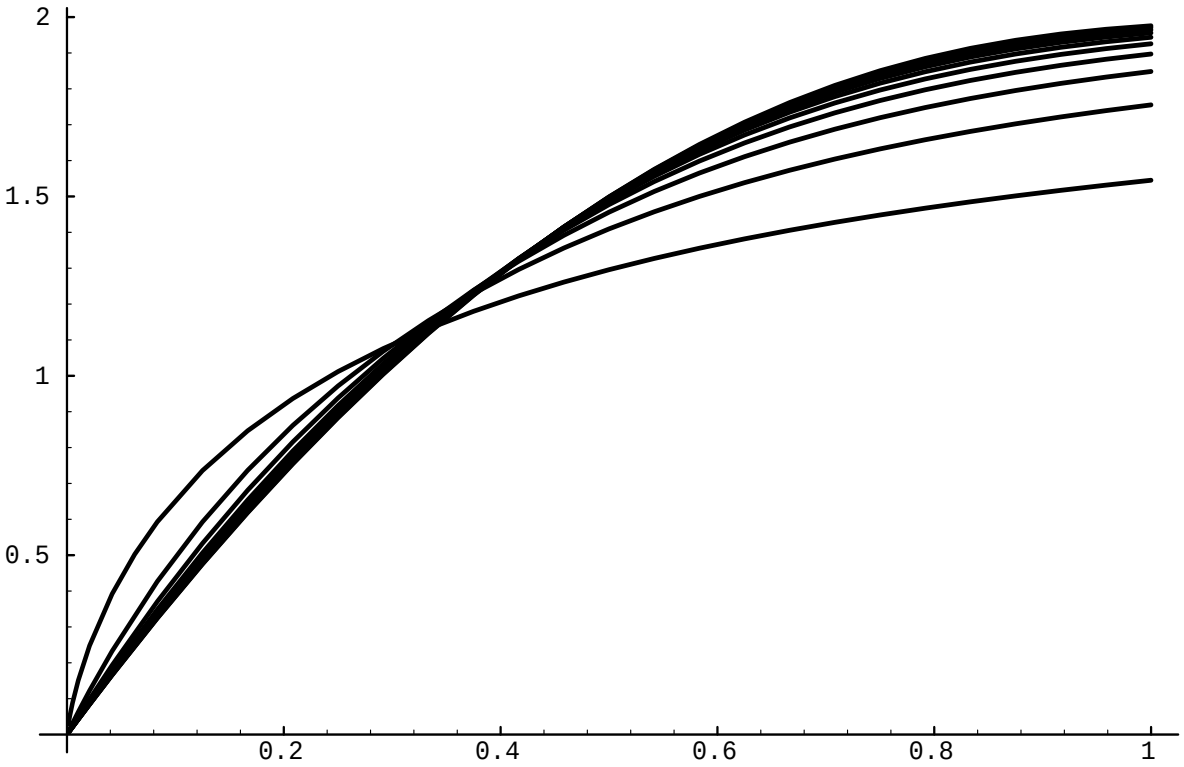

FIGURA III.4. — Grafico parametrico di E_γ nel caso con distribuzione lineare dell'idoneità, per $k = 1, \ldots, 10$.

4.10. — Qui $f(\bar{\gamma}) = E_\phi[f] = \frac{b}{3}$; sostituendo nell'equazione precedente e integrando si ottiene

$$E_{\bar{\gamma}}(k) = 4F\left(1, \frac{1}{k}; 1 + \frac{1}{k}; -3^k\right) - 2F\left(1, \frac{2}{k}; 1 + \frac{2}{k}; -3^k\right);$$

nello stesso modo,

$$E_{\gamma^*}(k) = \frac{2}{k}\left[\psi\left(\frac{k+1}{2k}\right) - \psi\left(\frac{k+2}{2k}\right) - \psi\left(\frac{1}{2k}\right) - \psi\left(\frac{1}{k}\right)\right].$$

La tabella seguente elenca i valori di $E_{\bar{\gamma}}(k)$ e $E_{\gamma^*}(k)$ per $k = 1, \ldots, 10$:

k	$E_{\bar{\gamma}}(k)$	$E_{\gamma^*}(k)$
1	1.1312	1.5452
2	1.1537	1.7553
3	1.1486	1.8484
4	1.1401	1.8971
5	1.1330	1.9256
6	1.1278	1.9438
7	1.1241	1.9560
8	1.1214	1.9647
9	1.1194	1.9710
10	1.1180	1.9758

Come nel caso con distribuzione uniforme, si osserva che $E_{\bar{\gamma}}(k)$ raggiunge un massimo a $k = 2$, per poi decrescere asintoticamente a $\frac{10}{9} = 1.1111\ldots$ al tendere di k all'infinito.

4.11. — Per $k = 1$ l'Equazione (11) diventa

$$E_\gamma = \frac{4}{b} f(\gamma) \left(\frac{f(\gamma) + b}{b} \ln \frac{f(\gamma) + b}{f(\gamma)} - 1 \right).$$

Quindi $E_\gamma = 0$ per $f(\gamma) = 0$; per $f(\gamma) = b$, $E_\gamma = 8 \ln 2 - 4 \sim 1.5452$.
Per $k = 2$,

$$E_\gamma = \frac{4}{b^2} f(\gamma)^2 \ln f(\gamma) + \frac{2}{b^2} f(\gamma) \left(2b \arctan \frac{b}{f(\gamma)} - f(\gamma) \ln(b^2 + f(\gamma)^2) \right).$$

Quindi $E_\gamma = 0$ per $f(\gamma) = 0$; per $f(\gamma) = b$, $E_\gamma = \pi - 2 \ln 2 \sim 1.7553$.

Al crescere di k il grafico converge lentamente a una curva quadratica convessa, la cui equazione si calcola dall'Equazione (9):

$$E_\gamma = \frac{4}{b} f(\gamma) - \frac{2}{b^2} f(\gamma)^2.$$

Altre distribuzioni dell'idoneità.

4.12. — Ovviamente le distribuzioni uniforme e lineare dell'idoneità studiate sopra sono puramente teoriche e raramente si incontrano nel corso della simulazione del processo evolutivo. Tuttavia esse ci hanno permesso di calcolare per via analitica gli integrali implicati; situazioni più realistiche possono essere investigate in modo numerico, per mezzo di simulazioni.

Infatti, la situazione tipica che può essere osservata durante la soluzione di diversi problemi di ottimizzazione mediante algoritmi evolutivi è caratterizzata da distribuzioni dell'idoneità più o meno a campana; in generale queste distribuzioni "reali" non sono necessariamente simmetriche né unimodali, ma ciononostante la loro forma complessiva si avvicina più a una campana che a un triangolo o a un rettangolo.

4.13. — La Figura III.5 mostra le curve empiriche ottenute mediante simulazione per una distribuzione dell'idoneità a campana con valor medio $f(\bar{\gamma}) = 10$ e funzione di densità di rpobabilità

$$\phi(t) = \frac{1}{\pi((t - 10)^2 + 1)}.$$

La scelta di questa particolare distribuzione non è stata dettata da altre considerazioni che la comodità di implementazione, dato che il suo unico scopo è semplicemente di fornire un'indicazione sommaria del comportamento di E_γ per una vasta classe di situazioni realistiche.

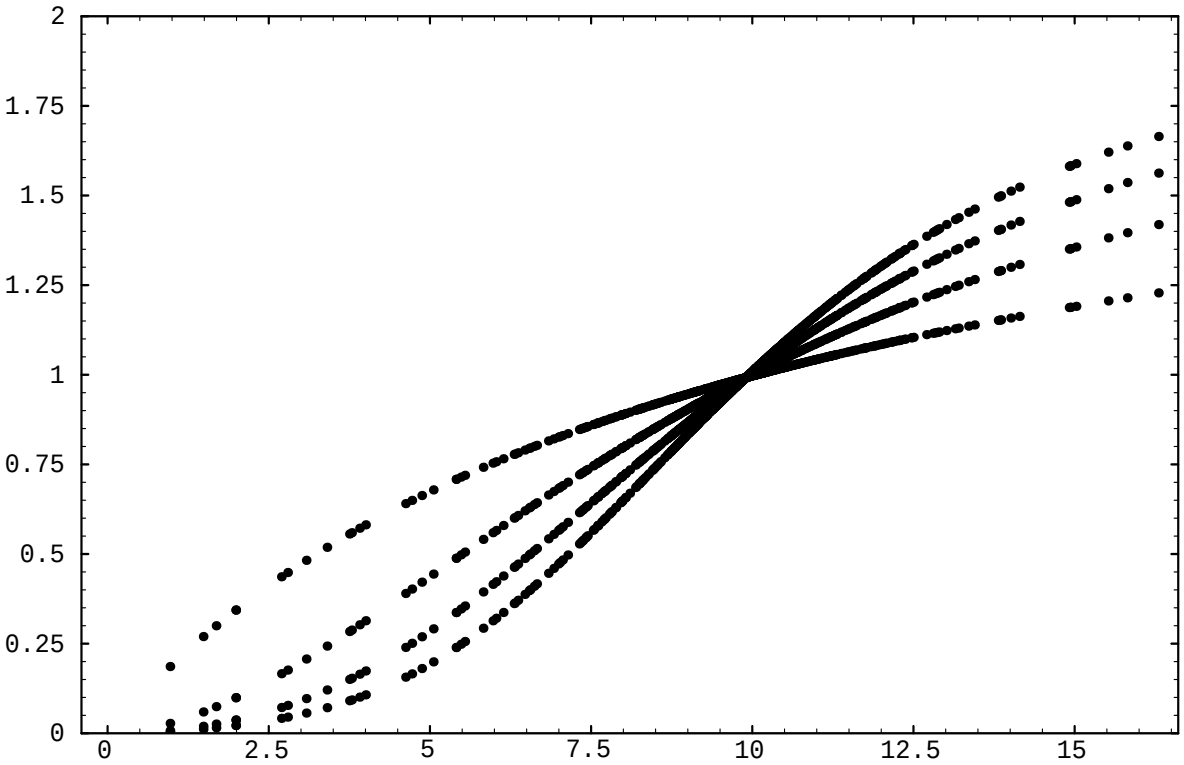

FIGURA III.5. — Grafico di E_γ in un caso con distribuzione dell'idoneità a campana, per $k = 1, \ldots, 4$, ottenuto mediante simulazioni con una popolazione di 500 individui.

È ancora possibile, tramite l'Equazione (9), ricavare il comportamento limite di questa distribuzione, la cui forma analitica è

$$E_\gamma = \frac{2}{\pi}(\arctan 10 - \arctan(10 - x)),$$

che corrisponde al grafico in Figura III.6: questo risultato è qualitativamente rappresentativo di una vasta classe di distribuzioni campaniformi o semicampaniformi dell'idoneità.

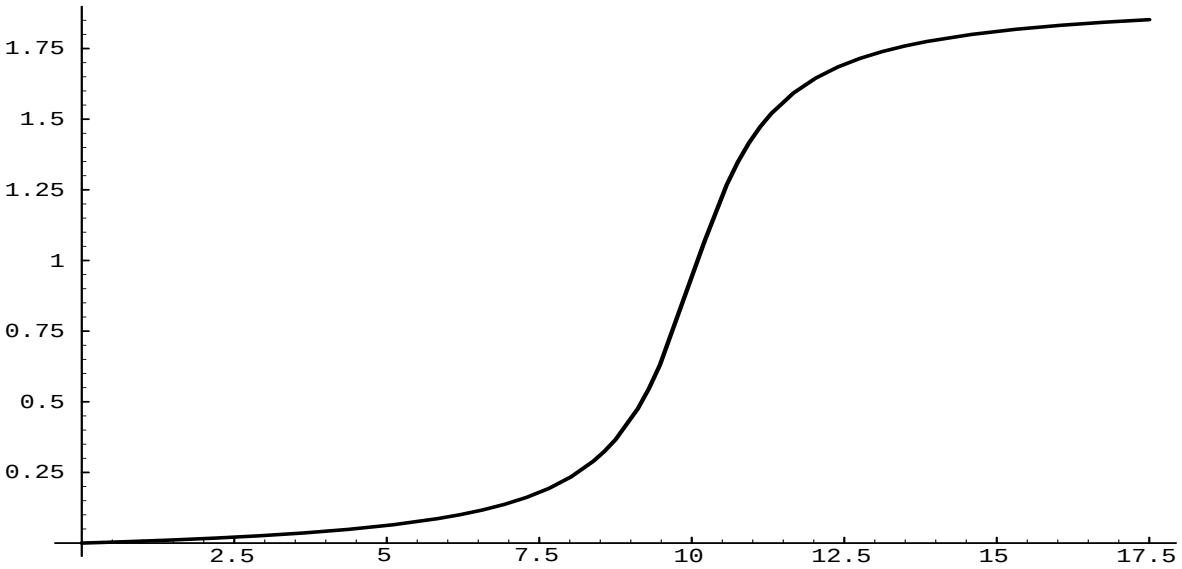

FIGURA III.6. — Andamento limite di E_γ per k tendente all'infinito nel caso con distribuzione dell'idoneità a campana con idoneità media $f(\bar{\gamma}) = 10$.

§ 5. — **Conclusioni.**

5.1. — Dall'analisi sviluppata nel paragrafo precedente, si evince che il comportamento della selezione competitiva dipende sia dalla scelta della funzione che collega la probabilità di vittoria all'idoneità a priori dei due avversari, sia dalla distribuzione dei valori dell'idoneità nella popolazione al momento della sua applicazione. In entrambi i casi, tuttavia, il numero atteso di occorrenze di un individuo sopra la media nella generazione successiva aumentano alle spese degli individui al di sotto della media; infatti, risulta esistere una lieve distorsione in favore degli individui nella media. Come si può osservare da Figura III.6, la selezione basata sulla competizione si avvicinerebbe, seppur in maniera confusa e al limite, a una selezione per troncamento (cfr. Cap. I §9.7 e segg.) con soglia $\tau = \frac{1}{2}$. Da esperimenti effettuati, pare che un algoritmo evolutivo che faccia uso della selezione competitiva abbia le migliori prestazioni quando il tasso di incrocio è relativamente basso, nell'ordine cioè del 20% o meno.

5.2. — La selezione competitiva non esibisce quella proprietà della selezione proporzionale all'idoneità che molti giudicano indesiderabile, per cui un superindividuo può soverchiare in poche generazione l'intera popolazione causando la convergenza prematura.

D'altro canto, nella parte bassa della distribuzione dell'idoneità, la selezione competitiva si comporta più o meno come la selezione proporzionale all'idoneità.

Capitolo IV

ALGORITMI EVOLUTIVI E LOGICA SFUMATA

Sebbene gli Algoritmi Evolutivi siano relativamente facili da implementare e costituiscano uno strumento potente per la soluzione di problemi complessi caratterizzati da spazi di ricerca di dimensioni enormi, essi richiedono di solito la supervisione di un essere umano per essere sfruttati con successo.

Sembra, a questo proposito, che tecniche ispirate alla Logica Sfumata possano contribuire a ridurre la quantità dell'intervento umano necessario per utilizzare un algoritmo evolutivo.

Questo capitolo si concentra su una particolare applicazione alla regressione simbolica per illustrare sulla scorta di un caso concreto come si possa costruire una base di conoscenza sfumata o, per usare un termine più suggestivo, un *governo fuzzy* che controlli un algoritmo evolutivo.

§ 1. — **Motivazioni.**

1.1. — Gli Algoritmi Evolutivi sono relativamente facili da implementare e, in generale, le loro prestazioni su problemi difficili si rivelano abbastanza soddisfacenti, soprattutto se confrontate con la piccola quantità di conoscenza specifica sul problema che deve essere da essi incorporata per funzionare. Tuttavia, tipicamente un algoritmo evolutivo richiederà, per così dire, della "supervisione umana", durante il suo utilizzo a regime come strumento pratico, per le seguenti ragioni:

- sebbene per alcuni problemi sia possibile riconoscere immediatamente una soluzione globalmente ottima non appena essa si presenti, in generale questo compito di riconoscimento è tutt'altro che banale, se non del tutto impossibile, e spesso esaminare in questo senso ogni soluzione candidata generata dall'algoritmo richiederebbe uno sforzo computazionale insostenibile;
- capita spesso che, per istanze diverse dello stesso problema, i parametri dello stesso identico algoritmo debbano essere regolati per ottenere la massima efficienza o persino per arrivare a una soluzione soddisfacente;
- infine, gli Algoritmi Evolutivi sono processi stocastici, cioè basati sul caso, e potrebbe sempre darsi che per qualche improbabile (e sfortunata) catena di coincidenze una singola esecuzione rimanga intrappolata in un ottimo locale: il supervisore umano allora normalmente si accorgerebbe che qualcosa non sta funzionando come previsto e reagirebbe a questa impressione arrestando e facendo ripartire il processo.

Il primo problema ha a che fare con il rilevamento dell'emergenza di una soluzione, il

secondo implica la regolazione di parametri da parte di un esperto e il terzo si occupa del monitoraggio del processo evolutivo.

1.2. — Qualsiasi tentativo di sviluppare uno strumento intelligente basato sugli Algoritmi Evolutivi dovrebbe tenere conto e affrontare queste questioni e un metodo per risolverle, se disponibile, sarebbe di certo benvenuto. Questa è la motivazione principale che ha fornito lo stimolo per investigare se e come l'intervento umano nell'uso degli Algoritmi Evolutivi possa essere ridotto se non, più ottimisticamente, eliminato.

In particolare si è concentrata l'attenzione su alcune tecniche di controllo basate sulla Teoria degli Insiemi Sfumati (*Fuzzy Set Theory*) [154, 156], in quanto esse consentono a un esperto umano (in questo caso lo sviluppatore dell'algoritmo) di incapsulare le sue conoscenze in qualche modo imprecise e qualitative sul sistema (in questo caso il processo evolutivo) in maniera semplice e naturale.

1.3. — Governo Fuzzy Si è dato il nome di *Governo Fuzzy* a quella collezione di regole e routine sfumate incaricate di controllare l'evoluzione di una popolazione, rilevare l'emergenza di una soluzione, regolare i parametri dell'algoritmo in modo dinamico (cfr. per esempio [86] e [12]) ed evitare comportamenti indesiderati come la convergenza prematura.

§ 2. — Un caso di studio: l'inferenza simbolica di formule.

Il problema — Struttura della Popolazione — La Connection Machine CM-200.

2.1. — Sebbene il suo obiettivo finale sia definire delle tecniche di applicabilità generale, questa ricerca ha preso le mosse da un particolare algoritmo evolutivo che era immediatamente disponibile e su cui si era lavorato per un tempo considerevole e sul quale per questo si era venuta formando una certa esperienza.

L'algoritmo in discussione è un algoritmo evolutivo ad elevato parallelismo per l'inferenza simbolica di formule induttive implementato su una Connection Machine CM-200. L'inferenza simbolica di formule induttive non è che un'istanza del problema più generale di inferire un'espressione analitica per una data funzione, il cui valore sia noto in un insieme di punti.

Il problema.

2.2. — In generale, il problema di inferire un'espressione analitica e in forma chiusa per una data funzione con dominio A e codominio B (per esempio potrebbe essere $A = \mathbb{R}^n$, $B = \mathbb{R}^m$) può essere formulato come segue: dato un insieme O di operatori o funzioni primitive, ciascuna con la sua arità, sia Φ una collezione di formule (4)

4) Tale collezione può essere finita, se si vincola la lunghezza massima delle formule,

comprendenti solo operatori in O, costanti ed n variabili $x_1, \ldots, x_n$ distinte. Ciascuna formula $\phi \in \Phi$ rappresenta una funzione analitica di n variabili nell'insieme A a valori nell'insieme B; usiamo lo stesso simbolo per denotare sia la funzione che la sua formula, in modo tale che si possa pensare a Φ come a un sottoinsieme dell'insieme di tutte le funzioni di n variabili

$$\Phi \subseteq \{f: A \to B\}.$$

Allora, se $\phi \in \Phi$, $\phi(\mathbf{x}) \in B$ è la funzione rappresentata da ϕ valutata in $\mathbf{x} \in A$.

2.3. — Si denoti con $\clubsuit: A \to B$ la funzione obiettivo; si supponga di essere in grado di valutare $\clubsuit$ in ogni punto di un dato sottoinsieme $S \subseteq A$ eseguendo un qualche esperimento o consultando una tabulazione, ma di ignorare la sua espressione analitica. Nel caso di un campione finito di dati, questo problema è chiamato *regressione simbolica* [84], in quanto si affronta sia la questione di decidere che tipo di funzione meglio spiega i dati, sia quella di trovare i coefficienti numerici che ne minimizzano l'errore.

Si desidera trovare una formula $\phi^* \in \Phi$ tale che, per ogni $\mathbf{x} \in S$,

$$\phi^*(\mathbf{x}) = \clubsuit(\mathbf{x}),$$

o, al minimo, nel caso in cui una tale ϕ^* non esista, una formula $\tilde{\phi} \in \Phi$ che minimizzi qualche opportuna distanza $d(\tilde{\phi}, \clubsuit)$.

2.4. — Nel seguito ci si concentra su un'istanza particolare del problema generale sopra descritto, specificamente quella di trovare formule induttive chiuse che siano soluzioni di alcuni casi speciali di equazioni di ricorrenza in una variabile.

Le equazioni considerate sono della forma:

$$\clubsuit(i) = \clubsuit(i-1) + h(i) \quad i = 0, 1, \ldots, \tag{1}$$

dove h è nota e $\clubsuit$ è la funzione incognita da inferire. Questa equazione può essere riscritta come

$$\clubsuit(n) = \sum_{i=1}^{n} h(i);$$

si supponga poi che $h(i)$ sia della forma i^k, con $k = 1, 2, \ldots$: in tal caso si ottiene la famiglia di funzioni

$$\clubsuit_k(n) = \sum_{i=1}^{n} i^k. \tag{2}$$

Questa equazione ha soluzioni (si veda per esempio [58], pag. 269)

$$\clubsuit_k(n) = \frac{1}{k+1} \sum_{i=0}^{k} \binom{k+1}{i} B_i n^{k-i+1}, \tag{3}$$

dove B_i è l'i-esimo coefficiente di Bernoulli.

---

o infinita, se si ammettono formule di lunghezza arbitraria ancorché finita.

2.5. — Questa famiglia di funzioni costituisce un conveniente caso di studio per un algoritmo evolutivo per la risoluzione di equazioni di ricorrenza per due ragioni principali:

- $\clubsuit_k(n)$ è facile da valutare per ogni intero positivo n per mezzo di una semplice procedura iterativa;
- poiché le soluzioni sono note per ogni k, è sempre possibile verificare la correttezza dei risultati ottenuti.

La scelta di questa particolare classe di funzioni come banco di prova per l'algoritmo evolutivo descritto di seguito non comporta in nessun modo che esso non possa essere in grado di affrontare con successo il problema più generale enunciato sopra; essa ha piuttosto a che fare con un'esigenza di rigore sperimentale. A conferma di ciò, nessuna caratteristica specifica della classe ristretta di funzioni viene sfruttata nel procedimento adottato.

Naturalmente, il problema specifico potrebbe essere risolto in maniera estremamente efficiente utilizzando un metodo ad hoc basato, per esempio, sulla manipolazione di funzioni generatrici [58]; tuttavia è interessante provare un approccio evolutivo, se non altro per le cose che si possono apprendere non solo su come risolvere questo problema particolare, o una sua variazione, ma sulla risoluzione di problemi in genere.

Struttura della popolazione.

2.6. — Sia $m = \|O\|$ il numero di operatori o funzioni primitive e si supponga, per semplicità, che tutti gli operatori siano binari. Sia L un insieme di simboli terminali, cioè costanti e variabili, con $\|L\| = n$.

Restringendo l'attenzione al caso di operatori binari, una formula di lunghezza $l = 2k+1$ può essere pensata come un albero con k nodi interni e $k+1$ foglie: esistono $k^m(k+1)^n$ alberi siffatti distinti. Il numero di formule ben formate di lunghezza minore o uguale a l, per $l = 1, 3, \ldots$, è

$$\|\Phi_l\| = \sum_{k=0}^{(l-1)/2} k^m(k+1)^n = O(l^{m+n}).$$

Si noti che non esistono formule ben formate di lunghezza pari. Il problema oggetto dunque è caratterizzato da uno spazio di ricerca enorme, la cui topologia non è ben studiata.

2.7. — Ora, sia F_l l'insieme di tutte le formule, ben formate o meno, di lunghezza non superiore a l; la cardinalità di tale insieme è

$$\|F_l\| = \sum_{k=1}^{l} (m+n)^k.$$

Ciò comporta che lo spazio di ricerca è *sparso:* per esempio, con $m = 4$ ed $n = 10$,

$$F_{21} \simeq 1.26 \cdot 10^{24} \quad \Phi_{21} \simeq 3.42 \cdot 10^{14},$$

cioè ci sono circa $3.68 \cdot 10^9$ formule illecite per ogni formula ammissibile. Questa considerazione suggerisce la progettazione di operatori genetici che preservino l'ammissibilità delle formule.

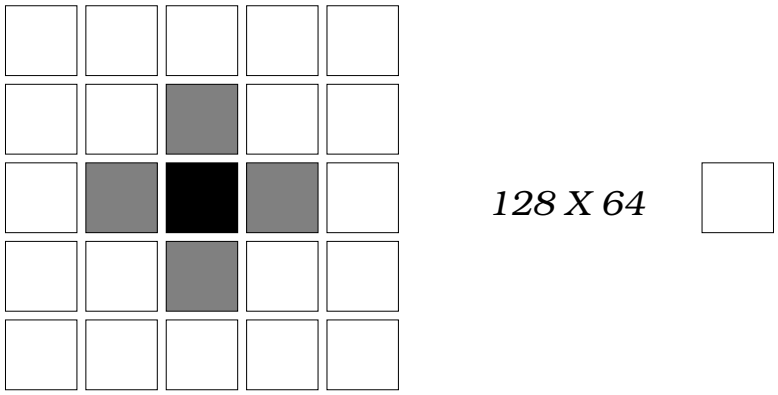

FIGURA IV.1. — Struttura della popolazione. I quadrati ombreggiati rappresentano gli immediati vicini dell'individuo posizionato nel quadrato nero.

2.8. — Durante una fase preliminare di esperimenti, si era osservato che, data una popolazione sufficientemente grande (nell'odine dei 1000 individui) di formule generate a caso, è molto probabile che una manciata di esse abbia un'idoneità molto superiore alla media. Strutturando la popolazione secondo un modello panmittico e utilizzando la selezione proporzionale all'idoneità, questi "superindividui" soverchierebbero abbastanza rapidamente l'intera popolazione.

Per questo motivo si è presentata l'opportunità di adottare una topologia apomittica per la popolazione, implementata distribuendo spazialmente gli individui su una griglia planare, come in [98, 99], restringendo la selezione e l'incrocio ai quattro immediati vicini di un individuo sulla griglia. Il massimo numero di copie che un superindividuo può riuscire a spandere nella popolazione in t generazioni è allora $2t^2 + 2t + 1$, mentre in una popolazione panmittica un singolo superindividuo potrebbe anche sopraffare tutti gli altri in una sola generazione.

Uno dei motivi per i quali l'algoritmo in questione è stato implementato su una CM-200 è che questa macchina supporta in modo naturale questo tipo di topologia.

La Connection Machine CM-200.

2.9. — La Connection Machine CM-200 è un elaboratore parallelo ad architettura a Singola Istruzione per Molti Dati (SIMD). Il tipo di elaborazione a cui questo tipo di architettura si indirizza è contraddistinto da grandi quantità di dati sui quali devono essere compiuti calcoli localizzati e facilmente parallelizzabili, come per esempio elaborazione delle immagini, simulazione di sistemi fisici di particelle, fluidodinamica, calcolo matriciale, reti neuronali e automi cellulari. Una Connection Machine è costituita da un gran numero (nell'ordine delle migliaia, da un minimo di 8192 a un massimo di 65536) di unità di elaborazione relativamente semplici e, prese da sole, abbastanza lente, dotate ciascuna di una piccola memoria privata e collegate tra di loro da una rete

di comunicazione a ipercubo. Un'unità di controllo centralizzata (*sequencer*) trasmette ad ogni ciclo l'istruzione da eseguire a tutte le unità di calcolo; queste la eseguiranno se attive o resteranno in attesa se momentaneamente disattivate.

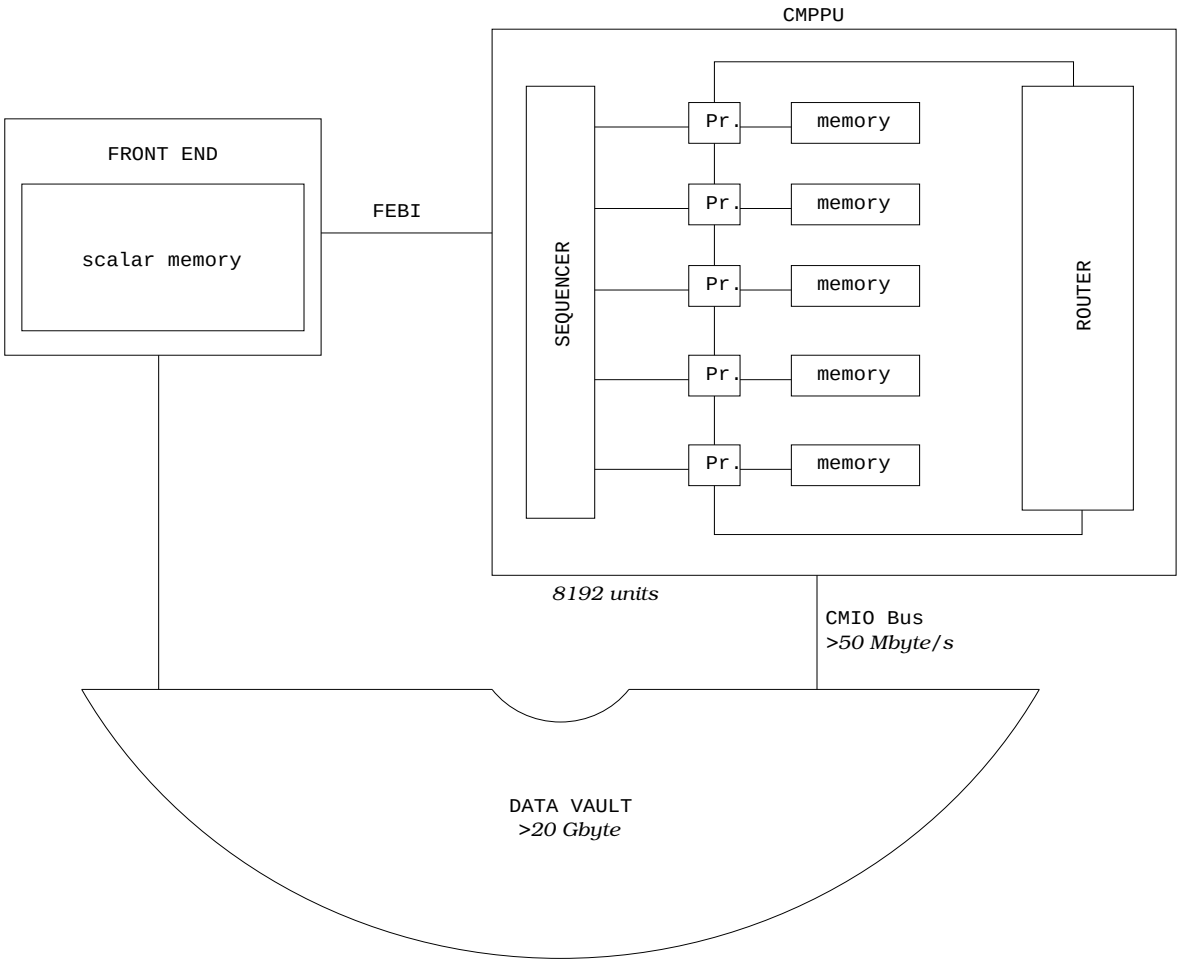

FIGURA IV.2. — Schema dei principali componenti di una Connection Machine CM-200.

Una descrizione dettagliata dell'hardware della CM-200 è oltre lo scopo di questo paragrafo, ma il lettore interessato può consultare [141].

2.10. — I dati vengono organizzati in speciali strutture dati "parallele" che possono essere pensate come matrici, i cui elementi sono allocati nelle memorie private delle unità di calcolo e designati con un unico nome. Ciascuna unità di calcolo esegue le istruzioni trasmesse dall'unità di controllo sugli elementi della matrice che risiedono nella propria memoria privata; in aggiunta a ciò, i dati possono essere passati da un'unità di calcolo all'altra attraverso la rete di connessioni. Il numero di elementi contenuti in una struttura dati parallela deve essere un multiplo del numero delle unità di calcolo disponibili, in modo tale che lo stesso numero di elementi possa essere assegnato alla memoria di ciascuna di esse. Le strutture dati scalari vengono mantenute nella memoria centrale del Front End, che è un elaboratore convenzionale, e sono visibili

ad ogni singola unità di calcolo.

§ 3. — Descrizione dell'algoritmo.

Codifica delle formule — Operatori genetici — Visualizzazione.

3.1. — La struttura complessiva dell'algoritmo può essere schematizzata come segue:

```
begin
    crea una popolazione iniziale;
    repeat
        riproduzione;
        incrocio;
        mutazione
    until condizione di terminazione;
end;
```

I genotipi sono immagazzinati sotto forma di stringhe di caratteri che rappresentano un'espressione in notazione polacca inversa. Nella versione originale dell'algoritmo la lunghezza massima dei genotipi poteva essere impostata dall'utente a seconda del probrema da risolvere; nella versione controllata da un governo fuzzy (cfr. §5.2) la lunghezza massima viene regolata automaticamente ad ogni generazione, partendo da un valore iniziale arbitrario di 70. Si osservi comunque che la memoria privata disponibile a ciascuna unità di calcolo della CM-200 consentirebbe genotipi di lunghezza fino a un massimo di più di 10000 caratteri.

La popolazione è disposta su una griglia 128×64 di processori, un individuo per ogni processore; la popolazione è perciò composta da 8192 individui. La disposizione a griglia si armonizza con l'impostazione di base della rete di comunicazione della CM-200.

La popolazione iniziale viene costituita da individui generati a caso con l'unico vincolo che l'espressione che essi rappresentano sia ben formata. La distribuzione di cui la popolazione iniziale è un campione è determinata da un parametro che influenza la lunghezza attesa dell'espressione generata.

Codifica delle formule.

3.2. — Un'espressione algebrica è rappresentata nel modo più naturale da un albero, i cui nodi interni contengono gli operatori o le funzioni primitive e le cui foglie sono variabili o costanti. Esempi di algoritmi evolutivi che utilizzano questo tipo di rappresentazione sono comuni in letteratura, specialmente nel campo della Programmazione Genetica (cfr. Cap. I, §7).

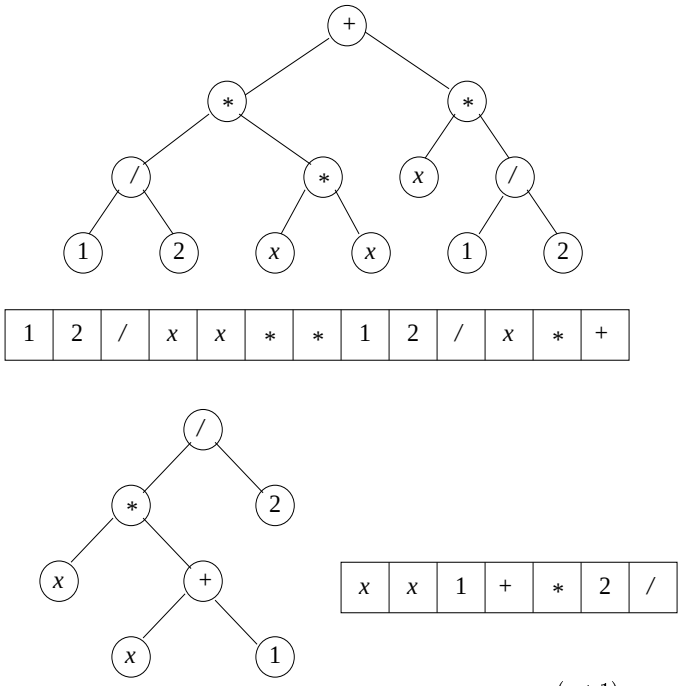

FIGURA IV.3. — Due esempi di codifica della formula $\frac{x(x+1)}{2}$ per mezzo di alberi e stringhe.

Una rappresentazione più efficiente fa uso di stringhe, i cui elementi possono essere variabili, costanti o operatori. Tali stringhe sono interpretabili come espressioni in notazione polacca inversa.

Nel caso in questione, un operatore può essere una delle quattro operazioni elementari, cioè $O = \{+, -, \times, /\}$. Si potrebbero prendere in considerazione anche altri operatori, ma questi quattro sono già sufficienti per costruire espressioni chiuse per tutte le formule induttive considerate.

3.3. — Mentre la rappresentazione mediante alberi è molto intuitiva e può essere gestita in modo dinamico durante il processo evolutivo, essa ha lo svantaggio di richiedere una quantità relativamente grande di memoria e un carico aggiuntivo di lavoro da parte del sistema, a causa dell'utilizzo di puntatori e di procedure di allocazione dinamica della memoria. D'altro canto la rappresentazione mediante stringhe richiede una quantità di memoria molto minore rispetto a quella che fa uso di alberi. Lo svantaggio di utilizzare una struttura dati statica è solo apparente; in realtà, l'hardware della CM-200 assegna ad ogni unità di calcolo una pari quantità di memoria, che non può essere sfruttata dalle altre in modo efficiente; ne segue che lo spazio non utilizzato da un genotipo corto non potrebbe comunque essere passato a un'altra unità per accomodare un genotipo più lungo.

Operatori genetici.

Sono stati implementati due operatori genetici, uno di mutazione ed uno di incrocio.

Per quanto riguarda l'incrocio, si è seguito fondamentalmente Koza [84]. Come al solito, due parametri, P_m e P_x, controllano il tasso di esecuzione della mutazione e dell'incrocio. Seguendo la letteratura [63], si sono condotti i primi esperimenti con $P_m = 0.05$ e $P_x = 0.6$; si è però osservato che, applicando il modello di selezione basato sulla competizione (cfr. Cap. III), un valore appropriato per la probabilità di incrocio è piuttosto $P_x = 0.1$.

A causa dell'architettura parallela della Connection Machine, il tempo necessario per eseguire il segmento di codice che effettua la mutazione e l'incrocio sulla popolazione nel suo complesso non dipende dai valori particolari assunti dalle probabilità P_m and P_x.

3.4. — Mutazione Prima di effettuare la mutazione, tutte le unità di calcolo decidono, indipendentemente e con la stessa probabilità P_m, se procedere o attendere che le altre abbiano completato l'operazione. Nel primo caso, un simbolo del genotipo viene selezionato a caso con probabilità uniforme e la sottostringa corrispondente al sottoalbero avente il simbolo scelto come radice viene sostituito da una nuova sottostringa generata a caso in modo da rappresentare una formula ben formata.

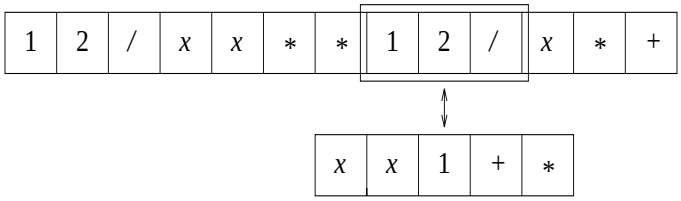

Figura IV.4. — Esempio di mutazione.

L'inserimento di una sottoespressione casuale introduce nuovo materiale genetico che si aggiunge a quello già presente all'interno della popolazione.

3.5. — Incrocio L'incrocio sostituisce una sottostringa del primo individuo genitore con una sottostringa estratta dal secondo: un simbolo viene selezionato a caso con probabilità uniforme in entrambi i genotipi; quindi le sottostringhe corrispondenti ai sottoalberi aventi i simboli scelti come radici vengono estratti e il primo viene sostituito con il secondo. Si noti che le due sottostringhe che partecipano allo scambio non sono in generale della stessa lunghezza.

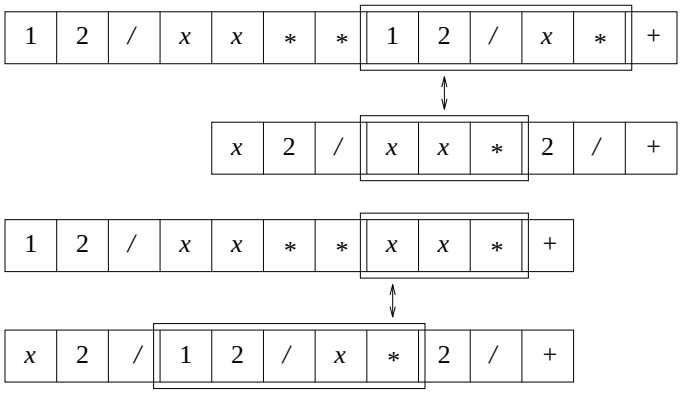

FIGURA IV.5. — Esempio di incrocio.

3.6. — SELEZIONE La selezione adottata è quella basata sulla competizione introdotta in [138] e descritta nel Cap. III. Ad ogni generazione, ciascun processore sceglie a caso con probabilità uniforme uno dei suoi quattro vicini immediati sulla griglia bidimensionale: si chiami γ il genotipo che risiede nel processore centrale e κ quello che risiede nel vicino scelto; i due genotipi vengono fatti competere tra di loro e, se $\mathbf{winner}(\gamma, \kappa) = \kappa$, questo prende il posto di γ nel processore centrale. Tutte le competizioni vengono disputate in parallelo e indipendentemente l'una dall'altra. Lo pseudocodice seguente illustra il funzionamento del modello in questo algoritmo:

for each processor p **do in parallel**
  begin
    scegliere un vicino q a caso;
    $genotype[p] :=$ **winner**$(genotype[p], genotype[q])$
  end;

3.7. — Nell'algoritmo in questione, $\mathbf{winner}(\gamma, \kappa)$ è definito come segue: sia $X \in N$ una variabile aleatoria a valori interi con distribuzione uniforme sull'insieme $\{1, 2, \ldots, m\}$, dove il valore di m dipende dall'istanza particolare del problema; [5] allora

$$\mathbf{winner}(\gamma, \kappa) = \begin{cases} \gamma & \text{if } |\gamma(X) - \clubsuit(X)| < |\kappa(X) - \clubsuit(X)| \\ \kappa & \text{if } |\gamma(X) - \clubsuit(X)| < |\kappa(X) - \clubsuit(X)| \end{cases}$$

In caso di pareggio, cioè quando $|\gamma(X) - \clubsuit(X)| = |\kappa(X) - \clubsuit(X)|$, vince l'individuo con il genotipo più corto. Questo è un modo per introdurre una pressione selettiva nella direzione di soluzioni che siano più veloci da calcolare, aumentando in questo modo l'efficienza dell'algoritmo. Se γ e κ sono anche uguali in lunghezza, quello che "gioca in trasferta" perde: in altre parole, se due individui sono indistinguibili, non c'è motivo di fare una copia.

Visualizzazione.

[5]) m era impostato dall'utente nella versione originale ed è ora controllato dinamicamente dal governo fuzzy (cfr. §5.10).

3.8. — Quella della visualizzazione è una questione centrale nell'algoritmo descritto, proncipalmente a causa dell'enorme quantità di dati e della necessità di fornirli in una forma concisa, chiara e leggibile.

Inoltre, andrebbe sottolineato il fatto che quasi tutto ciò che si è appreso sul funzionamento dell'algoritmo, lo si è appreso attraverso l'esame dai prospetti riassuntivi generati automaticamente in un gran numero di esecuzioni sperimentali. Per questa ragione una visualizzazione e una generazione di prospetti riassuntivi esaurienti ma tersi è sono un passo vitale verso l'implementazione di un governo fuzzy per un algoritmo evolutivo.

La quantità di dati associati ad ogni generazione complica lo sviluppo di una buona procedura di generazione di prospetti riassuntivi. Poiché è impossibile visualizzare tutti i dati, è necessario selezionarne un sottoinsieme piccolo ma significativo, che catturi la situazione in un dato istante. La dimensione di questo sottoinsieme è vincolata dall'esigenza di mantenere il costo computazionale della generazione del prospetto riassuntivo al di sotto di una percentuale trascurabile del costo totale.

3.9. — Tra i problemi di visualizzazione che si sono dovuti risolvere per implementare la procedura che genera i prospetti riassuntivi, il più importante è stato rappresentato dalla difficoltà di rendere l'espressione codificata da un genotipo nella forma più leggibile. La stringa genotipica, in notazione polacca inversa, deve essere tradotta nell'usuale notazione infissa e quindi deve essere semplificata e ridotta in forma normale. La semplificazione è necessaria perché spesso la sola traduzione produce espressioni caratterizzate da un elevato grado di ridondanza.

Tenendo in mente l'esigenza di mantenere il carico della semplificazione a un livello minimo, si è scelto di realizzare una procedura semplice ma efficace che trova un'espressione irriducibile a partire da ogni formula. Questo compito è svolto scandendo la stringa genotipica e considerando ciascun elemento come una frazione di polinomi: ciò permette di trattare la divisione e la moltiplicazione in maniera omogenea.

3.10. — Un'altra questione chiave riguarda la scelta degli individui da visualizzare. Una scelta naturale cadrebbe sull'individuo *migliore* all'interno della popolazione; tuttavia l'algoritmo, adottando la selezione competitiva, fa a meno di un'idoneità esplicita associata a ciascun individuo, per cui individuare il migliore è un compito non banale. In particolare, l'individuo che esibisce il più piccolo errore relativo nell'ultima valutazione potrebbe benissimo essere peggiore in media di altri individui con un errore lievemente maggiore.

3.11. — La selezione dell'individuo da visualizzare viene effettuata nel modo seguente: si denoti con $T(i,j)$ l'"età" dell'individuo che occupa la posizione (i,j) nella griglia, cioè il numero di generazioni dalla sua ultima modifica, con $B(i,j)$ il numero di volte in cui esso ha ottenuto il più piccolo errore relativo nelle ultime $T(i,j)$ generazioni e con $\bar{e}(i,j)$ il suo errore relativo mediato sulle ultime $T(i,j)$ generazioni: allora l'individuo da visualizzare è scelto a caso dall'inisieme ottenuto estraendo dalla popolazione gli individui che massimizzano $B(i,j)$, quindi da essi quelli di lunghezza minima che minimizzano $\bar{e}(i,j)$.

§ 4. — Elementi di Teoria degli Insiemi Sfumati.

Definizioni — Il Principio di Estensione — Relazioni sfumate — Logica Sfumata.

La Teoria degli Insiemi Sfumati è una teoria matematica dell'imprecisione, proposta nel 1965 da Lotfi A. Zadeh [153]. Questo paragrafo ne introduce in maniera succinta i concetti fondamentali.

Definizioni.

4.1. — Definizione Sia X un universo di oggetti e si denoti con x un generico elemento di X. Un *insieme sfumato* A in X è definito da una funzione (caratteristica) di appartenenza $f_A(x)$ che ad ogni oggetto in X associa un numero reale nell'intervallo $[0, 1]$. Il valore $f_A(x)$ rappresenta il "grado di appartenenza" di x ad A. Se A è un insieme nel senso convenzionale del termine, f_A si riduce all'usuale funzione caratteristica di A.

Si noti che, sebbene la funzione di appartenenza di un insieme sfumato sia effettivamente una misura, e quindi possa essere percepita come simile a una misura di probabilità, esistono delle differenze essenziali tra questi due concetti. Infatti la nozione di insieme sfumato è di natura completamente deterministica.

4.2. — Definizione Due insiemi sfumati A e B sono uguali, $A = B$, se e solo se, per ogni $x \in X$, $f_A(x) = f_B(x)$.

Il *complemento* di un insieme sfumato A è l'insieme sfumato $\bar{A}$ avente funzione di appartenenza, per ogni $x \in X$, $f_{\bar{A}}(x) = 1 - f_A(x)$.

A è *contenuto* in B (è un sottoinsieme sfumato di B, ovvero è minore o uguale a B), $A \subseteq B$, se e solo se, per ogni $x \in X$, $f_A(x) \leq f_B(x)$.

4.3. — Definizione L'*unione* di due insiemi sfumati A e B è l'insieme sfumato $C = A \cup B$ avente funzione di appartenenza, per ogni $x \in X$,

$$f_C(x) = \max\{f_A(x), f_B(x)\}.$$

In modo equivalente ma più intuitivo, l'unione di A e B è il più piccolo insieme sfumato che contiene sia A che B.

Analogamente, l'*intersezione* di due insiemi sfumati A e B è l'insieme sfumato $D = A \cap B$ avente funzione di appartenenza, per ogni $x \in X$,

$$f_D(x) = \min\{f_A(x), f_B(x)\}.$$

Come per gli insiemi convenzionali, si dice che due insiemi sfumati A e B sono *disgiunti* se la loro intersezione è vuota, cioè se, per ogni $x \in X$, $f_{A\cap B}(x) = 0$.

4.4. — Si dimostra facilmente che gli insiemi sfumati in X con le operazioni di unione, intersezione e complemento costituiscono un reticolo distributivo completo.

Valgono dunque le estensioni delle proprietà fondamentali degli insiemi convenzionali, come le Leggi di De Morgan, le Leggi distributive, ecc.

Una notevole eccezione è costituita dalla Legge del terzo escluso, che non vale in generale per gli insiemi sfumati: infatti, per ogni $x \in X$, $f_{A\cap\bar{A}}(x) \geq 0$. In altri termini, l'intersezione di un insieme sfumato con il suo complemento è, in generale, non vuota; è vuota solo nel caso particolare in cui A sia un insieme convenzionale.

Il Principio di estensione.

4.5. — DEFINIZIONE Sia X il prodotto cartesiano di n universi $X_1, \ldots, X_n$ e siano $A_1, \ldots, A_n$ altrettanti insiemi sfumati definiti rispettivamente in $X_1, \ldots, X_n$.

Se $t: X \to Y$ è un morfismo da X in un nuovo universo Y, l'insieme sfumato B, indotto in Y dal morfismo t a partire dagli insiemi sfumati $A_1, \ldots, A_n$, ha funzione di appartenenza, per ogni $y \in Y$,

$$f_B(y) = \sup_{(x_1,\ldots,x_n)\in t^{-1}(y)} \min\{f_{A_1}(x_1), \ldots, f_{A_n}(x_n)\}.$$

Si dice che B estende gli insiemi sfumati $A_1, \ldots, A_n$ in Y e questa definizione è nota come *Principio di estensione* per gli insiemi sfumati.

Relazioni sfumate.

4.6. — DEFINIZIONE Una *relazione sfumata* in X è un insieme sfumato sullo spazio prodotto $X \times X$. Più in generale, si definisce una relazione sfumata n-aria in X come un insieme sfumato R sullo spazio prodotto X^n. In tal caso, la funzione di appartenenza di R è della forma $f_R(x_1, \ldots, x_n)$, con $x_i \in X$, $i = 1, \ldots, n$.

4.7. — DEFINIZIONE Siano $R_1 \subseteq X \times Y$ ed $R_2 \subseteq Y \times Z$ due relazioni sfumate. La composizione di R_1 ed R_2, $R_1 \circ R_2$, ha funzione di appartenenza, per ogni $x \in X$, $z \in Z$,

$$f_{R_1\circ R_2}(x, z) = \sup_{y\in Y} \min\{f_{R_1}(x, y), f_{R_2}(y, z)\}.$$

Logica Sfumata.

4.8. — Dato un insieme sfumato A in un universo del discorso X, si può definire il *predicato sfumato* ad esso associato P_A, il cui valore di verità su un elemento $x \in X$ è dato da $f_A(x)$. Il predicato $P_A(x)$ può anche essere scritto nella forma $x \in A$. Si ottiene in questo modo un sistema logico, noto con il nome di *Logica Sfumata* (*Fuzzy Logic* in inglese), che si prefigge di formalizzare i meccanismi del ragionamento approssimativo. Come tale, esso si situa nel campo della logica a più valori, sebbene le sue motivazioni

si discostino significativamente da altri sistemi dello stesso tipo, quali, per esempio, la Logica di Lukasiewicz.

A questo proposito, vale la pena di accennare al fatto che molti dei concetti responsabili del successo della Logica Sfumata come logica del ragionamento approssimativo non fanno parte dei sistemi logici a più valori tradizionali. Tra questi si annoverano i concetti di *variabile linguistica,* di *forma canonica,* di *regola sfumata* e *quantificatore sfumato*, nonché metodi di inferenza come il ragionamento *interpolativo, sillogistico* e *disposizionale.*

4.9. — Strettamente connesso con quello di relazione sfumata è il concetto di *regola sfumata.* Una regola sfumata è un costrutto sintattico della forma

SE *antecedente* ALLORA *conseguente*,

in cui *antecedente* e *conseguente* sono formule ben formate in Logica Sfumata.

Le regole sfumate offrono un modo alternativo, compatto e potente, di esprimere dipendenze funzionali tra vari elementi di un sistema in modo modulare e, soprattutto, intuitivo. Come tali, esse hanno trovato vaste applicazioni pratiche, per esempio nel campo della controllistica e dei sistemi diagnostici.

4.10. — La semantica di un sistema di regole sfumate è governata dalle leggi del Calcolo delle regole sfumate [155]. In estrema sintesi, tutte le regole di un insieme di regole sfumate partecipano simultaneamente al processo di inferenza, ciascuna in misura proporzionale al valore di verità associato con il suo antecedente. Il risultato di un'inferenza è rappresentato da un insieme sfumato per ciascuna delle variabili dipendenti. Il grado di appartenenza di un valore di una variabile dipendente all'insieme sfumato associato dà una misura della sua compatibilità con i valori osservati delle variabili indipendenti.

Sia dato un sistema con n variabili indipendent $x_1, \ldots, x_n$ ed m variabili dipendenti $y_1, \ldots, y_m$; sia R l'insieme di r regole sfumate

$$\begin{array}{ll} \text{SE } P_1(x_1, \ldots, x_n) & \text{ALLORA } Q_1(y_1, \ldots, y_m), \\ \vdots & \vdots \\ \text{SE } P_r(x_1, \ldots, x_n) & \text{ALLORA } Q_r(y_1, \ldots, y_m), \end{array}$$

dove $P_1, \ldots, P_r$ e $Q_1, \ldots Q_r$ rappresentano predicati sfumati sulle variabili rispettivamente indipendenti e dipendenti, e si denoti con f_P il valore di verità di un predicato P. Allora la funzione di appartenenza che decrive l'insieme sfumato dei valori assunti dalle variabili dipendenti $y_1, \ldots, y_m$ del sistema R è data da

$$f_R(y_1, \ldots, y_m; x_1, \ldots, x_n) = \sup_{1 \le i \le r} \min\{f_{Q_i}(y_1, \ldots, y_m), f_{P_i}(x_1, \ldots, x_n)\}.$$

§ 5. — Il Governo Fuzzy.

Regolazione della lunghezza massima — Regolazione del tasso di mutazione — Regolazione dello intervallo di valutazione — Rilevamento dell'emergenza.

5.1. — Sono stati affrontati quattro problemi concernenti il controllo del processo evolutivo nell'algoritmo descritto in §3:

1. regolazione dinamica della lunghezza massima dei genotipi al fine di rendere l'algoritmo indipendente dall'istanza del problema e, nello stesso tempo, di guadagnare in velocità, dato che genotipi più corti sono più rapidi da valutare;
2. intervento sul tasso di mutazione in modo tale da approfittare dei fenomeni di convergenza prematura per velocizzare l'ottimizzazione;
3. regolazione dell'intervallo da cui vengono estratti i punti usati per valutare gli individui durante la competizione, al fine di scongiurare le eccezioni dovute ad *underflow* ed *overflow* durante la valutazione delle formule;
4. rilevamento dell'emergenza di una soluzione e arresto del processo evolutivo solo una volta che sia evidente che essa rappresenta un ottimo globale.

Regolazione della lunghezza massima.

5.2. — Un'importante limitazione hardware delle macchine SIMD, come la Connection Machine, deriva dal fatto che tutte le unità di elaborazione devono essere sempre sincronizzate. Per un algoritmo come quello descritto ciò implica che durante la valutazione dei genotipi l'esecuzione non può continuare finché l'ultimo processore non ha finito di valutare il suo individuo: in altre parole, l'elaborazione procede alla velocità del processo più lento. Dato che il tempo necessario per valutare un individuo è direttamente proporzionale alla lunghezza del suo genotipo, diviene evidente il vantaggio di limitare la lunghezza massima consentita, seppure in una misura che non si riveli controproducente.

Purtroppo, il valore ottimale di questo parametro dipende non solo dalla particolare istanza del problema, ma anche dallo stadio in cui si trova la ricerca della soluzione. Per esempio, la fase iniziale della ricerca non necessita di genotipi lunghi ma, con il procedere dell'evoluzione e con l'apparire di selezioni più raffinate, diventa necessario lasciare che si formino genotipi più lunghi (e quindi più complessi).

Per questo motivo non è possibile determinare una lunghezza massima una volta per tutte le istanze del problema. Nella versione dell'algoritmo senza governo fuzzy il supervisore umano doveva semplicemente scegliere un valore appropriato seguendo qualche regola empirica prima di lanciare un'esecuzione; questo compito è ora assunto dal governo fuzzy, che per giunta continua a regolare il parametro in tutto il corso dell'evoluzione.

5.3. — La lunghezza massima L_{t+1} ammissibile alla generazione $t+1$ è posta inizialmente uguale a L_t, e viene quindi modificata prendendo in considerazione la lunghezza media dei genotipi, $\bar{l}_t$, e la lunghezza l_t^* del genotipo migliore (secondo il criterio illustrato in §3.11). Le regole sfumate hanno la forma seguente:

Regola L1.

$$\begin{array}{ll} \text{SE} & (l_t^* - \bar{l}_t) \in P \wedge (L_t - l_t^*) \in P \\ \text{ALLORA} & (L_{t+1} - L_t) \in AP \end{array}$$

Regola L2.

$$\begin{array}{ll} \text{SE} & (l_t^* - \bar{l}_t) \in P \wedge (L_t - l_t^*) \notin P \\ \text{ALLORA} & (L_{t+1} - L_t) \in AM \end{array}$$

Regola L3.

$$\begin{array}{ll} \text{SE} & (l_t^* - \bar{l}_t) \in N \wedge (L_t - l_t^*) \in P \\ \text{ALLORA} & (L_{t+1} - L_t) \in DP \end{array}$$

Regola L4.

$$\begin{array}{ll} \text{SE} & (l_t^* - \bar{l}_t) \in N \wedge (L_t - l_t^*) \notin P \\ \text{ALLORA} & (L_{t+1} - L_t) \in DM \end{array}$$

5.4. — Le funzioni di appartenenza $\mu_P(x)$ e $\mu_N(x)$, relative agli insiemi sfumati P (per *positivo*) e N (per *negativo*), sono definite come segue:

$$\mu_P(x) = \begin{cases} 0 & \text{se } x \leq a, \\ (x-a)/(b-a) & \text{se } a < x < b, \\ 1 & \text{se } x \geq b, \end{cases}$$

dove $a = 3$ e $b = 20$, e

$$\mu_N(x) = \mu_P(-x).$$

Le funzioni di appartenenza $\mu_{AP}(y)$, $\mu_{AM}(y)$, $\mu_{DP}(y)$ e $\mu_{DM}(y)$, relative agli insiemi sfumati AP (per *aumentare poco*), AM (per *aumentare molto*), DP (per *diminuire poco*) e DM (per *diminuire molto*), che descrivono la variazione $y = L_{t+1} - L_t$ della lunghezza massima consentita, sono definite come segue:

$$\mu_{AP}(y) = \begin{cases} y/a & \text{se } 0 < y \leq a, \\ (b-y)/(b-a) & \text{se } a < y < b, \\ 0 & \text{altrove}, \end{cases}$$

dove $a = 5$ e $b = 10$,

$$\mu_{AM}(y) = \begin{cases} 0 & \text{se } y \leq a, \\ (y-c)/(c-a) & \text{se } a < y < c, \\ 1 & \text{se } y \geq c, \end{cases}$$

dove $a = 5$ e $c = 20$ e

$$\mu_{DP}(y) = \mu_{AP}(-y), \qquad \mu_{DM}(y) = \mu_{AM}(-y).$$

5.5. — Dopo aver determinato il nuovo valore della lunghezza massima consentita ai genotipi, gli individui il cui genotipo supera quel valore vengono eliminati dalla popolazione e rimpiazzati con copie dell'individuo migliore (secondo il criterio illustrato in §3.11). Ciò provoca un leggero aumento della pressione selettiva che però è del tutto trascurabile, dato che casi siffatti si presentano molto di rado.

Si è osservato che una fase di contrazione per questo parametro spesso coincide con o è un segno premonitore di un fenomeno di convergenza, e questo potrebbe essere utilizzato come un ulteriore indice di convergenza. Tuttavia ci possono essere eccezioni, in particolare quando la convergenza avviene su un genotipo più lungo della media.

Regolazione del tasso di mutazione.

5.6. — La mutazione negli Algoritmi Evolutivi si rivela utile in due modi: durante la ricerca normale, a un tasso molto basso, essa è responsabile dell'introduzione di innovazioni qua e là, lasciando all'incrocio il tempo di provarne le varie combinazioni; in presenza di una situazione di convergenza, magari prematura, quando un unico genotipo molto al di sopra della media comincia ad assumere il controllo della popolazione, riducendo così la diversità e facendo sì che alla fine il processo evolutivo rimanga bloccato, la mutazione può essere applicata in dosi massicce per reintrodurre diversità e correggere la traiettoria evolutiva. Questa è una tecnica distruttiva da usare con cautela, col rischio, altrimenti, di sprecare considerevoli risorse computazionali.

Se applicata in modo intelligente, la mutazione può aiutare non soltanto a sfuggire dai minimi locali, ma anche a trarre profitto dalla convergenza prematura, sfruttandola come una "base di lancio" per la scoperta di soluzioni migliori. Bisogna tuttavia avvertire il lettore che, sebbene ciò è certamente vero per il problema affrontato, questo potrebbe non valere in generale, specialmente per problemi in cui gli ottimi, locali e non, giacciono separati da vaste regioni a bassa idoneità.

5.7. — Le soluzioni al problema affrontato consistono in diversi "blocchi" (i singoli monomi che compongono il polinomio in Equazione (3)), ciascuno con un peso differente. Il processo evolutivo di ricerca tende a scoprire per primi i blocchi il cui contributo alla soluzione corretta è più importante. Una volta che questi blocchi si sono diffusi nella popolazione, la competizione tra approssimazioni relativamente buone della soluzione inizia a far emergere i blocchi meno importanti. È perciò facile che genotipi che rappresentano esattamente, sebbene solamente, i primi termini di ordine elevato di una soluzione conseguano risultati migliori di altri nelle fasi iniziali, quindi sopprimendoli. Alzando il tasso di mutazione, una tecnica di ricerca cieca, l'algoritmo può allora sfruttare questo gran numero di soluzioni subottimali e parziali per andare in cerca di raffinamenti. In effetti, qui la mutazione funziona bene perché è applicata a un gran numero di individui simili tra di loro. Una volta scoperti genotipi migliori, il

tasso di mutazione può, anzi deve, essere riportato a livelli normali, per non corrompere il nuovo materiale e permettergli di farsi strada nella popolazione.

Vale la pena di far notare che lo stesso corso di eventi avrebbe luogo ugualmente con un tasso di mutazione basso e costante, con la differenza di richiedere molto più tempo.

Il governo fuzzy quindi regola il tasso di mutazione ad ogni generazione per gestire ogni situazione nella maniera più efficiente, utilizzando un predicato sfumato di *convergenza*, che è illustrato qui di seguito.

5.8. — Il tasso di riproduzione Il tasso di riproduzione R_t alla generazione t è definito, nell'algoritmo in questione, come la frequenza di sostituzione degli individui al momento della riproduzione. Questa definizione è strettamente collegata alle modalità di funzionamento del modello di selezione basato sulla competizione (cfr. §3.6): un individuo viene sostituito se e solo se perde una competizione; esso non ciene sostituito se vince *oppure* pareggia.

In media, in una popolazione *diversificata*, cioè in una popolazione in cui gli individui sono tutti diversi l'uno dall'altro, esattamente metà degli individui sarà sostituita. In realtà, in questo caso, R_t è distribuito secondo una distribuzione binomiale

$$P[R_t = r] = \binom{\|X_t\|}{\|X_t\| \cdot r} \frac{1}{2^{\|X_t\|}}, \quad \text{per } 0 \leq r \leq 1,$$

ed ha quindi valore atteso $E[R_t] = \frac{1}{2}$ e varianza $\mathrm{var}[R_t] = \frac{1}{4\|X_t\|}$; considerando che nell'implementazione attuale $\|X_t\| = 8192$ for all t, si ha che $\mathrm{var}[R_t] \simeq 0.00003$, per cui si assumerà semplicemente che sia $R_t = \frac{1}{2}$ quando X_t è diversificata.

Al crescere del numero di individui tra loro identici, un maggior numero di competizioni risulterà in un pareggio, e dunque R_t diminuirà; a un estremo, se tutti gli individui sono identici, non vi sarà alcuna sostituzione ed R_t sarà nullo.

Ora, la quantità $2R_t$ è uno stimatore della versione fenotipica dell'indice di diversità $D(X_t)$ della Definizione II.5.6, e può quindi essere utilizzato come un indice della diversità.

Stime alternative della diversità fenotipica potrebbero essere il diametro dell'idoneità $R(X_t)$ (Definizione II.5.3) e la varianza della distribuzione dell'idoneità; tuttavia si è scelto il tasso di riproduzione poiché esso era già disponibile e "gratis".

5.9. — Il controllo sfumato della mutazione Si è definito un insieme sfumato C (per *convergenza*) sulla base di $D(X_t)$, con una funzione di appartenenza trapezoidale μ_C estratta dall'esperienza maturata con un gran numero di esecuzioni di prova dell'algoritmo:

$$\mu_C(D(X_t)) = \begin{cases} 1 & \text{se } D(X_t) \leq a, \\ (b - D(X_t))/(b-a) & \text{se } a < D(X_t) < b, \\ 0 & \text{se } D(X_t) \geq b, \end{cases}$$

dove $a = 0.24$ e $b = 0.32$. In altri termini, valori di $D(X_t) \geq b$ sono ritenuti abbastanza tipici di una situazione normale, mentre in presenza di valori di $D(X_t) \leq a$ vi è la

certezza che l'evoluzione è in una condizione di convergenza; i casi intermedi sono considerati incerti e quest'incertezza è riflessa in modo appropriato dalla funzione di appartenenza.

Le regole incaricate della regolazione del tasso di mutazione sono le seguenti:

Regola M1.

SE $D(X_t) \in C$
ALLORA $P_m = 0.6$

Regola M2.

SE $D(X_t) \notin C$
ALLORA $P_m = 0.05$

Regolazione dell'intervallo di valutazione.

5.10. — Il modello di selezione introdotto nel Cap. III e ripreso in §3.6 si basa sull'estrazione casuale di un punto di valutazione da un insieme di numeri interi positivi $\{1, 2, \ldots, m\}$. Dato che nella versione dell'algoritmo con governo fuzzy m cambia di generazione in generazione, si scriverà m_t per indicare il valore di m alla t-esima generazione. Il parametro m dipende dalla particolare istanza del problema: in generale, esso dovrebbe essere il più grande intero per cui la valutazione dei genotipi corrispondenti alla soluzione corretta può essere eseguita senza incorrere in eccezioni di underflow o di overflow.

Mentre l'overflow è facile da rivelare, poiché la capacità della codifica in virgola mobile viene oltrepassata e quindi risulta sempre in un'eccezione nell'esecuzione o in un risultato non numerico, l'underflow può passare inosservata. Un'eccezione di underflow ha luogo quando, per esempio, viene sommato un numero molto piccolo a un numero molto grande: se la differenza tra i due è sufficientemente grande, la mantissa può non avere abbastanza cifre di precisione per tener conto del numero piccolo e l'arrotondamento del risultato annulla l'apporto dell'addendo più piccolo e restituisce semplicemente l'addendo più grande.

5.11. — Le regole fuzzy che adattano il parametro m_t un modo da soddisfare i requisiti sopra esposti sono le seguenti, dove ν_{of} e ν_{uf} sono le frequenze rispettivamente di overflow e underflow nel corso dell'ultima fase valutazione:

Regola I1.

SE $\nu_{uf} \notin E \wedge \nu_{of} \notin E$
ALLORA *aumentare* m

Regola I2.

SE $\nu_{uf} \in E \vee \nu_{of} \in E$
ALLORA *diminuire* m

L'insieme E (per *elevata*) naturalmente è sfumato con funzione di appartenenza

$$\mu_E(x) = \begin{cases} 0 & \text{se } x \leq a, \\ (x-a)/(b-a) & \text{se } a < x < b, \\ 1 & \text{se } x \geq b, \end{cases}$$

dove $a = 0.05$ e $b = 0.6$. Il valore iniziale del limite superiore dell'intervallo di valutazione è arbitrariamente fissato a $m_0 = 200$.

Si è osservato in §5.7 che le soluzioni al problema trattato sono composte da termini con diversi pesi relativi. I termini di grado basso diventano relativamente più importanti per punti di valutazione più vicini all'origine, mentre per punti di valutazione lontani dall'origine prevalgono i termini di grado elevato: questo suggerisce, come sviluppo futuro, l'aggiunta di regole che prendano questo fatto in considerazione nel regolare m, al fine di privilegiare i termini di grado baso o elevato a seconda della fase della ricerca in cui si trova l'algoritmo.

Rilevamento dell'emergenza.

5.12. — Le regole per il rilevamento dell'emergenza possono fare uso di alcune delle statistiche introdotte nel Cap. II, e precisamente di statistiche collegate all'idoneità e di indici di diversità. Sebbene l'algoritmo descritto non usi un'idoneità esplicita per il suo funzionamento interno, si è accennato in §3.11 a come sia possibile calcolare una stima dell'idoneità di ciascun individuo presente nella popolazione come una funzione degli errori relativi medi durante la sua vita.

Le regole usate dal governo fuzzy per stabilire un grado di confidenza sull'emergenza di una soluzione ottimale sono le sequenti:

Regola E1.

SE $f(X_t) \in B \wedge D(X_t) \in C$
ALLORA NOT *emergenza*

Regola E2.

SE $f(X_t) \in A \wedge D(X_t) \in C$
ALLORA *emergenza*

Regola E3.

SE $f(X_t) \in B \wedge D(X_t) \notin C$
ALLORA NOT *emergenza*

Regola E4.

SE $f(X_t) \in A \wedge f^*(X_t)/f(X_t) \in G$
ALLORA *superindividuo*

5.13. — Le funzioni di appartenenza $\mu_B(x)$ e $\mu_A(x)$, relative agli insiemi sfumati B (per *bassa*) e A (per *alta*), sono definite in modo appropriato, così come la funzione di appartenenza $\mu_G(x)$, relativa all'insieme sfumato G (per *grande*).

Azioni correttive in un caso come quello della Regola E3 possono comprendere l'aumento della pressione selettiva e l'azzeramento del tasso di mutazione.

Un superindividuo (cfr. Cap. I, §9.4) tipicamente appare per qualche fortunata coincidenza e quindi è necessario prendere subito provvedimenti per assicurare la sua sopravvivenza nella generazione successiva. Le strategie che garantiscono la sopravvivenza dell'individuo migliore nella generazione successive sono dette elitistiche (cfr. Cap. I, §9.9).

§ 6. — **Conclusioni.**

6.1. — L'algoritmo sopra descritto è stato sperimentato su nove istanze diverse del problema, precisamente quelle con $k = 1, \ldots, 9$, con e senza il governo fuzzy descritto nel paragrafo precedente.

Il numero medio di generazioni necessarie per trovare la soluzione senza governo fuzzy ma sotto la supervisione umana o senza supervisione ma con il governo fuzzy si è rivelato pressoché indistinguibile. Tuttavia, il tempo macchina per generazione si è dimostrato minore nel secondo caso, grazie principalmente agli effetti della regolazione dinamica della lunghezza massima dei genotipi.

6.2. — I risultati raccolti dimostrano che le prestazioni dell'algoritmo evolutivo controllato dal governo fuzzy sono confrontabili a quelle dello stesso algoritmo fatto funzionare con la supervisione di un operatore umano.

Chiaramente questo rappresenta solo un risultato parziale, la cui validità e applicabilità nel caso generale resta ancora da provare; ciononstante, esso lascia ben sperare che questa sia la direzione giusta per ridurre l'esigenza di intervento umano nello sfruttamento pratico degli Algoritmi Evolutivi.

BIBLIOGRAFIA

[1] E. H. L. Aarts, A. E. Eiben, and K. M. van Hee. *A General Theory of Genetic Algorithms*. Computing Science Notes. Eindhoven University of Technology, Eindhoven, 1989.

[2] J. Antonisse. A new interpretation of schema notation that overturns the binary encoding constraint. In J. D. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, San Mateo, CA, 1989. Morgan Kaufmann.

[3] S. Arnone, A. Loraschi, and A. Tettamanzi. A genetic approach to portfolio selection. *Neural Network World, International Journal on Neural and Mass-Parallel Computing and Information Systems*, 3(6):597–604, 1993.

[4] H. Asoh and H. Mühlenbein. On the mean convergence time of genetic populations without selection. Technical report, GMD, Sankt Augustin, 1994.

[5] N. Baba. Utilization of stochastic automata and genetic algorithms for neural network learning. In R. Männer and B. Manderick, editors, *Parallel Problem Solving from Nature, 2*, Amsterdam, 1992. Nort-Holland.

[6] D. Back and F. Hoffmeister. Extended selection mechanisms in genetic algorithms. In R. K. Belew and L. B. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, San Mateo, CA, 1991. Morgan Kaufmann.

[7] T. Bäck and H.-P. Schwefel. An overview of evolutionary algorithms for parameter optimization. *Evolutionary Computation*, 1(1):1–23, Spring 1993.

[8] J. Baker. Adaptive selection methods for genetic algorithms. In J. J. Grefenstette, editor, *Proceedings of the First International Conference on Genetic Algorithms*, Hillsdale, NJ, 1985. Lawrence Erlbaum Associates.

[9] J. W. Bala and H. Wechsler. Learning to detect targets using scale-space and genetic search. In S. Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, San Mateo, CA, 1993. Morgan Kaufmann.

[10] W. Banzhaf. Genetic programming for pedestrians. In S. Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, San Mateo, CA, 1993. Morgan Kaufmann.

[11] K. Bennett, M. C. Ferris, and Y. E. Ioannidis. A genetic algorithm for database query optimization. In R. K. Belew and L. B. Booker, editors, *Proceedings of the*

Fourth International Conference on Genetic Algorithms, San Mateo, CA, 1991. Morgan Kaufmann.

[12] A. Bergmann, W. Burgard, and A. Hemker. Adjusting parameters of genetic algorithms by fuzzy control rules. In K.-H. Becks and D. Perret-Gallix, editors, *New Computing Techniques in Physics Reserach III*. 1993. QC793.47 E47 I58 1993 (Physics).

[13] G. Berry and G. Boudol. The chemical abstract machine. *Theoretical Computer Science: the Journal of the EATCS*, 96(1):217, 1992.

[14] A. Bertoni and M. Dorigo. Implicit parallelism in genetic algorithms. *Artificial Intelligence*, 61(2):307–314, 1993.

[15] A. D. Bethke. *Genetic Algorithms as Function Optimizers*. PhD thesis, University of Michigan, Ann Arbor, 1981.

[16] H.-G. Beyer. Some aspects of the 'evolution strategy' for solving TSP-like optimization problems appearing at the design studies of a $0.5TeV$ e^+e^--linear collider. In R. Männer and B. Manderick, editors, *Parallel Problem Solving from Nature, 2*, Amsterdam, 1992. Nort-Holland.

[17] J. N. Bhuyan, V. V. Raghavan, and V. K. Elayavalli. Genetic algorithms for clustering with an ordered representation. In R. K. Belew and L. B. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, San Mateo, CA, 1991. Morgan Kaufmann.

[18] G. E. P. Box. Evolutionary operation: A method for increasing industrial productivity. *Journal of the Royal Statistical Society*, 6(2):271–286, 1957. C.

[19] M. F. Branlette and R. Cusic. A comparative evaluation of search methods applied to parametric design of aircraft. In J. D. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, San Mateo, CA, 1989. Morgan Kaufmann.

[20] H. Braun. On solving travelling salesman problems by genetic algorithms. In H. P. Schwefel and R. Männer, editors, *Parallel Problem Solving from Nature. 1st Workshop PPSN I*, Berlin, 1991. Springer-Verlag.

[21] H. J. Bremermann. The evolution of intelligence. the nervous system as a model of its environment. Technical Report 1, Contract No. 477(17), Department of Mathematics, University of Washington, Seattle, 1958.

[22] A. Brindle. Genetic agorithms for function optimization. Technical Report TR81-2, Department of Computer Science, University of Alberta, Edmonton, 1981.

[23] M. G. Bulmer. *The Mathematical Theory of Quantitative Genetics*. Clarendon Press, Oxford, 1980.

[24] C. Caldwell and V. S. Johnston. Tracking a criminal suspect through "face-space" with a genetic algorithm. In R. K. Belew and L. B. Booker, editors, *Proceedings*

of the Fourth International Conference on Genetic Algorithms, San Mateo, CA, 1991. Morgan Kaufmann.

[25] D. L. Calloway. Using a genetic algorithm to design binary phase-only filters for pattern recognition. In R. K. Belew and L. B. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, San Mateo, CA, 1991. Morgan Kaufmann.

[26] V. Capasso. *Introduzione alla Teoria dei Processi Stocastici.* Quaderni M.U.R.S.T. 40%. Dip. di Matematica U. Dini – Università di Firenze, Viale Morgagni 67a, 50134 Firenze, 1992.

[27] R. A. Caruana and J. D. Schaffer. Representation and hidden bias: Gray vs. binary coding for genetic algorithms. In J. Laird, editor, *Proceedings of the fifth International Conference on Machine Learning*, San Mateo, CA, 1988. Morgan Kaufmann.

[28] C.-H. H. Chu. A genetic algorithm approach to the configuration of stack filters. In J. D. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, San Mateo, CA, 1989. Morgan Kaufmann.

[29] G. A. Cleveland and S. F. Smith. Using genetic algorithms to schedule flow shop releases. In J. D. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, San Mateo, CA, 1989. Morgan Kaufmann.

[30] J. P. Cohoon, S. U. Hegde, W. N. Martin, and D. Richards. Punctuated equilibria: A parallel genetic algorithm. In John J. Grefenstette, editor, *Genetic algorithms and their application: Proceedings of the Second International Conference on Genetic Algorithms*, Hillsdale, NJ, 1987. Lawrence Erlbaum Associated.

[31] R. Collins and D. Jefferson. Selection in massively parallel genetic algorithms. In R. K. Belew and L. B. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 249–256, San Mateo, CA, 1991. Morgan Kaufmann.

[32] A. Colorni, M. Dorigo, and V. Maniezzo. Genetic algorithms and highly constrained problems: The time-table case. In H. P. Schwefel and R. Männer, editors, *Parallel Problem Solving from Nature. 1st Workshop PPSN I*, Berlin, 1991. Springer-Verlag.

[33] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms.* The MIT Press, Cambridge, Massachussets, 1990.

[34] C. Darwin. *On the Origin of Species by Means of Natural Selection.* John Murray, 1859.

[35] L. Davis. Applying adaptive algorithms to epistatic domains. In *Proceedings of the 9th International Joint Conference on Artificial Intelligence*, 1985.

[36] L. Davis. *Handbook of Genetic Algorithms.* VNR Computer Library. Van Nostrand Reinhold, New York, 1991.

[37] R. Dawkins. *The blind Watchmaker*. Norton, 1987.

[38] K. Deb and D. E. Goldberg. A comparative analysis of selection schemes used in genetic algorithms. In Gregory J. E. Rawlins, editor, *Foundations of Genetic Algorithms*, San Mateo, CA, 1991. Morgan Kaufmann.

[39] K. A. DeJong and W. M. Spears. Using genetic algorithms to solve NP-Complete problems. In J. D. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, San Mateo, CA, 1989. Morgan Kaufmann.

[40] M. Dorigo and V. Maniezzo. Parallel genetic algorithms: Introduction and overview of current research. In J. Stenders, editor, *Parallel Genetic Algorithms: Theory and Applications*, Amsterdam, 1992. IOS Press.

[41] A. E. Eiben, E. H. L. Aarts, and K. M. van Hee. Global convergence of genetic algorithms: a Markov chain analysis. In H. P. Schwefel and R. Männer, editors, *Parallel Problem Solving from Nature. 1st Workshop PPSN I*, Berlin, 1991. Springer-Verlag.

[42] W. Feller. *An Introduction to Probability Theory and Its Applications*, volume 1. John Wiley & Sons, New York, 1950.

[43] J. Felsenstein. The theoretical population genetics of variable selection and migration. *Annual Review of Genetics*, 10:253–280, 1976.

[44] S. J. Flockton and M. S. White. Pole-zero system identification using genetic algorithms. In S. Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, San Mateo, CA, 1993. Morgan Kaufmann.

[45] D. B. Fogel. *Evolving Artificial Intelligence*. PhD thesis, University of California, San Diego, 1992.

[46] L. J. Fogel. Autonomous automata. *Industrial Research*, 4:14–19, 1962.

[47] L. J. Fogel. *On the Organization of Intellect*. PhD thesis, University of California, Los Angeles, 1964.

[48] L. J. Fogel, A. J. Owens, and M. J. Walsh. *Artificial Intelligence through Simulated Evolution*. John Wiley & Sons, New York, 1966.

[49] A. Fraser. Simulation of genetic systems by automatic digital computers: I. introduction. *Australian Journal of Biological Science*, 10:484–491, 1957.

[50] P. S. Gabbert, D. E. Brown, C. L. Huntley, B. P. Markowicz, and D. E. Sappington. A system for learning routes and schedules with genetic algorithms. In R. K. Belew and L. B. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, San Mateo, CA, 1991. Morgan Kaufmann.

[51] M. R. Garey and D. S. Johnson. *Computers and Intractability: A guide to the Theory of NP-Completeness*. Freeman, New York, 1979.

[52] M. Gerrits and P. Hogeweg. Redundant coding of an NP-Complete problem allows effective genetic algorithm search. In H. P. Schwefel and R. Männer,

editors, *Parallel Problem Solving from Nature. 1st Workshop PPSN I*, Berlin, 1991. Springer-Verlag.

[53] A. M. Gillies. *Machine Learning Procedures for Generating Image Domain Feature Detectors*. PhD thesis, University of Michigan, Ann Arbor, 1985.

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

[55] D. E. Goldberg. The theory of virtual alphabets. In H. P. Schwefel and R. Männer, editors, *Parallel Problem Solving from Nature. 1st Workshop PPSN I*, Berlin, 1991. Springer-Verlag.

[56] D. E. Goldberg and X. Lingle. Alleles, loci and the traveling salesman problem. In John J. Grefenstette, editor, *Genetic algorithms and their application: Proceedings of the Second International Conference on Genetic Algorithms*, Hillsdale, NJ, 1987. Lawrence Erlbaum Associated.

[57] D. E. Goldberg and P. Segrest. Finite Markov chain analysis of genetic algorithms. In John J. Grefenstette, editor, *Genetic algorithms and their application: Proceedings of the Second International Conference on Genetic Algorithms*, pages 1–8, Hillsdale, NJ, 1987. Lawrence Erlbaum Associated.

[58] R. L. Graham, D. E. Knuth, and O. Patashnik. *Concrete Mathematics: A foundation for Computer Science*. Addison-Wesley, Reading, MA, 1988.

[59] J. J. Grefenstette. Parallel adaptive algorithms for function optimization (preliminary report). Technical Report CS-81-19, Computer Science Department, Vanderbuilt University, 1981.

[60] J. J. Grefenstette. A system for learning control strategies with genetic algorithms. In J. D. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, San Mateo, CA, 1989. Morgan Kaufmann.

[61] H. A. Güvenir and İ. Şirin. A genetic algorithm for classification by feature partitioning. In S. Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, San Mateo, CA, 1993. Morgan Kaufmann.

[62] P. J. B. Hancock. Recombination operators for the design of neural nets by genetic algorithm. In R. Männer and B. Manderick, editors, *Parallel Problem Solving from Nature, 2*, Amsterdam, 1992. Nort-Holland.

[63] J. Hesser and R. Männer. Towards an optimal mutation probability for genetic algorithms. In H. P. Schwefel and R. Männer, editors, *Parallel Problem Solving from Nature. 1st Workshop PPSN I*, volume 496 of *Lecture Notes in Computer Science*, pages 23–32, Berlin, October 1–3 1991. Springer-Verlag.

[64] J. Hesser, R. Männer, and O. Stucky. Optimization of steiner trees using genetic algorithms. In J. D. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, San Mateo, CA, 1989. Morgan Kaufmann.

[65] J. H. Holland. *Adaptation in Natural and Artificial Systems*. The University of Michigan Press, Ann Arbor, 1975.

[66] R. B. Hollstien. *Artificial Genetic Adaptation in Computer Systems*. PhD thesis, University of Michigan, Ann Arbor, 1971.

[67] A. Homaifar, S. Guan, and G. E. Liepins. A new approach on the traveling salesman problem by genetic algorithms. In S. Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, San Mateo, CA, 1993. Morgan Kaufmann.

[68] J. E. Hopcroft and J. D. Ullman. *Formal Languages and Their Relation to Automata*. Addison-Wesley series in Computer Science and Information Processing. Addison-Wesley, Reading, MA, 1969.

[69] M. Hulin. Circuit partitioning with genetic algorithms using a coding scheme to preserve the structure of a circuit. In H. P. Schwefel and R. Männer, editors, *Parallel Problem Solving from Nature. 1st Workshop PPSN I*, Berlin, 1991. Springer-Verlag.

[70] K. J. Hunt. Optimal control system synthesis with genetic algorithms. In R. Männer and B. Manderick, editors, *Parallel Problem Solving from Nature, 2*, Amsterdam, 1992. Nort-Holland.

[71] P. Husbands. An ecosystem model for integrated production planning. *The International Journal of Computer Integrated Manufacturing*, 6(1&2):74–86, 1993.

[72] P. Husbands, F. Mill, and S. Warrington. Genetic algorithms, production plan optimization and scheduling. In H. P. Schwefel and R. Männer, editors, *Parallel Problem Solving from Nature. 1st Workshop PPSN I*, Berlin, 1991. Springer-Verlag.

[73] Jr. J. L. Blanton and R. L. Wainwright. Multiple vehicle routing with time and capacity constraints using genetic algorithms. In S. Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, San Mateo, CA, 1993. Morgan Kaufmann.

[74] W. Jakob, M. Gorges-Schleuter, and C. Blume. Application of genetic algorithms to task planning and learning. In R. Männer and B. Manderick, editors, *Parallel Problem Solving from Nature, 2*, Amsterdam, 1992. Nort-Holland.

[75] T. Johnson and P. Husbands. System identification using genetic algorithms. In H. P. Schwefel and R. Männer, editors, *Parallel Problem Solving from Nature. 1st Workshop PPSN I*, Berlin, 1991. Springer-Verlag.

[76] D. R. Jones and M. A. Beltramo. Solving partitioning problems with genetic algorithms. In R. K. Belew and L. B. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, San Mateo, CA, 1991. Morgan Kaufmann.

[77] G. Jones, R. D. Brown, D. E. Clark, P. Willet, and R. C. Glen. Searching databases of two-dimensional and three-dimensional chemical structures using genetic algorithms. In S. Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, San Mateo, CA, 1993. Morgan Kaufmann.

[78] K. A. De Jong. *An Analysis of the Behaviour of a Class of Genetic Adaptive Systems*. PhD thesis, University of Michigan, Ann Arbor, 1975.

[79] K. Juliff. A multi-chromosome genetic algorithm for pallet loading. In S. Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, San Mateo, CA, 1993. Morgan Kaufmann.

[80] B. A. Julstrom. A genetic algorithm for the rectilinear steiner problem. In S. Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, San Mateo, CA, 1993. Morgan Kaufmann.

[81] C. L. Karr. Design of an adaptive fuzzy logic controller using a genetic algorithm. In R. K. Belew and L. B. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, San Mateo, CA, 1991. Morgan Kaufmann.

[82] M. J. Keith and M. C. Martin. Genetic programming in C++: Implementation issues. In K. E. Kinnear, editor, *Advances in Genetic Programming*, Complex Adaptive Systems. The MIT Press, Cambridge, Massachussets, 1994.

[83] J. R. Koza. Concept formation and decision tree induction using the genetic programming paradigm. In H. P. Schwefel and R. Männer, editors, *Parallel Problem Solving from Nature. 1st Workshop PPSN I*, Berlin, 1991. Springer-Verlag.

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

[85] B. Kröger, P. Schwenderling, and O. Vornberger. Parallel genetic packing of rectangles. In H. P. Schwefel and R. Männer, editors, *Parallel Problem Solving from Nature. 1st Workshop PPSN I*, Berlin, 1991. Springer-Verlag.

[86] M. Lee and H. Takagi. Dynamic control of genetic algorithms using fuzzy logic techniques. In S. Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, San Mateo, CA, 1993. Morgan Kaufmann.

[87] M. Lee and H. Takagi. Embedding apriori knowledge into an integrated fuzzy system design method based on genetic algorithms. In *Proceedings of the 5th IFSA World Congress (IFSA'93)*, pages Vol. II, 1293–1296, July 4–9 1993.

[88] M. Lee and H. Takagi. Integrating design stages of fuzzy systems using genetic algorithms. In *Proceedings of the 2nd International Conference on Fuzzy Systems (FUZZ-IEEE'93)*, pages Vol. I, 612–617, 1993.

[89] D. M. Levine. A genetic algorithm for the set partitioning problem. In S. Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, San Mateo, CA, 1993. Morgan Kaufmann.

[90] S. Levy. *Artificial Life: The quest for a new creation*. Pantheon Books, New York, 1992.

[91] M. Lidd. Traveling salesman problem domain application of a fundamentally new approach to genetic algorithms. Technical Report Contract F4920-90-G-0033, Research sponsored in part by Air Force Office of Scientific Research and Office of Naval Research, 1991.

[92] J. Lienig and K. Thulasiraman. A genetic algorithm for channel routing in vlsi circuits. *Evolutionary Computation*, 1(4):293–311, Winter 1993.

[93] G. E. Liepins. On global convergence of genetic algorithms. In Su-Shing Chen, editor, *Neural and Stochastic Methods in Image and Signal Processing*, volume SPIE-1766, pages 61–65. The International Society for Optical Engineering, San Diego, CA, July 1992.

[94] S.-E. Ling. Integrating genetic algorithms with a prolog assignment program as a hybrid solution for a polytechnic timetable broblem. In R. Männer and B. Manderick, editors, *Parallel Problem Solving from Nature, 2*, Amsterdam, 1992. Nort-Holland.

[95] R. Sh. Liptser and A. N. Shiryayev. *Theory of Martingales.* Mathematics and Its Applications (Soviet Series). Kluwer Academic Publishers, Dordrecht, 1989.

[96] C. B. Lucasius and G. Kateman. Application of genetic algorithms in chemometrics. In J. D. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, San Mateo, CA, 1989. Morgan Kaufmann.

[97] C. B. Lucasius, S. Werten, A. H. J. M. van Aert, G. Kateman, and M. J. J. Blommers. Conformational analysis of DNA using genetic algorithms. In H. P. Schwefel and R. Männer, editors, *Parallel Problem Solving from Nature. 1st Workshop PPSN I*, Berlin, 1991. Springer-Verlag.

[98] B. Manderick and P. Spiessens. Fine-grained parallel genetic algorithms. In J. D. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, San Mateo, CA, 1989. Morgan Kaufmann.

[99] B. Manderick and P. Spiessens. A massively parallel genetic algorithm: implementation and first analysis. In R. K. Belew and L. B. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 279–286, San Mateo, CA, 1991. Morgan Kaufmann.

[100] M. Manela, N. Thornhill, and J. A. Campbell. Fitting spline functions to noisy data using a genetic algorithm. In S. Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, San Mateo, CA, 1993. Morgan Kaufmann.

[101] R. E. Marks. Breeding hybrid strategies: Optimal behavior for oligopolists. In J. D. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, San Mateo, CA, 1989. Morgan Kaufmann.

[102] R. San Martin and J. P. Knight. Genetic algorithms for optimization of integrated circuit synthesis. In S. Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, San Mateo, CA, 1993. Morgan Kaufmann.

[103] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs.* Springer-Verlag, Berlin, 1992.

[104] R. Milner. *Communication and Concurrency.* International Series in Computer Science. Prentice-Hall, Englewood Cliffs, 1976.

[105] A. M. Mood, F. A. Graybill, and D. C. Boes. *Introduction to the Theory of Statistics*. McGraw-Hill Series in Probability and Statistics. McGraw-Hill, New York, 1950.

[106] H. Mühlenbein. Evolution in time and space–the parallel genetic algorithm. In Gregory J. E. Rawlins, editor, *Foundations of Genetic Algorithms*, San Mateo, CA, 1991. Morgan Kaufmann.

[107] H. Mühlenbein, M. Gorges-Schleuter, and O. Krämer. Evolution algorithms in combinatorial optimization. *Parallel Computing*, 7:65–85, 1988.

[108] H. Mühlenbein and D. Schlierkamp-Voosen. The science of breeding and its application to the breeder genetic algorithm (bga). *Evolutionary Computation*, 1(4):335–360, Winter 1993.

[109] H. Mühlenbein and G. von Laszewski. Partitioning a graph with a parallel genetic algorithm. In H. P. Schwefel and R. Männer, editors, *Parallel Problem Solving from Nature. 1st Workshop PPSN I*, Berlin, 1991. Springer-Verlag.

[110] T. Munakata and D. J. Hashier. A genetic algorithm applied to the maximum flow problem. In S. Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, San Mateo, CA, 1993. Morgan Kaufmann.

[111] R. Nakano and T. Yamada. Conventional genetic algorithm for job shop problems. In R. K. Belew and L. B. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, San Mateo, CA, 1991. Morgan Kaufmann.

[112] A. E. Nix and M. D. Vose. Modeling genetic algorithms with markov chains. *Annals of Mathematics and Artificial Intelligence*, 5(1):79–88, 1992.

[113] P. Nordin. A compiling genetic programming system that directly manipulates the machine code. In K. E. Kinnear, editor, *Advances in Genetic Programming*, Complex Adaptive Systems. The MIT Press, Cambridge, Massachussets, 1994.

[114] J.-P. Nordvik and J.-M. Renders. Genetic algorithms and their potential for use in process control: A case study. In R. K. Belew and L. B. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, San Mateo, CA, 1991. Morgan Kaufmann.

[115] M. O. Odetayo and D. R. McGregor. Genetic algorithm for inducing control rules for a dynamic system. In J. D. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, San Mateo, CA, 1989. Morgan Kaufmann.

[116] C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization, Algorithms and Complexity*. Prentice-Hall, Englewood Cliffs, 1982.

[117] C. B. Pettey, M. R. Leuze, and J. J. Grefenstette. A parallel genetic algorithm. In John J. Grefenstette, editor, *Genetic algorithms and their application: Proceedings of the Second International Conference on Genetic Algorithms*, Hillsdale, NJ, 1987. Lawrence Erlbaum Associated.

[118] D. Polani and T. Uthmann. Adaptation of kohonen feature map topologies by genetic algorithms. In R. Männer and B. Manderick, editors, *Parallel Problem Solving from Nature, 2*, Amsterdam, 1992. Nort-Holland.

[119] D. J. Powell and M. M. Skolnick. Using genetic algorithms in engineering design optimization with non-linear constraints. In S. Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, San Mateo, CA, 1993. Morgan Kaufmann.

[120] D. J. Powell, S. S. Tong, and M. M. Skolnick. EnGENEous domain independent, machine learning for design optimization. In J. D. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, San Mateo, CA, 1989. Morgan Kaufmann.

[121] W. F. Punch, E. D. Goodman, M. Pei, L. Chia-Shun, P. Hovland, and R. Enbody. Further research on feature selection and classification using genetic algorithms. In S. Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, San Mateo, CA, 1993. Morgan Kaufmann.

[122] A. T. Rahmani and N. Ono. A genetic algorithm for channel routing problem. In S. Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, San Mateo, CA, 1993. Morgan Kaufmann.

[123] V. J. Rayward-Smith and A. Clare. On finding steiner vertices. *Networks*, 16:293–294, 1986.

[124] I. Rechenberg. *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution.* Fromman-Holzboog Verlag, Stuttgart, 1973.

[125] P. Roosen and F. Meyer. Determination of chemical equilibria by means of an evolution strategy. In R. Männer and B. Manderick, editors, *Parallel Problem Solving from Nature, 2*, Amsterdam, 1992. Nort-Holland.

[126] G. Rudolph. Parallel approaches to stochastic global optimization. In W. Joosen and E. Milgrom, editors, *Parallel Computing: from Theory to sound Practice. Proceedings of the European Workshop on Parallel Computing*, Amsterdam, 1992. IOS Press.

[127] G. Rudolph. Convergence analysis of canonical genetic algorithms. *IEEE Transactions on Neural Networks*, 5(1):96–101, January 1994.

[128] E. Sanchez and P. Pierre. Fuzzy logic and genetic algorithms in information retrieval. In *3rd International Conference on Fuzzy Logic, Neural Networks and Soft Computing*, 1994.

[129] J. D. Schaffer and L. J. Eshelman. Designing multiplierless digital filters using genetic algorithms. In S. Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, San Mateo, CA, 1993. Morgan Kaufmann.

[130] S. Schulze-Kremer. Genetic algorithms for protein tertiary structure prediction. In R. Männer and B. Manderick, editors, *Parallel Problem Solving from Nature, 2*, Amsterdam, 1992. Nort-Holland.

[131] H.-P. Schwefel. Kybernetische Evolution als Strategie der experimentellen Forschung in der Strömungstechnik. Master's thesis, Technische Universität Berlin, Berlin, 1965.

[132] H.-P. Schwefel. *Numerical optimization of computer models.* Wiley, Chichester; New York, 1981.

[133] H.-P. Schwefel. Collective phenomena in evolutionary systems. In *Preprints of the 31st Annual Meeting of the International Society for General System Research*, Budapest, 1987.

[134] W. Siedlecki and J. Sklansky. Constrained genetic optimization via dynamic reward-penalty balancing and its use in pattern recognition. In J. D. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, San Mateo, CA, 1989. Morgan Kaufmann.

[135] G. Syswerda. Uniform crossover in genetic algorithms. In J. D. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, San Mateo, CA, 1989. Morgan Kaufmann.

[136] G. Syswerda and J. Palmucci. The application of genetic algorithms to resource scheduling. In R. K. Belew and L. B. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, San Mateo, CA, 1991. Morgan Kaufmann.

[137] R. Tanese. Distributed genetic algorithms. In J. D. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, San Mateo, CA, 1989. Morgan Kaufmann.

[138] A. Tettamanzi. A distributed model for selection in evolutionary algorithms. Technical Report 110-94, Dip. di Scienze dell'Informazione – Università degli Studi di Milano, Milano, Italy, 1994.

[139] S. R. Thangiah, R. Vinayagamoorthy, and A. V. Gubbi. Vehicle routing with time deadlines using genetic and local algorithms. In S. Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, San Mateo, CA, 1993. Morgan Kaufmann.

[140] D. Thierens and L. Vercauteren. A topology exploiting genetic algorithms to control dynamic systems. In H. P. Schwefel and R. Männer, editors, *Parallel Problem Solving from Nature. 1st Workshop PPSN I*, Berlin, 1991. Springer-Verlag.

[141] Thinking Machines Corporation, Cambridge, Massachussets. *CM User's Guide*, version 6.1 edition, October 1991. Reprinted, with corrections, December 1992.

[142] P. Thrift. Fuzzy logic synthesis with genetic algorithms. In R. K. Belew and L. B. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, San Mateo, CA, 1991. Morgan Kaufmann.

[143] N. L. J. Ulder, E. H. L. Aarts, H.-J. Bandelt, P. J. M. van Laarhoven, and E. Pesch. Genetic local search algorithms for the traveling salesman problem. In

H. P. Schwefel and R. Männer, editors, *Parallel Problem Solving from Nature. 1st Workshop PPSN I*, Berlin, 1991. Springer-Verlag.

[144] R. Unger and J. Moult. A genetic algorithm for 3d protein folding simulations. In S. Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, San Mateo, CA, 1993. Morgan Kaufmann.

[145] C. L. Valenzuela and A. J. Jones. Evolutionary divide and conquer (i): A novel genetic approach to the tsp. *Evolutionary Computation*, 1(4):313–333, Winter 1993.

[146] D. Whitley, T. Starkweather, and D. Fuquay. Scheduling problems and traveling salesmen: The genetic edge recombination operator. In J. D. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, San Mateo, CA, 1989. Morgan Kaufmann.

[147] D. Williams. *Probability with Martingales*. Cambridge University Press, Cambridge, 1991.

[148] N. Wirth. *Algorithms + Data Structures = Programs*. Prentice-Hall series in automatic computation. Prentice-Hall, Englewood Cliffs, 1976.

[149] P. Wyard. Context free grammar induction using genetic algorithms. In R. K. Belew and L. B. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, San Mateo, CA, 1991. Morgan Kaufmann.

[150] T. Yamada and R. Nakano. A genetic algorithm applicable to large-scale job-shop problems. In R. Männer and B. Manderick, editors, *Parallel Problem Solving from Nature, 2*, Amsterdam, 1992. Nort-Holland.

[151] J.-J. Yang and R. R. Korfhage. Query optimization in information retrieval using genetic algorithms. In S. Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, San Mateo, CA, 1993. Morgan Kaufmann.

[152] D. C. Youvan, A. P. Arkin, and M. M. Yang. Recursive ensemble mutagenesis: A combinatorial optimization technique for protein engineering. In R. Männer and B. Manderick, editors, *Parallel Problem Solving from Nature, 2*, Amsterdam, 1992. Nort-Holland.

[153] L. A. Zadeh. Fuzzy sets. *Information and Control*, 8:338–353, 1965.

[154] L. A. Zadeh. *Fuzzy Sets and Applications: Selected Papers*. John Wiley & Sons, New York, 1987.

[155] L. A. Zadeh. The calculus of fuzzy if-then rules. *AI Expert*, 7(3):22–27, March 1992.

[156] H.-J. Zimmermann. *Fuzzy Set Theory and its Applications*. International series in management science/operations research. Kluwer, Boston, 1985.