

UNIVERSITÀ  
DEGLI STUDI  
DI PADOVA

**dii** DIPARTIMENTO  
DI INGEGNERIA  
INDUSTRIALE

DIPARTIMENTO DI INGEGNERIA INDUSTRIALE

CORSO DI LAUREA IN INGEGNERIA AEROSPAZIALE

# Ottimizzazione di strutture reticolari tramite algoritmi genetici

**Relatore**

Prof. Ugo Galvanetto

**Laureando**

Tommaso Bari

ANNO ACCADEMICO 2023-2024

Data di laurea 20/09/2024



# Sommario

In questo elaborato viene presentata una strategia di ottimizzazione di strutture reticolari attuata tramite un algoritmo evolutivo. Sono esposte le varie strutture dati utilizzate, gli operatori genetici e alcuni casi studio atti a validare l'algoritmo stesso e a dimostrarne l'efficacia.

In particolare gli individui sono codificati tramite matrici d'adiacenza in modo da consentire all'algoritmo di poter generare qualsiasi tipo di struttura reticolare al fine di trovare quella ottimale, imponendo soltanto i vincoli iniziali del problema. Si dimostra inoltre l'efficacia dell'implementazione di strategie per il mantenimento della diversità genetica come la correzione del valore di *fitness* e la *random injection*.

Infine con la trattazione di due casi studio è stata dimostrata l'efficacia dell'algoritmo comparandolo con i principali risultati della lettatura presente attualmente sull'argomento.



# Indice

<b>1 Teoria</b>	<b>1</b>
1.1 Le strutture reticolari . . . . .	1
1.1.1 Risoluzione: il metodo degli spostamenti . . . . .	2
1.2 Gli algoritmi genetici . . . . .	4
1.2.1 Genotipo e fenotipo . . . . .	4
1.2.2 Selezione e fitness . . . . .	5
1.2.3 Operatori genetici . . . . .	6
1.2.4 Convergenza . . . . .	6
<b>2 Algoritmo di ottimizzazione</b>	<b>9</b>
2.1 Obiettivi . . . . .	9
2.2 Funzionamento . . . . .	10
2.3 Rappresentazione dei dati . . . . .	12
2.4 Operatori genetici . . . . .	14
2.4.1 Elitismo . . . . .	14
2.4.2 Crossover . . . . .	14
2.4.3 Mutazioni . . . . .	15
2.5 Validità della struttura . . . . .	17
2.6 Funzione di fitness . . . . .	19
2.7 Diversità genetica . . . . .	20
2.7.1 Fitness corretta . . . . .	20
2.7.2 Random injection . . . . .	20
<b>3 Analisi e risultati</b>	<b>21</b>
3.1 Trave piana a 10 aste . . . . .	21
3.2 Trave piana a 39 aste . . . . .	24
<b>4 Conclusioni</b>	<b>27</b>



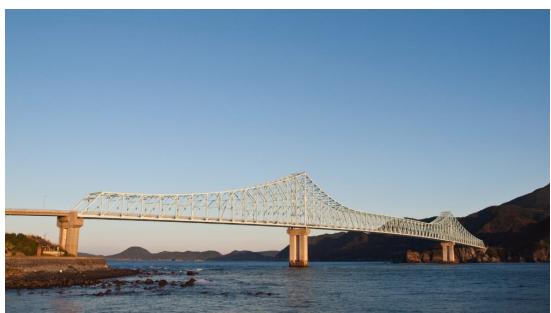
# Capitolo 1

## Teoria

### 1.1 Le strutture reticolari

Le travature reticolari sono strutture complesse, costituite da un insieme di aste collegate fra loro in punti detti nodi [1, p.H-185]. Trovano largo impiego nell'ingegneria civile dove risultano indispensabili nella copertura di grandi luci con strutture leggere, come ad esempio nel ponte Ikitsuki in Giappone nell'iconica Torre Eiffel di Parigi.

Nell'ambito spaziale sono utilizzate principalmente nella costruzione di *frame* per stazioni orbitali o satelliti: ne sono un esempio l'ITS (*Integrated Truss Structure*) della stazione spaziale internazionale e le montature del telescopio Hubble. Con il continuo sviluppo della stampa 3D e della lavorazione dei materiali compositi i telai reticolari stanno prendendo piede anche nella crescente industria dei micro e nano satelliti.



(a) Ponte Ikitsuki [2]



(b) Torre Eiffel [3]

Figura 1.1

In questo elaborato saranno descritte e utilizzate travature reticolari piane a due dimensioni. Tali strutture sono composte da una serie di nodi collegati tra loro da aste e vincolate a terra tramite cerniere o carrelli. La rappresentazione dei nodi come carrelli implica la nullità dei momenti alle estremità delle aste e le conseguenti sollecitazioni di taglio. Si può quindi dedurre



(a) Vista della travatura principale della ISS [4]



(b) *Frame* del telescopio spaziale Hubble [5]

Figura 1.2

che le aste risultano sottoposte solo a sforzo normale: per convenzione questo è assunto positivo se di trazione, negativo se di compressione.

Le strutture reticolari si classificano in base al numero di gradi di libertà. Per convenzione si considerano i nodi come corpi aventi 2 gradi di libertà (spostamento verticale e orizzontale) e le aste come vincoli interni che tolgoano ognuna un grado di libertà. Esaminando una generica struttura avente  $n$  nodi,  $j$  aste e  $r$  vincoli esterni essa può essere classificata come:

- Labile: se  $j + r < 2n$  poichè questa presenta dei gradi di libertà residui.
- Isostatica: se  $j + r = 2n$ , la struttura non presenta gradi di libertà né vincoli sovrabbondanti.
- Iperstatica: se  $j + r > 2n$ , la struttura presenta uno o più vincoli sovrabbondanti.

### 1.1.1 Risoluzione: il metodo degli spostamenti

Le travature isostatiche possono essere facilmente risolte tramite le equazioni classiche dell'equilibrio. In un problema iperstatico invece sono presenti un numero di incognite maggiori delle equazioni ed è quindi necessario considerare l'effetto delle deformazioni sotto le opportune condizioni al contorno. Per la risoluzione delle strutture iperstatiche è stato fatto uso del metodo degli spostamenti<sup>1</sup>[6].

Tale metodo si basa introducendo l'elasticità delle aste per esprimere gli sforzi normali sulle in funzione della loro deformazione e conseguentemente dei loro spostamenti. Per una generica asta che collega i nodi  $i$  e  $j$  se ne può scrivere la deformazione facendo riferimento alla Figura 1.3:

$$\varepsilon_{ij} = \frac{\Delta\ell_{ij}}{\ell_{ij}} = \frac{(u_j - u_i)\cos\alpha + (v_j - v_i)\sin\alpha}{\ell_{ij}}$$

---

<sup>1</sup>Il metodo consente anche la risoluzione di problemi iperstatici

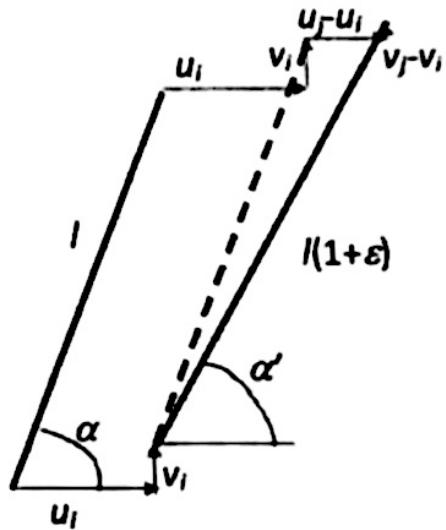


Figura 1.3: Spostamenti e deformazioni nelle aste [6]

Ora applicando la legge di Hooke<sup>2</sup>:

$$\begin{aligned} \sigma_{ij} &= E \varepsilon_{ij} \\ \frac{N_{ij}}{A_{ij}} &= E \left[ \frac{(u_j - u_i) \cos \alpha + (v_j - v_i) \sin \alpha}{\ell_{ij}} \right] \\ N_{ij} &= EA_{ij} \left[ \frac{(u_j - u_i) \cos \alpha + (v_j - v_i) \sin \alpha}{\ell_{ij}} \right] \end{aligned} \quad (1.1)$$

Si può quindi procedere alla scrittura delle equazioni di equilibrio per i singoli nodi in funzione degli spostamenti. Per una generica struttura composta da  $n$  nodi avente  $r$  reazioni vincolari esterne si ottengono  $2n$  equazioni al cui interno sono presenti  $2n + r$  incognite, per rendere risolvibile il sistema è necessario inserire l'effetto dei vincoli esterni: questi bloccano uno o più direzioni di spostamento riducendo di  $r$  il numero di incognite.

Il sistema risulta infine risolvibile. Una volta determinato il valore degli spostamenti basterà utilizzare l'Equazione 1.1 per il calcolo degli sforzi nelle aste.

---

<sup>2</sup>Si considera il modulo di elasticità  $E$  costante per tutte le aste

## 1.2 Gli algoritmi genetici

Gli algoritmi genetici<sup>3</sup> sono algoritmi euristici<sup>4</sup> che si basano sul principio di selezione naturale Darwiniana [7] in cui gli elementi più adatti all’ambiente hanno maggiore possibilità di sopravvivere e di trasmettere le loro caratteristiche ai successori.

Sono stati trattati per la prima volta da John Holland nel 1975 [8] e sono ora applicati in numerosi campi, principalmente per la soluzione di problemi complessi dove non si ha conoscenza completa dello spazio di ricerca.

Il processo avviene a generazioni, in ognuna di queste una popolazione di soluzioni viene classificata grazie ad una funzione di *fitness* che misura il livello di adattamento ogni struttura al problema posto. Gli individui con valori più elevati vengono selezionati e hanno maggiore possibilità che il loro patrimonio genetico (la capacità di risolvere il problema) venga preservato nelle generazioni successive. Durante il processo evolutivo alcuni individui subiscono delle mutazioni con la possibilità di far emergere nuovi caratteri vincenti che consentono l’esplorazione di nuove soluzioni anche completamente diverse da quelle che si possono andare a prevedere con processi analitici.

L’utilizzo degli algoritmi evolutivi risulta di particolare utilità laddove la soluzione al problema posto non sia raggiungibile in modo efficiente tramite algoritmi classici ad esempio lineari o polinomiali. In questi casi sono l’ovvia alternativa al *random search*, ossia l’esplorazione casuale dello spazio delle soluzioni.

Gli algoritmi genetici possono anche essere utilizzati per la ricerca di soluzioni a problemi multiobiettivo ricercando l’ottimo paretiano [9] tramite adeguate funzioni di fitness. In questo elaborato ci si è limitati per semplicità alla trattazione di algoritmi genetici a singolo obiettivo.

### 1.2.1 Genotipo e fenotipo

Un individuo all’interno di un algoritmo genetico ha due rappresentazioni differenti:

- Fenotipo: come l’individuo appare nel problema reale; in questo caso la struttura reticolare è il fenotipo dell’individuo.
- Genotipo: come l’individuo è codificato a livello genetico.

Per ottenere un buon algoritmo è necessario che il fenotipo sia correttamente rappresentato dal genotipo e che quest’ultimo sia facilmente utilizzabile all’interno degli operatori genetici e nel processo di selezione.

---

<sup>3</sup>Il termine genetici fa riferimento specifico ad algoritmi rappresentati tramite stringhe di bit, in questo elaborato il termine è reso intercambiabile con “evolutivi” come sovente fatto in letteratura

<sup>4</sup>algoritmi che in dati problemi si avvicinino con buona probabilità alla soluzione cercata

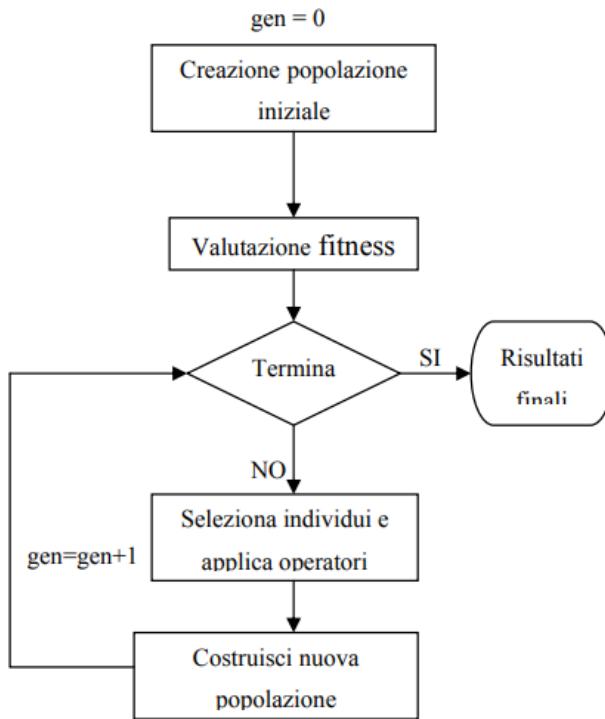


Figura 1.4: Schema esempio di un algoritmo evolutivo [10]

### 1.2.2 Selezione e fitness

Al fine di selezionare i migliori individui da trasportare da una generazione all'altra è necessario classificarli, questo avviene tramite una funzione detta di *fitness*. Essa indica la capacità di una soluzione nel raggiungere gli obiettivi del problema posto.

Una volta classificati, gli individui migliori vengono selezionati in modo che i loro geni possano passare alla popolazione della generazione successiva attraverso gli operatori genetici (1.2.3). I principali metodi di selezione sono [10]:

- Proporzionali: ad ogni individuo viene assegnata una probabilità di essere scelto; Questa sarà tanto maggiore quanto è maggiore il suo valore di *fitness*.
- *K-tournament*: gli individui vengono fatti competere all'interno di tornei nei quali vince quello con la miglior *fitness*. I partecipanti al torneo vengono tipicamente scelti in modo casuale tra la popolazione.

In entrambi i casi gli individui con maggior *fitness* hanno più possibilità di essere selezionati.

### 1.2.3 Operatori genetici

Gli operatori genetici sono azioni che vengono eseguite sul genoma degli individui di una generazione al fine di generare nuovi individui per la generazione successiva: I principali operatori genetici sono [10]:

1. Riproduzione: copia semplicemente l'individuo nella nuova popolazione.
2. Mutazione: variazione casuale all'interno del genotipo di un individuo; le mutazioni possono essere favorevoli o sfavorevoli.
3. *Crossover*: equivalente della riproduzione in ambito biologico; il genotipo di due individui genitori viene combinato dando origine ad un terzo individuo figlio.

### 1.2.4 Convergenza

La convergenza di un algoritmo verso la miglior soluzione non è sempre garantita ed esso tenderà ad una sola soluzione ottimale, rappresentata generalmente da un massimo locale e perdendo di vista altre possibili soluzioni.

Questo è ben dimostrato dai risultati ottenuti dalle simulazioni svolte dove, con gli stessi parametri applicati, l'algoritmo converge verso soluzioni diverse tra loro, non sempre assolutamente ottimali.

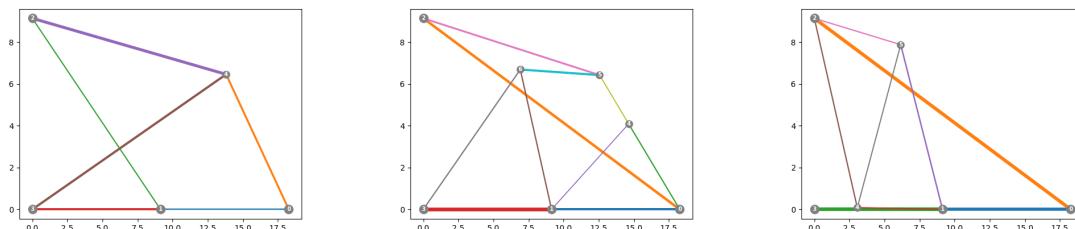


Figura 1.5: Differenti soluzioni al termine delle simulazioni (Esempi estratti dal caso studio 3.1)

Questo fenomeno prende il nome di convergenza prematura ed è dovuto principalmente al processo di selezione e riproduzione che porta implicitamente all'uniformazione degli individui verso un solo genotipo ottimizzato. Ciò è dovuto al fatto che un individuo con un valore molto più alto di *fitness* avrà più possibilità di generare figli e, di conseguenza, questi saranno più simili a lui andando a ridurre la diversità genetica della popolazione [11], impedendo successive evoluzioni. Se non controllato il fenomeno può portare alla completa sostituzione della popolazione con il genotipo di un singolo individuo.

Si può quindi dedurre che il mantenimento di una diversità genetica all'interno della popolazione è fondamentale per garantire un'esplorazione maggiore dello spazio delle soluzioni e uscire dove necessario dai minimi locali [12]; questo può essere ottenuto introducendo meccanismi che aumentino la pressione evolutiva verso gli individui simili.

Gli approcci possono essere molteplici. Si può ad esempio optare per meccanismi che classificano gli individui all'interno di nicchie ecologiche con risorse limitate [13]: individui simili andranno a occupare la stessa nicchia riducendo il numero di risorse per ognuno. Questo viene implementato diminuendo il valore di *fitness* degli individui presenti all'interno delle nicchie più affollate. Meccanismi più complessi possono introdurre una vera e propria speciazione [14] e monitorando l'andamento di ogni specie durante l'evoluzione, andando ad estinguere le specie meno adatte e preservando le meglio ottimizzate.



# Capitolo 2

## Algoritmo di ottimizzazione

### 2.1 Obiettivi

L'obiettivo dell'algoritmo proposto è quello di, dati una serie di nodi (vincolati o non) e carichi all'interno di uno spazio 2D, costruire la miglior struttura reticolare in grado di risolvere il problema valutando:

1. Gradi di libertà: la struttura non deve essere labile o presentare nodi disgiunti dal corpo principale.
2. Punti di snervamento: la tensione nelle aste non deve mai superare la tensione limite imposta, ottenuta dividendo la tensione di snervamento del materiale  $R_e$  per un opportuno coefficiente di sicurezza  $g_r$ .

$$\sigma_{ij} < \sigma_{amm} = \frac{R_e}{g_r} \quad (2.1)$$

3. Rottura per instabilità: il carico compressione non deve risultare maggiore del carico critico di Eulero ridotto di un opportuno coefficiente di sicurezza  $g_{r2}$  [1, eq. H.275]:

$$N_{ij} < N_{ij,cr} = \frac{\pi^2 EI_{min}}{\ell_{ij,0}^2 g_{r2}} \quad (2.2)$$

Dove:

- $E$  = modulo elastico del materiale
- $I_{min}$  = momento di inerzia minimo della sezione.
- $\ell_0$  = lunghezza libera di inflessione, pari a  $\ell_0 = \ell$  per travi incernierate a entrambi gli estremi [1, p. H-139].

4. Efficacia strutturale: capacità di utilizzare al meglio il materiale impegnato per sopportare i carichi. Per una singola asta questo si può definire come:

$$\eta_{ij} = \frac{|\sigma_{ij}|}{\sigma_{amm}} \quad (2.3)$$

5. Spostamento massimo: lo spostamento massimo di un nodo della struttura deve essere minore del valore imposto dal problema  $d_{max}$ .

## 2.2 Funzionamento

L'algoritmo è stato programmato in *Python* [15] affidandosi all'utilizzo della libreria *numpy* per la gestione delle matrici e dei vettori e a *matplotlib* [16] per la minima componente grafica.

Si è deciso di mantenere il codice il più *vanilla*<sup>1</sup> possibile riducendo al minimo l'utilizzo di librerie esterne per svolgere le principali operazioni in modo da poter avere un maggiore controllo sugli operatori genetici e sull'evoluzione dell'algoritmo.

Il diagramma di funzionamento dell'algoritmo è esposto in Figura 2.1; esso si compone di quattro fasi principali:

1. Generazione di una popolazione iniziale in modo casuale e inizio del processo evolutivo.
2. Calcolo della fitness e della fitness corretta (2.6) per ogni individuo della generazione.
3. Creazione della nuova generazione tramite gli operatori genetici (2.4). L'incidenza degli individui generati da ogni operazione rispetto alla popolazione totale è definito da relativi coefficienti imposti a inizio simulazione.
4. Verifica delle condizioni di uscita: se il numero di generazioni è maggiore di quello impostato oppure se è stato raggiunto il valore richiesto di *fitness*. Nel caso una delle due condizioni sia rispettata l'algoritmo termina; in caso contrario riesegue tutto a partire dal passo 2 con la nuova generazione creata al passo 3.

---

<sup>1</sup>Scrittura di programmi che non utilizzano librerie esterne o framework [17]

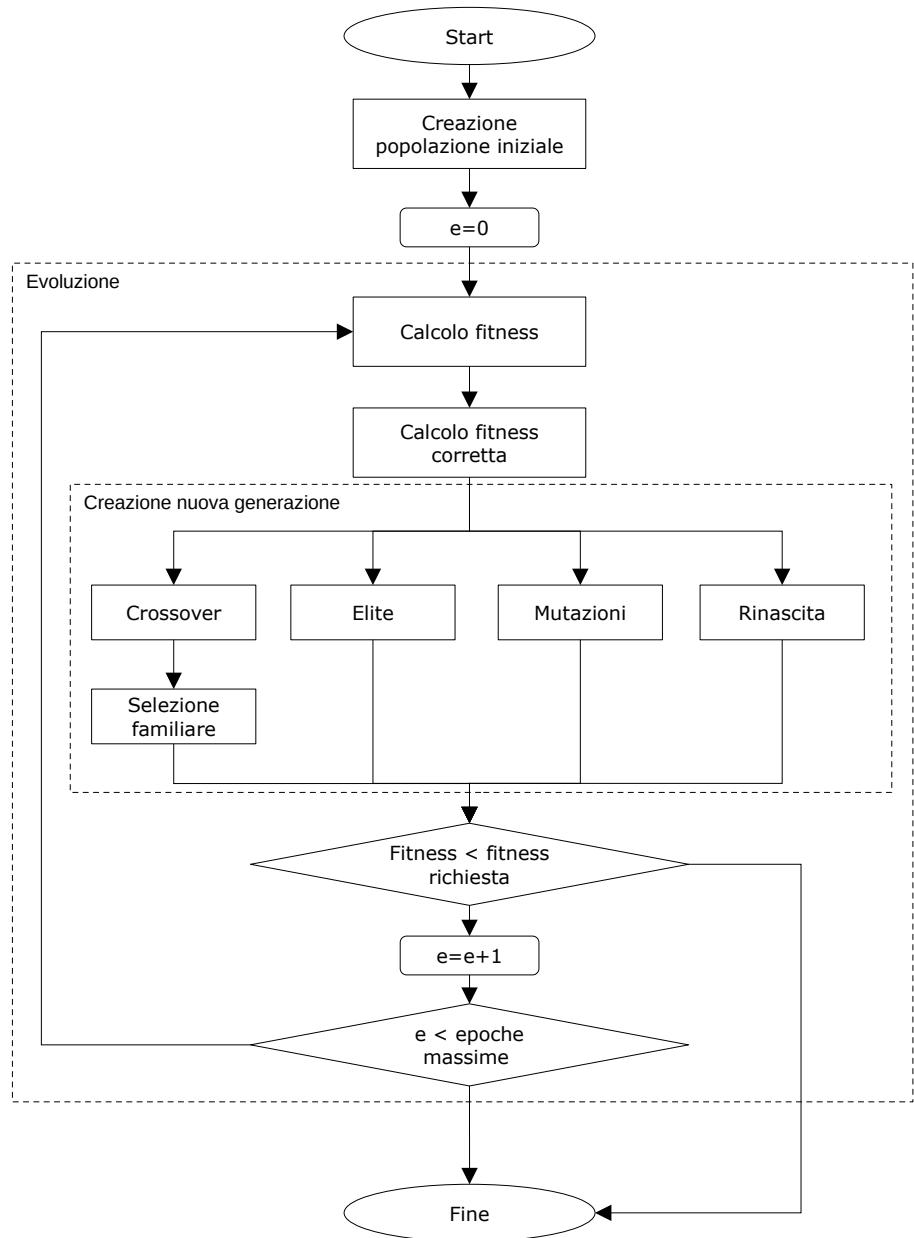


Figura 2.1: Schema di funzionamento

## 2.3 Rappresentazione dei dati

Come esposto in 1.2.1 all'interno di un algoritmo evolutivo esiste una fondamentale distinzione tra come l'individuo viene rappresentato (fenotipo) e come viene rappresentato a livello di dati (genotipo). Nel caso di una struttura reticolare gli elementi da codificare nel genotipo sono i nodi, le aste e i carichi esterni.

I nodi vengono rappresentati come vettori riga:

$$n_i = [x_i, y_i, u_i, v_i, R_{x,i}, R_{y,i}, P_{x,i}, P_{y,i}] \quad (2.4)$$

Dove:

- $x_i$  = coordinata x.
- $y_i$  = coordinata y.
- $u_i$  = spostamento orizzontale.
- $v_i$  = spostamento verticale.
- $R_{x,i}$  = reazione vincolare esterna orizzontale<sup>1</sup>.
- $R_{y,i}$  = reazione vincolare esterna verticale<sup>1</sup>.
- $P_{x,i}$  = carico esterno orizzontale.
- $P_{y,i}$  = carico esterno verticale.

I valori delle reazioni vincolari sono presenti solo se il nodo è bloccato in una o più direzioni. Allo stesso modo i carichi esterni sono valorizzati solo se imposti dal problema (Figura 2.2).

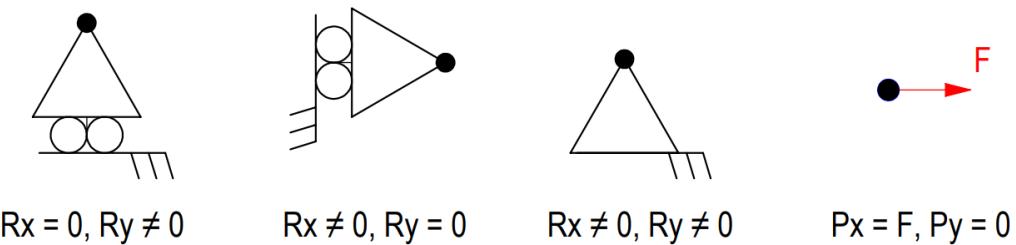


Figura 2.2: Esempi di rappresentazione di vincoli e carichi

<sup>1</sup>L'algoritmo non consente il calcolo di strutture con vincoli ad angolazioni diverse da 0 e 90 gradi

Le aste sono rappresentate tramite matrici d'adiacenza. Queste, utilizzate nella teoria dei grafi [18], consentono la codifica delle connessioni impostando il valore dell'elemento nell'intersezione tra la riga e la colonna aventi indici dei nodi alle estremità dell'asta in esame. Una struttura composta da  $n$  nodi verrà associata ad essa una matrice  $A$  di dimensione  $n \times n$  tale che, considerando i nodi  $i$  e  $j$ :

$$A(i, j) = \begin{cases} 1 & \text{se } i \text{ è connesso a } j \text{ da un asta} \\ 0 & \text{se } i \text{ e } j \text{ non sono collegati} \end{cases} \quad (2.5)$$

Si può facilmente dimostrare che la matrice di adiacenza in questo caso è sempre simmetrica: se il nodo  $i$  è collegato al nodo  $j$  e conseguentemente la cella  $A(i, j) = 1$  è ovvio che anche  $j$  è collegato a  $i$ , da cui  $A(i, j) = A(j, i) = 1$ . La diagonale invece risulta nulla poiché un nodo non può mai essere collegato a se stesso.

Ad ogni asta sono poi associati i relativi valori di lunghezza, area della sezione, carico, tensione ed efficienza strutturale. Si ottengono quindi altrettante matrici di adiacenza valorizzate come in Figura 2.6 (esempio relativo all'area).

$$A_A(i, j) = \begin{cases} A_{ij} & \text{se } i \text{ è connesso a } j \text{ da un asta} \\ 0 & \text{altrimenti} \end{cases} \quad (2.6)$$

Il genotipo di un individuo è quindi definito da  $n$  vettori nodo e 6 matrici di adiacenza come rappresentato in Figura 2.3.

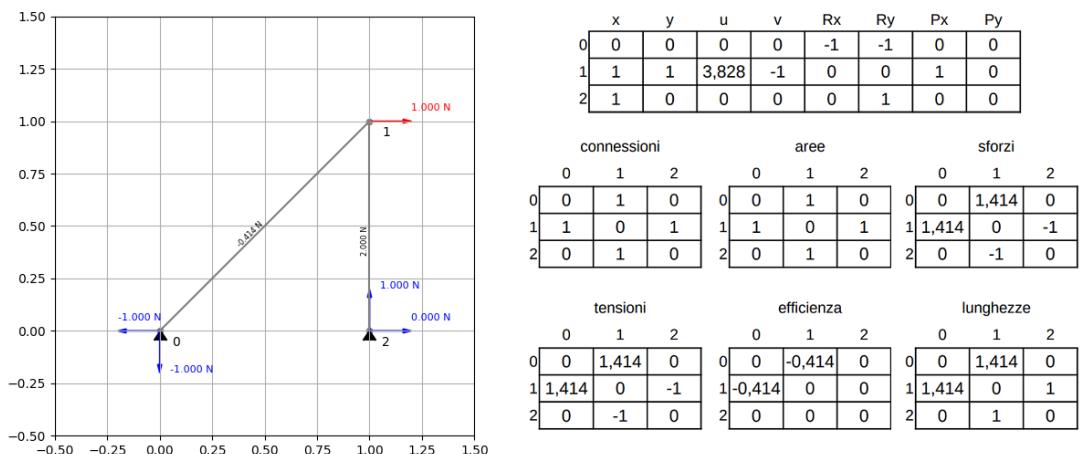


Figura 2.3: Esempio di struttura con relativo genotipo

## 2.4 Operatori genetici

### 2.4.1 Elitismo

Per elitismo si intende l'applicazione del meccanismo di riproduzione ai migliori individui di una generazione. Questo meccanismo se mantenuto in valori opportuni consente la preservazione dei genomi migliori.

Un valore troppo alto di individui trasportati può però portare al fallimento dell'ottimizzazione in quanto verrebbe notevolmente ridotta l'effetto dei restanti operatori genetici.

All'interno del programma, dato il coefficiente di elitismo  $K_e$  vengono trasportati da una generazione alla successiva i  $K_e \times N$  migliori individui ordinati secondo la fitness corretta ( $N$  = totale individui nella generazione).

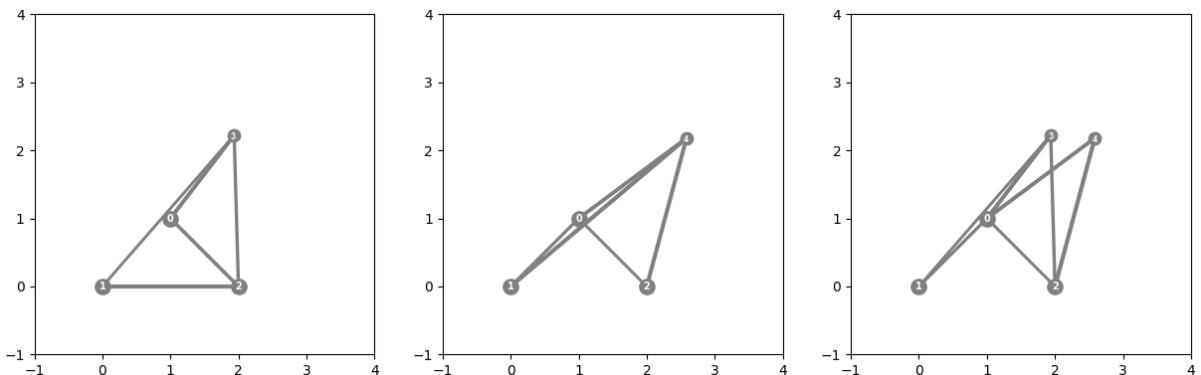
### 2.4.2 Crossover

Nell'operazione di crossover due individui genitori vengono uniti insieme al fine di creare un nuovo individuo figlio. E' stato preso come base il *crossover* utilizzato dall'algoritmo NEAT [14] vista la somiglianza delle strutture dati.

Il crossover avviene attraverso le seguenti fasi:

1. I due genitori vengono selezionati fra la popolazione tramite *K-turnament* (1.2.2). Poichè gli individui sono 2 questa prende il nome di *binary-turnamen*.
2. I nodi dei 2 individui genitori vengono allineati in un unico genoma. I nodi comuni a entrambi vengono considerati una volta sola. Dati quindi  $n_1$  e  $n_2$  rispettivamente nodi del primo e secondo genitore e  $n_c$  nodi comuni il nuovo genoma possiede  $k = n_1 + n_2 - n_c$  nodi. Per estrarre i nodi comuni vengono utilizzate le loro coordinate.
3. Per ogni genitore le matrici di adiacenza relative alle connessioni e alle aree vengono espansse alla dimensione  $k \times k$  in modo da essere compatibili con il genoma.
4. Le matrici di adiacenza dell'individuo figlio (sempre di dimensione  $k \times k$ ) vengono popolate prendendo i valori dalle matrici espansse dei genitori. La probabilità che un valore venga preso da uno o dall'altro genitore è definita sulla base della sua *fitness*.
5. Dal nuovo genoma dell'individuo figlio vengono eliminati i nodi liberi (ossia disgiunti da ogni asta) che non fanno parte dei vincoli iniziali del problema.
6. Nell'individuo figlio i nodi tra loro distanti meno di un valore  $R_n$  imposto dalla simulazione vengono uniti insieme.

7. Se i 2 genitori hanno lo stesso valore di *fitness* (quindi risultano molto simili o uguali) viene forzata una mutazione sull'individuo figlio.
8. Processo di selezione familiare: viene calcolato il valore di *fitness* del figlio e comparato con quello dei genitori. Fra i 3 individui vengono selezionati per la generazione successiva solo i 2 con la miglior *fitness*. Questo processo, comparabile all'elitismo, consente l'eliminazione di figli notevolmente meno performanti dei propri genitori.



(a) Fenotipo: da destra genitore 1, genitore 2 e figlio. I nodi imposti dal problema sono 0,1 e 2

	0	1	2	3	4
0	0	0	1	1	0
1	0	0	1	1	0
2	1	1	0	1	0
3	1	1	1	0	0
4	0	0	0	0	0

	0	1	2	3	4
0	0	1	1	0	1
1	1	0	0	0	1
2	1	0	0	0	1
3	0	0	0	0	0
4	1	1	1	0	0

	0	1	2	3	4
0	0	1	1	1	1
1	1	0	0	1	0
2	1	0	0	1	1
3	1	1	1	0	0
4	1	0	1	0	0

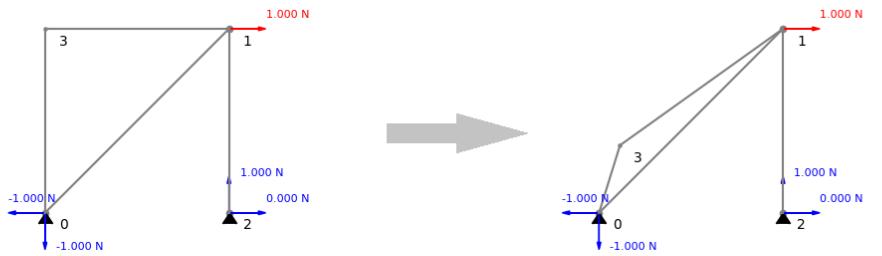
(b) Genotipo: nel genotipo figlio sono indicati in giallo e in arancio i caratteri presi rispettivamente dal genitore 1 e 2, i restanti non colorati sono comuni a entrambi

Figura 2.4: Operazione di crossover

### 2.4.3 Mutazioni

Nell'algoritmo sono stati implementati 5 tipi di mutazioni differenti controllate tramite altrettanti coefficienti  $K_{m,j}$ , il numero totale di mutazioni viene invece definito dal coefficiente  $K_m$  che rappresenta la probabilità che un individuo subisca una mutazione.

1. Spostamento di un nodo: vengono modificate in modo casuale le coordinate x e y di un nodo sempre scelto casualmente (ad esclusione dei nodi imposti come vincolo dal problema). Figura 2.5.



	x	y	u	v	Rx	Ry	Px	Py
0	0	0	0	0	-1	-1	0	0
1	1	1	3,828	-1	0	0	1	0
2	1	0	0	0	0	1	0	0
3	0	1	3,828	0	0	0	0	0

	x	y	u	v	Rx	Ry	Px	Py
0	0	0	0	0	-1	-1	0	0
1	1	1	3,828	-1	0	0	1	0
2	1	0	0	0	0	0	1	0
3	0,114	0,367	4,002	-1,243	0	0	0	0

Figura 2.5: Le coordinate del nodo 3 sono state modificate casualmente

2. Aggiunta di un nodo: viene aggiunto un nodo alla struttura, il punto di posizionamento è casuale.

Al fine di non generare una struttura labile il nodo viene collegato tramite 2 aste ad altri 2 diversi nodi scelti casualmente tra quelli già presenti. La sezione delle aste viene valorizzata in modo casuale (all'interno dei limiti imposti). Figura 2.6

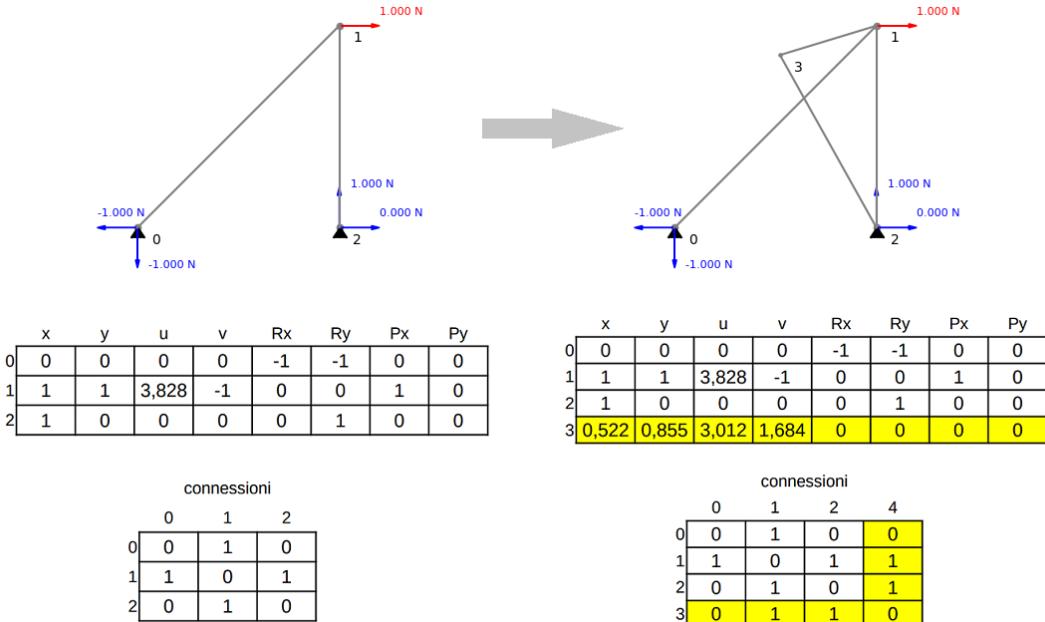


Figura 2.6: Il nodo 3 è stato aggiunto alla struttura e collegato con i nodi 1 e 2. Le dimensioni della matrice di adiacenza sono state aggiornate di conseguenza

3. Rimozione di un nodo: viene rimosso un nodo scelto casualmente (ad esclusione dei nodi

imposti come vincolo dal problema) e tutte le aste ad esso collegate. Processo inverso rispetto a quanto esposto in Figura 2.6.

4. Variazione sezione di un asta: la sezione di un'asta viene modificata casualmente all'interno dei limiti imposti.
  5. Variazione dello stato di una connessione: viene scelta una coppia di nodi, se questi risultano connessi da un'asta essa viene rimossa, in caso contrario aggiunta. Se viene aggiunta un'asta l'area viene valorizzata in modo casuale.
- Questo tipo di mutazione può essere invalidante in quanto la struttura risultante può essere labile. Per alcuni esempi si veda Figura 2.7

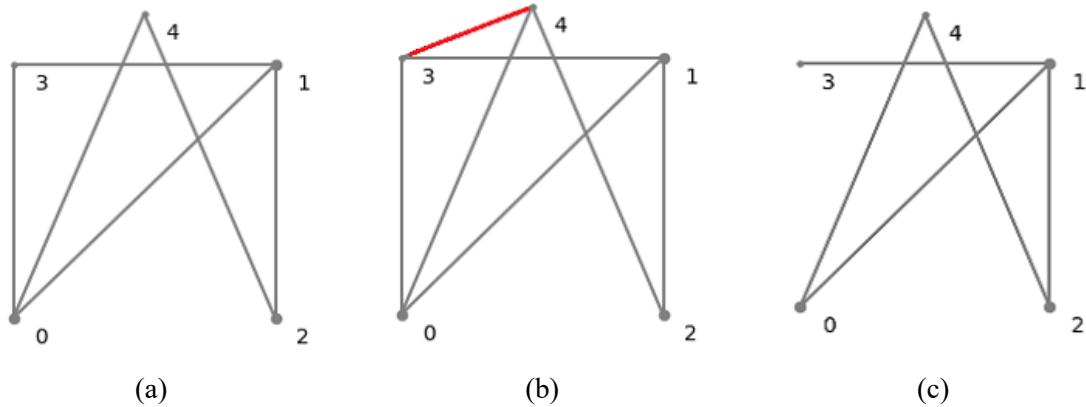


Figura 2.7: a) struttura originale b) aggiunta connessione tra nodi 3 e 4 (rosso) c) rimossa connessione tra nodi 1 e 3, struttura labile

## 2.5 Validità della struttura

Prima di poter calcolare la *fitness* un individuo è necessario verificare che questo rispetti alcuni requisiti:

1. Gradi di libertà: affinchè una struttura sia valida questa deve essere isostatica o iperstatica, dato quindi un numero  $n$  di nodi collegati da  $m$  aste e  $r$  vincoli esterni deve risultare valida:

$$2n - m - r \geq 0 \quad (2.7)$$

Tuttavia è possibile che l'algoritmo generi delle strutture non riconducibili a vere e proprie travature reticolari. In questi casi si ha la presenza di strutture isostatiche secondo la (2.7) che presentano però sotto-strutture mobili.

Al fine di identificare questi casi specifici viene calcolato il numero di aste  $o$  che collegano

due nodi aventi vincoli esterni e confrontato con il numero di vincoli sovrabbondanti. Una struttura risulta quindi staticamente valida se <sup>2</sup>:

$$\begin{cases} 2n - m - r > 0 \\ o \neq 0 \end{cases} \quad \text{se } 2n - m - r = 0 \quad (2.8)$$

Considerando l'esempio in Figura 2.8 si ha che  $2 \times 3 - 6 - 6 = 0$  (la struttura risulterebbe isostatica) ma  $o = 3$  poichè ci sono 3 aste che collegano tra di loro le cerniere 1,2 e 3. La struttura risulterà non valida.

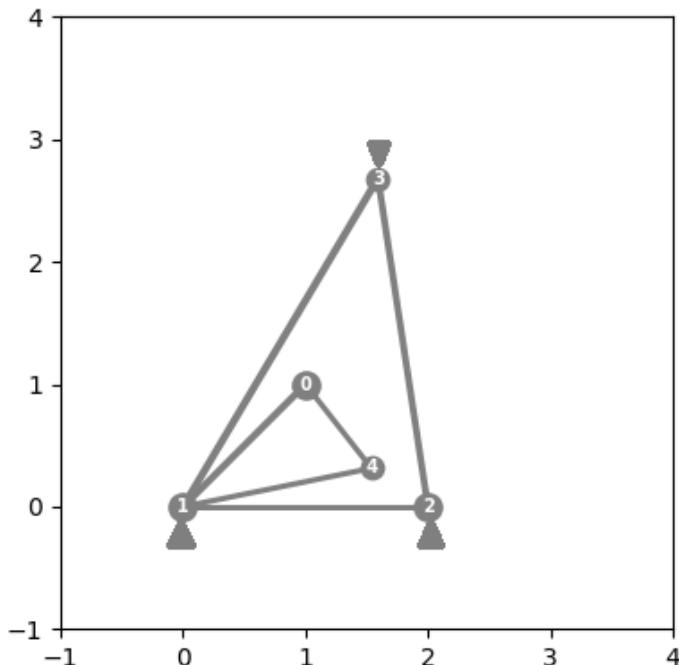


Figura 2.8: Esempio di struttura non propriamente reticolare. I nodi 1,2 e 3 sono cerniere. La sotto-struttura 1-0-4-1 può ruotare liberamente intorno a 1

2. Aste collineari: al fine di avere strutture costruttivamente valide l'algoritmo si occupa di verificare se esistono aste tra loro collineari.

Dati i nodi 1,2 e 3 si ha che 1 giace sul segmento passante tra 2 e 3 se:

$$D_{23} \approx D_{12} + D_{13} \quad (2.9)$$

Dove  $D_{ij}$  è la distanza tra i nodi  $i$  e  $j$ . Nell'equazione è utilizzato il vincolo di circa uguale poichè è definito un margine di collinearità e non una collinearità assoluta.

---

<sup>2</sup>Le condizioni proposte sono state ricavate per via puramente empirica e non viene fornita una dimostrazione matematica rigorosa

Se i nodi 2 e 3 sono tra loro collegati ed è presente almeno uno dei due altri collegamenti 1-2 e 2-3 allora si ha il caso di travi collineari. Una struttura avente anche solo una serie di aste collineari è considerata non valida

## 2.6 Funzione di fitness

La funzione di *fitness* è specifica per il problema analizzato. Per la sua definizione si è partiti dai seguenti presupposti:

1. L'obbiettivo primario è la riduzione di massa complessiva.
2. Le strutture non valide devono essere fortemente penalizzate in modo da non poter essere selezionate per le generazioni successive.
3. La struttura deve essere penalizzata se si rompe (trazione o buckling) o se supera la deformazione massima imposta dal problema  $d_{amm}$ .
4. Una struttura è tanto migliore quanto più alta è la sua efficienza strutturale.

E' stata quindi creata una funzione per casi che rispetti tali condizioni:

$$f = \begin{cases} 1 & \text{struttura non valida (2.5)} \\ 1 - 10^{-\sigma_{max}/\sigma_{amm}} & \text{se } \sigma_{max} > \sigma_{amm} \\ 1 - 10^{-d_{max}/d_{amm}} & \text{se } d_{max} > d_{amm} \\ 1 - (\eta_m + 0.1) & \text{nel resto dei casi} \end{cases} \quad (2.10)$$

Dove  $\eta_m$  è la media pesata dell'efficienza strutturale<sup>3</sup> di ogni asta rispetto alla sua massa:

$$\eta_m = \frac{\sum_{i=1}^n \frac{\sigma_i m_i}{\sigma_{amm,i}}}{\sum_{i=1}^n m_i} \quad (2.11)$$

La decisione dell'utilizzare la media pesata rispetto alla massa per il calcolo dell'efficienza complessiva nasce dal fatto che un ottimizzazione che considesse solo la massa portava più facilmente al blocco dell'algoritmo intorno ad un massimo locale. L'aggiunta del valore 0.1 all'efficienza strutturale consente di migliorare significativamente il valore delle strutture valide (non rientranti nei primi 3 casi dell'equazione 2.10) e di renderle più facilmente selezionabili per le generazioni successive.

---

<sup>3</sup>La presenza  $i$  all'interno della tensione ammissibile è conseguente al fatto che ogni asta ha la propria tensione ammissibile: la minima tra trazione e buckling

Si fa notare che la funzione di fitness è invertita rispetto a quanto presente nella maggioranza degli algoritmi: le strutture migliori hanno valori più bassi. E' stata scelta questa forma puramente per comodità nella scrittura del programma informatico.

## 2.7 Diversità genetica

### 2.7.1 Fitness corretta

Al fine di mantenere una corretta diversità genetica tra gli individui la *fitness* calcolata con la funzione (2.10) viene scalata sulla base del numero di individui simili.

Questo processo avviene in modo progressivo: al momento del calcolo della *fitness* dell'individuo  $i = k$  l'algoritmo conta tra gli individui precedenti  $i \in [0 : k - 1]$  quelli che possiedono una *fitness* compresa in un intorno  $r$  (di dimensione fissata dai parametri evolutivi) di quella dell'individuo  $i$ . Posto  $n$  il numero degli individui che soddisfano il criterio la fitness corretta viene calcolata come:

$$f_c = 2^n f \quad (2.12)$$

In questo modo si hanno 2 vantaggi:

- Individui simili a quelli già presenti hanno una probabilità notevolmente ridotta di essere selezionati per la riproduzione.
- Grazie al calcolo progressivo il primo individuo per intorno viene mantenuto con il suo valore di fitness originale. Avendo due individui  $i = k$  e  $i = v$  con  $k < v$  compresi stessa intorno di fitness l'individuo  $k$  al momento del calcolo avrà  $n = 0$  (l'individuo  $v$  non è ancora stato calcolato),  $v$  avrà invece  $n = 1$  poichè viene contata la presenza di  $k$ .

### 2.7.2 Random injection

Ad ogni generazione vengono introdotti nella nuova popolazione  $R = N \times K_r$  individui generati in modo casuale come la popolazione iniziale.  $K_r$  è il rapporto tra gli individui generati e la popolazione  $N$ .

Questo consente un apporto di nuovi genomi alla popolazione corrente e risulta utile quando l'algoritmo si trova in una situazione di stallo per causa della poca diversità genetica. Tuttavia il valore di  $K_r$  non deve risultare troppo elevato per evitare di saturare la popolazione con nuovi individui casuali e impedire all'algoritmo di convergere.

# Capitolo 3

## Analisi e risultati

### 3.1 Trave piana a 10 aste

E' stato preso come riferimento il modello più popolare nell'ambito dell'ottimizzazione strutturale di travi reticolari, esso è riportato in Figura 3.1. I parametri della struttura sono i seguenti [19]:

- Carico applicato ai nodi inferiori  $P = 445000 \text{ N}$
- Lunghezza unitaria  $\ell = 9.14 \text{ m}$
- densità  $\rho = 2.77 \times 10^3 \text{ kg/m}^3$
- Modulo elastico  $E = 6.9 \times 10^{10} \text{ N/m}^2$
- Tensione limite consentita  $\sigma_{amm} = 1.72^8 \text{ N/m}^2$
- Spostamento massimo consentito in direzione verticale  $v_{max} = \pm 5.08 \times 10^{-2} \text{ m}$

L'algoritmo è stato forzato a non verificare le aste a buckling in quanto nella letteratura attuale mancano informazioni sui parametri da utilizzare [20].

La simulazione comprende sia l'ottimizzazione della sezione delle aste sia delle posizioni dei nodi e connessioni tra essi.

Sono state svolte 10 simulazioni con i seguenti parametri:

- 2000<sup>1</sup> generazioni con 100 individui ciascuna.
- Coefficiente di mutazione totale pari a 0.1, i singoli tipi di mutazione sono invece distribuiti come 33% posizione nodi, 33% aree, 22% connessioni, 11% per inserimento nodi e 1% per eliminazione nodi.

---

<sup>1</sup>Un valore così alto consente alla funzione di fitness di stabilizzarsi sul valore migliore.

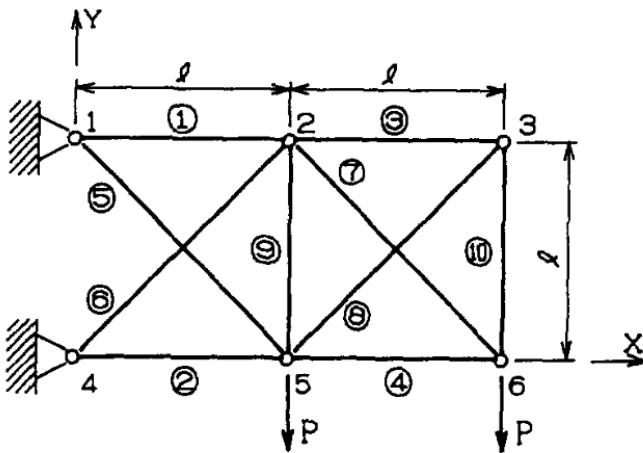


Figura 3.1: Problema delle 10 aste [21]

- Coefficiente di elitarismo 0.01 (un solo individuo viene riprodotto da una generazione alla successiva).
- Coefficiente di *random injection* pari a 0.05. Ogni generazione vengono generati 5 nuovi individui con genoma casuale.

n	fitness	massa [kg]	$d$ massimo [m]
1	0.7111	2478.02	0.05071
2	0.6785	2225.23	0.05054
3	0.6636	2483.63	0.05074
4	0.6769	2225.46	0.05079
5	0.7197	3227.51	0.05077
<b>6</b>	<b>0.6753</b>	<b>2214.13</b>	<b>0.05079</b>
7	0.7058	3180.43	0.05079
8	0.7113	2943.13	0.05069
9	0.7056	3049.03	0.05079
10	0.7126	3115.43	0.05061

Tabella 3.1: Risultati simulazioni. In grassetto è indicato il valore migliore

Per validare la simulazione è stato creata la struttura che ha avuto i migliori risultati (Tabella 3.1, n.6) all'interno del programma di modellazione 3D *Solidworks 2024* e analizzato utilizzando il modulo *Simulation* (questo presenta una specifica modalità per la risoluzione di travature reticolari [22]). Per semplicità le singole aste sono considerate come cilindriche piene.

La massa della struttura in *Solidworks* risulta pari a 2218.46 kg, con un errore minore dell 1% rispetto alla massa ottenuta dall'algoritmo di 2214.13 kg.

I valori per le singole aste sono riassunti in Tabella 3.2 e mostano una buona concordanza tra i

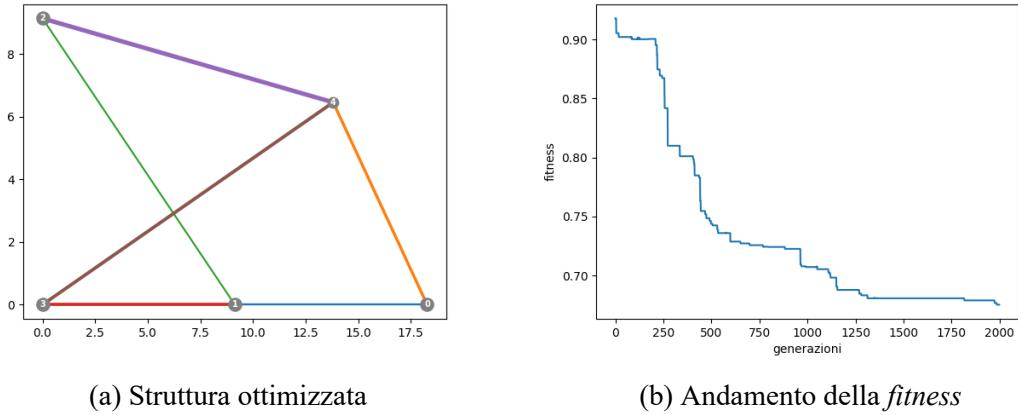


Figura 3.2: Risultati problema a 10 aste, miglior soluzione

valori. Fa eccezione lo spostamento massimo calcolato che supera di circa 1% quanto imposto dal problema (Figura 3.3,  $5.132 > 5.08$  cm), questo è sicuramente dovuto alla miglior precisione nel calcolo delle deformazioni da parte del FEM.

n	area [ $m^2$ ]	diametro [m]	forza assiale [ $10^5 N$ ]	tensione [ $10^7 Pa$ ]	forza assiale [ $10^5 N$ ]	tensione [ $10^7 Pa$ ]	err. forza assiale %	err. tensione %
0-1	0.0053	0.082	-3.08	-5.80	-3.08	-5.82	0.003	0.357
0-4	0.0112	0.119	5.41	4.83	5.41	4.86	0.000	0.701
1-2	0.0039	0.070	6.29	16.1	6.29	16.4	0.001	1.341
1-3	0.0124	0.126	-7.53	-6.07	-7.53	-6.04	0.001	0.553
2-4	0.0203	0.161	9.07	4.47	9.07	4.45	0.000	0.286
3-4	0.0140	0.134	-6.43	-4.59	-6.43	-4.56	0.002	0.727

Tabella 3.2: Valori delle aste della miglior simulazione

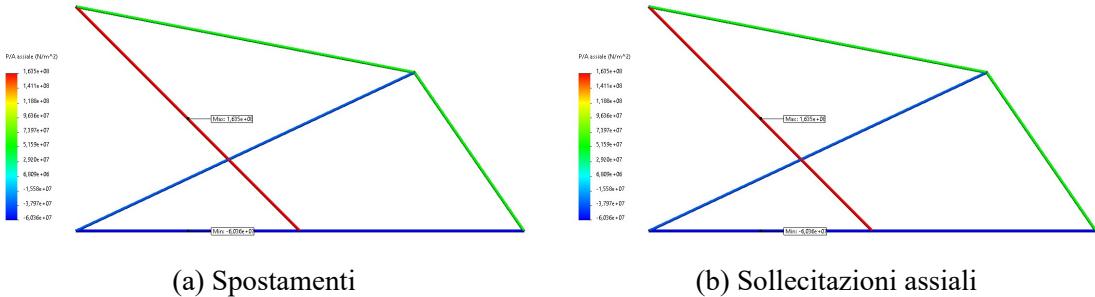


Figura 3.3: Risultati simulazione Solidworks

Dalla Tabella 3.3 si può notare come l'algoritmo proposto risulta in linea con i risultati presenti nell'attuale letteratura. Questo consente di affermare che l'algoritmo evolve correttamente verso la struttura migliore rispettando i vincoli imposti dal problema.

	massa [kg]
Aminifar, Aminifar, 2013 [19]	2261.42
E. Perez, Behdi, 2007 [23]	2278.55
Galante, 1996 [24]	2492.15
Petrovi, 2017 [20]	2257.24
Frans, Arfiadi, 2014 [25]	2122.62
Mortazavi, Toğan, 2016 [26]	2704.80
<b>Questo elaborato</b>	<b>2214.13</b>

Tabella 3.3: Confronto dei risultati con quelli presenti in letteratura

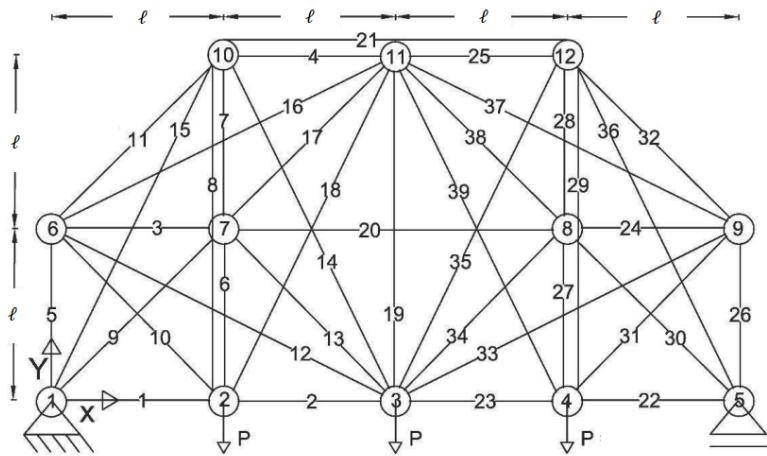


Figura 3.4: Problema a 39 aste [28]

## 3.2 Trave piana a 39 aste

Il secondo caso studio proposto è rappresentato i Figura 3.4. I parametri del problema sono [27]:

- Carico applicato ai nodi inferiori  $P = 88964.43 \text{ N}$
- Lunghezza unitaria  $\ell = 3.048 \text{ m}$
- Densità  $\rho = 2.77 \times 10^3 \text{ kg/m}^3$
- Modulo elastico  $E = 6.9 \times 10^{10} \text{ N/m}^2$
- Tensione limite consentita  $\sigma_{amm} = 1.378^8 \text{ N/m}^2$
- Spostamento massimo consentito in direzione verticale  $v_{max} = \pm 5.08 \times 10^{-2} \text{ m}$
- Area limite  $A \in [1.45; 0.0032] \times 10^{-3} \text{ m}$

E' stata eseguita sia l'ottimizzazione della sezione delle aste sia delle posizioni dei nodi e connessioni tra essi. Sono state svolte 10 simulazioni con i seguenti parametri:

- $2000^2$  generazioni con 200 individui ciascuna.
- Coefficiente di mutazione totale pari a 0.25, i singoli tipi di mutazione sono invece distribuiti come 11% posizione nodi, 55% aree, 22% connessioni, 11% per inserimento nodi e 1% per eliminazione nodi.
- Coefficiente di elitarismo 0.01 (un solo individuo viene riprodotto da una generazione alla successiva).
- Coefficiente di *random injection* pari a 0.1. Ogni generazione vengono generati 20 nuovi individui con genoma casuale.

n	fitness	massa [kg]	$d$ massimo [m]
1	<b>0.8155</b>	<b>106.15</b>	<b>0.0394</b>
2	0.8158	116.84	0.0404
3	0.8150	123.38	0.0424
4	0.8235	125.31	0.0349
5	0.8186	115.81	0.0403
6	0.8182	113.42	0.0394
7	0.8229	123.53	0.0366
8	0.8213	131.38	0.0399
9	0.8181	109.77	0.0385
10	0.8180	111.10	0.0386

Tabella 3.4: Risultati simulazioni. In grassetto è indicato il valore migliore

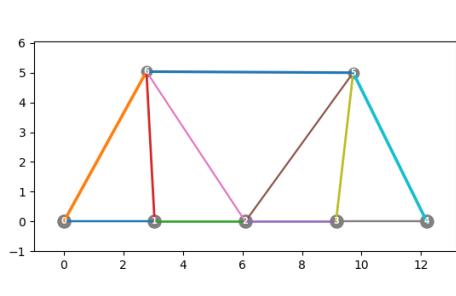
Confrontando i risultati ottenuti con la letteratura (Tabella 3.5) emerge come l'algoritmo non riesca ad ottimizzare perfettamente la struttura ottenendo risultati superiori di una media di circa 50%. Tale inefficienza è quasi sicuramente dovuta ad una non perfetta combinazione di parametri evolutivi e per risolvere il problema risulterebbe necessario un gran numero di simulazioni in modo da isolare e analizzare dettagliatamente e in modo statistico l'effetto specifico di ogni coefficiente<sup>3</sup>.

Dalle strutture generate emergono però dei risultati interessanti: l'algoritmo tende fin dalle prime generazioni a prediligere strutture il più simmetriche possibile, questo avviene in assenza di alcuna imposizione di simmetria (Figura 3.6) come vincolo del problema ma semplicemente tramite il processo di evoluzione. Si può quindi affermare che il funzionamento dell'algoritmo in se risulta comunque validato, ma migliorabile in alcuni aspetti (4).

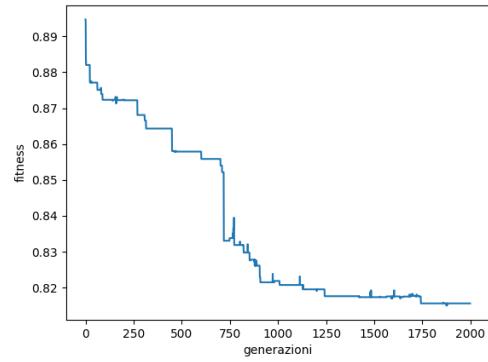
---

<sup>2</sup>Un valore così alto consente alla funzione di fitness di stabilizzarsi sul valore migliore.

<sup>3</sup>Dalle prove eseguite è emerso come la variazione minima di un coefficiente di mutazione possa portare a riduzioni di massa di oltre 50/80 kg



(a) Struttura ottimizzata



(b) Andamento della fitness

Figura 3.5: Risultati problema a 39 aste, miglior soluzione

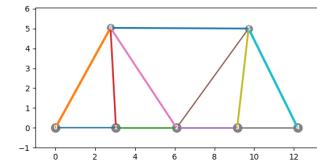
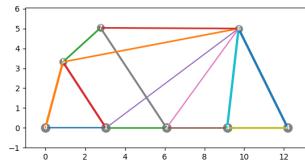
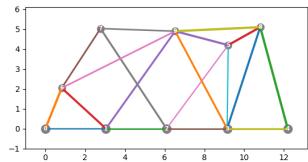


Figura 3.6: Esempio di soluzioni campionate durante il processo di evoluzione (da sx rispettivamente a 200, 400 e 600 generazioni), si può notare come l'algoritmo tenda alla simmetria

	massa [kg]
Deb, Gulati, 2001 [27]	87.16
Mortazavi, Toğan, 2016 [26]	82.37
Harsono, Prayogo, Prasetyo et al., 2020 [29]	85.21
Tejani, Savsani, Patel et al., 2018 [30]	86.22
Dang, Nguyen-Van, Thai et al., 2022 [31]	86.76
<b>Questo elaborato</b>	<b>106.15</b>

Tabella 3.5: Confronto dei risultati con quelli presenti in letteratura

# Capitolo 4

## Conclusioni

L'algoritmo presentato è stato implementato seguendo un approccio didattico al fine di analizzare i singoli operatori genetici tipici di un algoritmo evolutivo e i loro effetti; di conseguenza l'efficacia generale è stata considerata in modo marginale.

Perciò in seguito delle simulazioni svolte sono stati individuati i seguenti aspetti da migliorare in una futura versione dell'algoritmo:

- Generazione della popolazione iniziale: la popolazione iniziale viene generata in modo totalmente casuale a partire dai dati del problema, i vincoli imposti sono il numero massimo di nodi generabili e la condizione che il numero di connessioni tra i nodi permetta alla struttura di essere al più isostatica (1.1).

Si è notato che la popolazione iniziale ha un forte impatto sull'andamento dell'evoluzione: meglio gli individui sono diversificati all'inizio più facilmente l'algoritmo converge. Per consentire quindi un miglioramento delle *performance* si potrebbe implementare un'inizializzazione della popolazione più complessa ad esempio distribuendo i nodi secondo *pattern* prestabiliti invece che casualmente.

- Meccanismi di speciazione complessi: all'interno dell'algoritmo è inserito un semplice sistema di controllo della popolazione tramite correzione della *fitness* (2.7); esistono in letteratura meccanismi più complessi [12] [14] [32] che consentono di garantire una migliore diversità genetica e conseguentemente un algoritmo più performante.
- Parallelizzazione del calcolo<sup>1</sup>: gli algoritmi evolutivi si prestano al calcolo parallelo in quanto si possono generare  $n$  popolazioni da far evolvere simultaneamente e in modo indipendente per poi andare ad estrarre al termine il miglior risultato. Un'altra alternativa potrebbe essere che, dopo un numero fissato di generazioni, tutte le  $n$  popolazioni vengono mescolate e l'evoluzione riprende.

---

<sup>1</sup>esecuzione simultanea del codice sorgente di uno o più programmi [33]

- Ottimizzazione multi obiettivo: una struttura reticolare può essere ottimizzata considerando aspetti differenti: efficienza rispetto ai carichi, massa complessiva, volume totale occupato, ecc... . L'algoritmo esposto in questo elaborato per calcolare la *fitness* di un individuo unisce tramite media ponderata l'efficienza strutturale e di massa. Analizzare i due parametri in modo indipendente, ad esempio attraverso la ricerca dell'ottimo Paretiano [9], potrebbe portare ad una soluzione migliore del problema [34].

# Bibliografia

- [1] L. Caligaris, S. Fava e C. Tommasello, *Manuale di meccanica*, 2<sup>a</sup> ed. Hoepli, 3 gen. 2016, ISBN: 978-88-203-6645-2.
- [2] *Ikitsuki Bridge*, in *Wikipedia*.
- [3] Z. Liang. «5 interessanti strutture reticolari nel mondo | SkyCiv Engineering.» (), indirizzo: <https://skyciv.com/it/industry/5-interesting-truss-structures-in-the-world/> (visitato il 08/09/2024).
- [4] «Integrated Truss Structure,» NASA official site. (), indirizzo: <https://www.nasa.gov/international-space-station/integrated-truss-structure/> (visitato il 08/09/2024).
- [5] «Building Hubble,» NASA Science. (), indirizzo: <https://science.nasa.gov/gallery/hubble-under-development/> (visitato il 08/09/2024).
- [6] U. Galvanetto, *Appunti Ed Esercizi Sul Metodo Degli Spostamenti Applicato Ai Sistemi Piani Di Travi*. Libreria Progetto Padova, 2010, ISBN: 9000000000074.
- [7] C. Darwin 1809-1882, *On the Origin of Species by Means of Natural Selection, or Preservation of Favoured Races in the Struggle for Life*. London : John Murray, 1859, 1859.
- [8] J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence* (Complex Adaptive Systems), 1st MIT Press ed. Cambridge, Mass: MIT Press, 1992, 211 pp., ISBN: 978-0-262-08213-6.
- [9] *Ottimo paretiano*, in *Enciclopedia Treccani*.
- [10] G. Folino, «Algoritmi evolutivi e programmazione genetica: strategie di progettazione e parallelizzazione,» Consiglio Nazionale delle Ricerche Istituto di Calcolo e Reti ad Alte Prestazioni, RT-ICAR-CS-03-17, 2003.
- [11] D. Jong e K. Alan, «Analysis of the behavior of a class of genetic adaptive systems,» 1975.

- [12] B. Miller e M. Shaw, «Genetic Algorithms with Dynamic Niche Sharing for Multimodal Function Optimization,» in *Proceedings of IEEE International Conference on Evolutionary Computation*, Nagoya, Japan: IEEE, 1996, pp. 786–791, ISBN: 978-0-7803-2902-7.
- [13] K. Deb e D. E. Goldberg, «An Investigation of Niche and Species Formation in Genetic Function Optimization,» in *Proceedings of the 3rd International Conference on Genetic Algorithms*, San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1 giu. 1989, pp. 42–50, ISBN: 978-1-55860-066-9.
- [14] K. O. S. a. R. Miikkulainen, «Evolving Neural Networks Through Augmenting Topologies,» 2002.
- [15] G. Van Rossum e F. L. Drake, *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace, 2009, ISBN: 1-4414-1269-7.
- [16] J. D. Hunter, «Matplotlib: A 2D Graphics Environment,» *Computing in Science & Engineering*, vol. 9, n. 3, pp. 90–95, 2007.
- [17] mcsavvy. «Programming Languages: What the heck does “Vanilla” mean?» Medium. (), indirizzo: <https://medium.com/@mcsavvy/programming-languages-what-the-heck-does-vanilla-mean-cbbf81031756> (visitato il 04/09/2024).
- [18] R. Diestel, *Graph Theory* (Graduate Texts in Mathematics 173), 2. ed. New York, NY: Springer, 2000, 312 pp., ISBN: 978-0-387-98976-1.
- [19] F. Aminifar e F. Aminifar, «Optimal design of truss structures via an augmented genetic algorithm,» *Turkish Journal of Engineering & Environmental Sciences*, gen. 2013.
- [20] N. Petrovi, «COMPARISON OF APPROACHES TO 10 BAR TRUSS STRUCTURAL OPTIMIZATION WITH INCLUDED BUCKLING CONSTRAINTS,» 2017.
- [21] T. Yokota, T. Taguchi e M. Gen, «A solution method for optimal weight design problem of 10 bar truss using genetic algorithms,» *Computers & Industrial Engineering*, vol. 35, n. 1-2, pp. 367–372, ott. 1998, ISSN: 03608352.
- [22] «Trusses - 2024 - SOLIDWORKS Help.» (), indirizzo: [https://help.solidworks.com/2024/english/SolidWorks/cworks/c\\_trusses.htm?verRedirect=1](https://help.solidworks.com/2024/english/SolidWorks/cworks/c_trusses.htm?verRedirect=1) (visitato il 02/09/2024).
- [23] R. E. Perez e K. Behdi, «Particle Swarm Optimization in Structural Design,» in *Swarm Intelligence, Focus on Ant and Particle Swarm Optimization*, F. T.S., cur., I-Tech Education and Publishing, 1 dic. 2007, ISBN: 978-3-902613-09-7.

- [24] M. Galante, «GENETIC ALGORITHMS AS AN APPROACH TO OPTIMIZE REAL-WORLD TRUSSES,» *International Journal for Numerical Methods in Engineering*, vol. 39, n. 3, pp. 361–382, 15 feb. 1996, ISSN: 0029-5981, 1097-0207.
- [25] R. Frans e Y. Arfiadi, «Sizing, Shape, and Topology Optimizations of Roof Trusses Using Hybrid Genetic Algorithms,» *Procedia Engineering*, vol. 95, pp. 185–195, 2014, ISSN: 18777058.
- [26] A. Mortazavi e V. Toğan, «Simultaneous size, shape, and topology optimization of truss structures using integrated particle swarm optimizer,» *Structural and Multidisciplinary Optimization*, vol. 54, n. 4, pp. 715–736, ott. 2016, ISSN: 1615-147X, 1615-1488.
- [27] K. Deb e S. Gulati, «Design of truss-structures for minimum weight using genetic algorithms,» *Finite Elements in Analysis and Design*, vol. 37, n. 5, pp. 447–465, mag. 2001, ISSN: 0168874X.
- [28] *Truss Topology Optimization: A Review Past, Present, and Future*, 1. Auflage. Saarbrücken: Scholars' Press, 2018, ISBN: 978-620-2-31370-4.
- [29] K Harsono, D Prayogo, K. E. Prasetyo, F. T. Wong e D Tjandra, «Comparative Study of Particle Swarm Optimization Algorithms in Solving Size, Topology, and Shape Optimization,» *Journal of Physics: Conference Series*, vol. 1625, n. 1, p. 012015, 1 set. 2020, ISSN: 1742-6588, 1742-6596.
- [30] G. G. Tejani, V. J. Savsani, V. K. Patel e P. V. Savsani, «Size, shape, and topology optimization of planar and space trusses using mutation-based improved metaheuristics,» *Journal of Computational Design and Engineering*, vol. 5, n. 2, pp. 198–214, 1 apr. 2018, ISSN: 2288-5048.
- [31] K. D. Dang, S. Nguyen-Van, S. Thai, S. Lee, V. H. Luong e Q. X. Lieu, «A single step optimization method for topology, size and shape of trusses using hybrid differential evolution and symbiotic organisms search,» *Computers & Structures*, vol. 270, p. 106 846, ott. 2022, ISSN: 00457949.
- [32] A. Sharma, «A New Optimizing Algorithm Using Reincarnation Concept,» in *2010 11th International Symposium on Computational Intelligence and Informatics (CINTI)*, nov. 2010, pp. 281–288.
- [33] *Calcolo parallelo*, in *Wikipedia*.
- [34] S. D. Rajan, «Sizing, Shape, and Topology Design Optimization of Trusses Using Genetic Algorithm,» *Journal of Structural Engineering*, vol. 121, n. 10, pp. 1480–1487, ott. 1995, ISSN: 0733-9445, 1943-541X.