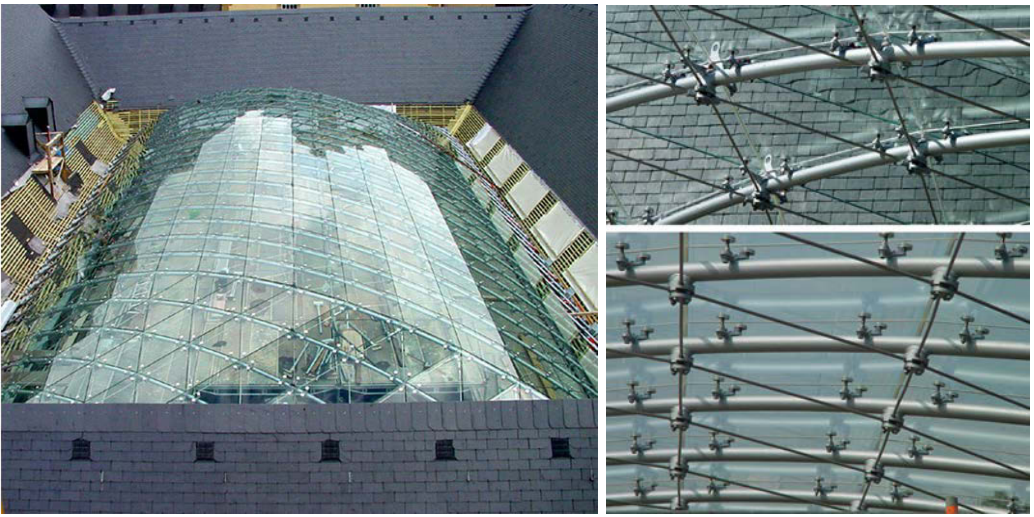
FORM FINDIG E OTTIMIZZAZIONE DEL BUCKLING DI GRID SHELLS CON L'UTILIZZO DI ALGORITMI GENETICI

GRID SHELL /// 01

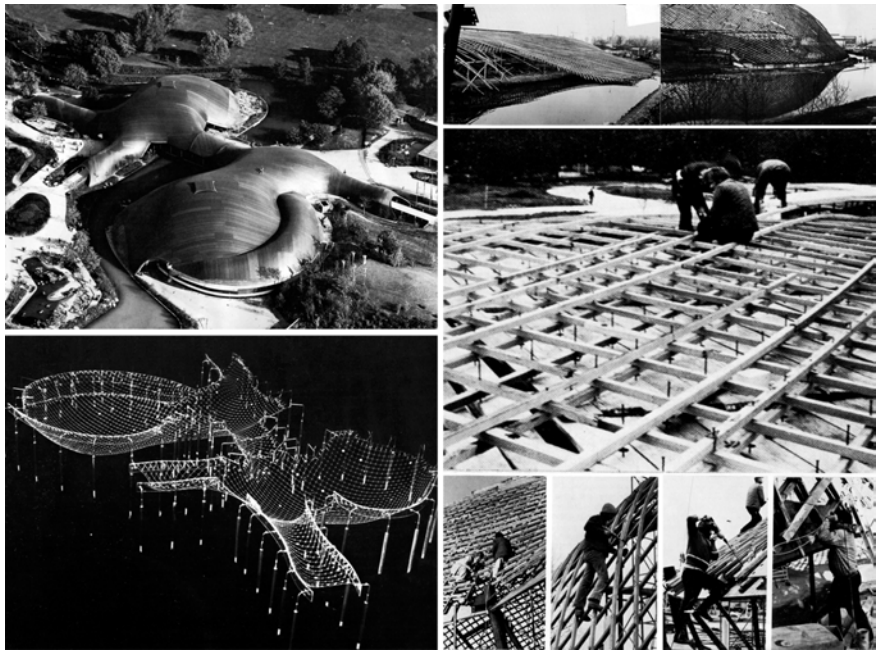
Schlaich Bergermann und Parterners, Swimming Bath Aquatoll, Neckarsulm, 1989

“Una grid shell (gitterschale in tedesco) è una struttura di barre, curva nello spazio. Le barre formano una griglia piana con maglia rettangolare e distanza costante tra ciascun nodo. La forma della grid shell è ottenuta per inversione di una rete sospesa. Nel modo in cui una griglia sospesa dà la curva ideale di un arco senza flessione, così l’inversione della rete conduce ad una forma funicolare nella quale il grid shell non presenta flessione.”

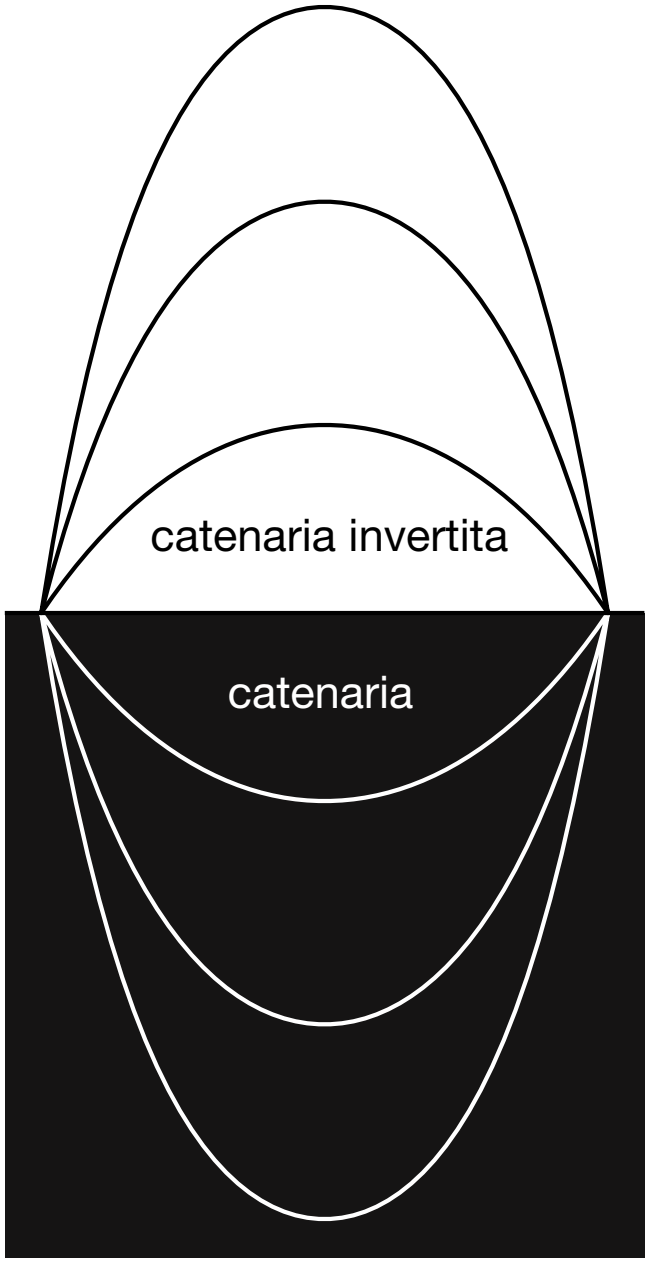
FREI OTTO



RFR, Copertura del chiostro dell’abbazia di Neumunster, Neumunster, 2002



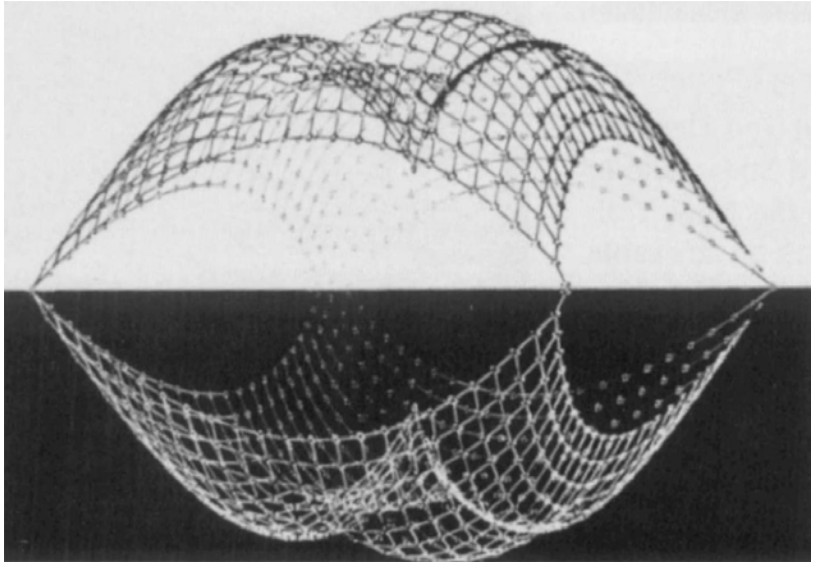
Frei Otto, Multihalle Exhibition Garden, Mennheim, 1968



FORM-FINDING /// 02

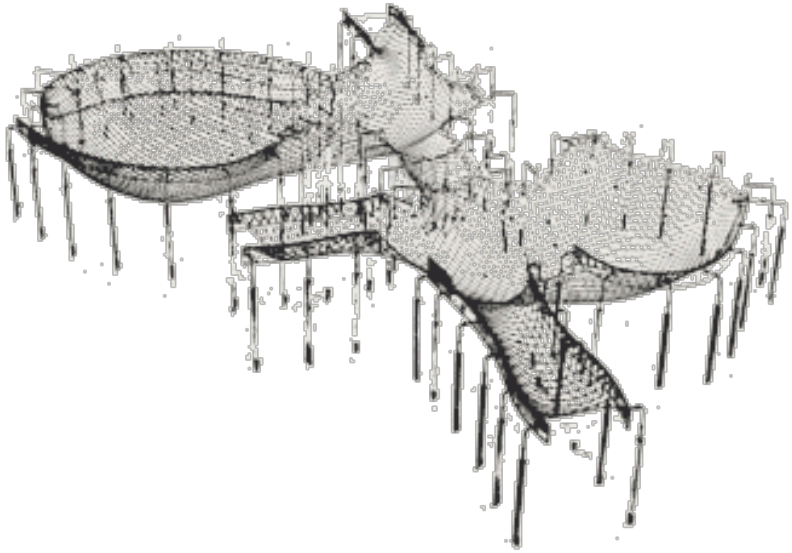
La catenaria è la curva secondo cui si dispone una fune che supponiamo omogenea, flessibile e non estendibile, appesa a due punti estremi, che sia lasciata pendere soggetta soltanto al proprio peso. Invertendo tale forma, in considerazione del fatto che una catenaria ha la proprietà di avere in ogni suo punto una distribuzione uniforme del suo peso totale, si ottiene una geometria resistente per forma.

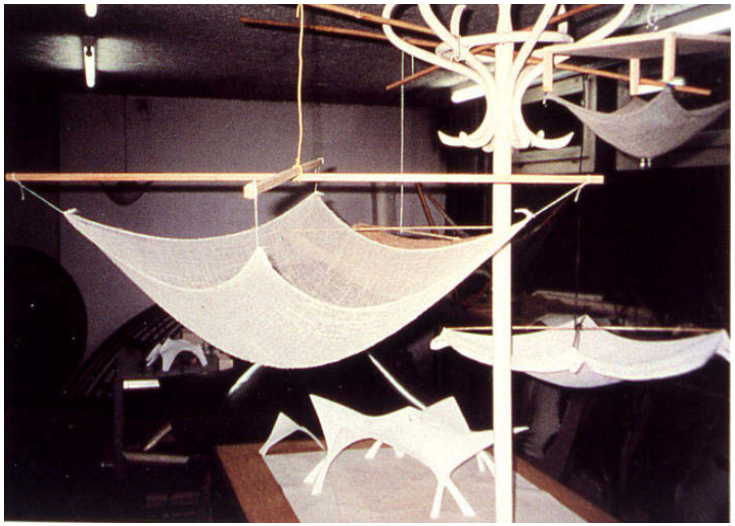
Lo stesso principio può essere adottato per ottenere delle geometrie più complesse. Partendo da una maglia piana deformabile, vincolata opportunatamente, sottoposta alla forza di gravità, si ottiene una geometria che, se invertita, corrisponde alla funicolare del sistema di forze relativo alla particolare combinazione di carichi introdotta nel modello.



FORM-FINDING /// 03

Antonio Gaudi, Colonia Güell

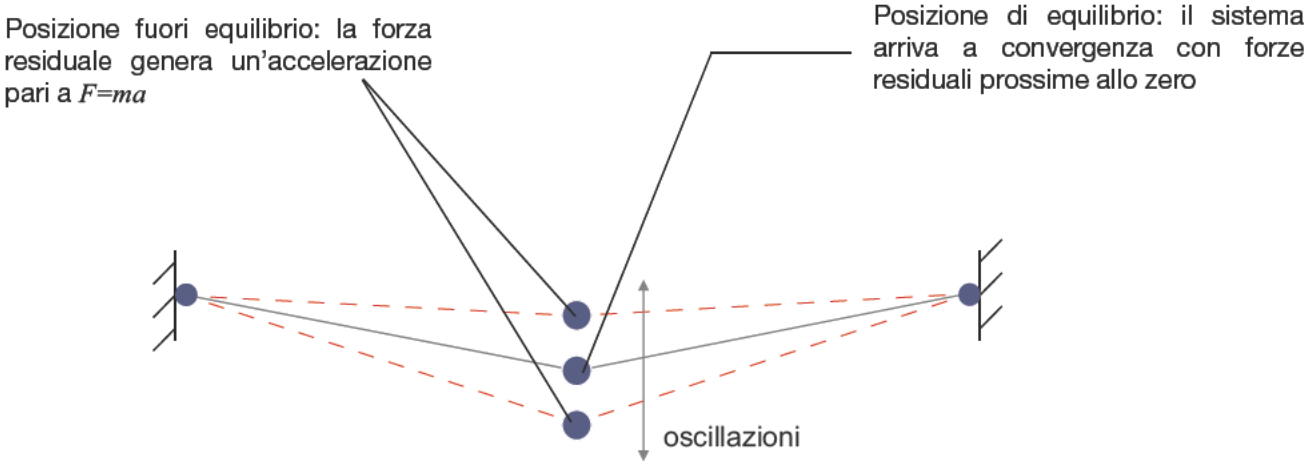




Heinz Isler: modelli fisici per l'inversione delle membrane tese (sinistra)
Deitingen Service Station, Deitingen, Svizzera, 1968 (destra)

Frei Otto, Multihalle Exhibition Garden, Mennheim, 1968

Dynamic Relaxation Method



Il modello si muove sotto l'azione delle forze fuori equilibrio fino a che l'equilibrio non viene raggiunto

Il metodo ricerca l'equilibrio statico attraverso uno schema iterativo che simula un processo pseudo-dinamico nel tempo. L'equazione su cui si basa è quella del moto per un mezzo discreto:

$$P_{ij} = \left[\sum K\delta\right]_{ij} + M_{ij}\ddot{\delta}_{ij} + C\dot{\delta}_{ij}$$

Si definisco le forze residue come differenza fra forze esterne e interne:

$$R_{ij} = P_{ij} - \left[\sum K\delta\right]_{ij}$$

quindi si può scrivere l'equazione 1 come:

$$R_{ij} = M_{ij}\ddot{\delta}_{ij} + C\dot{\delta}_{ij}$$

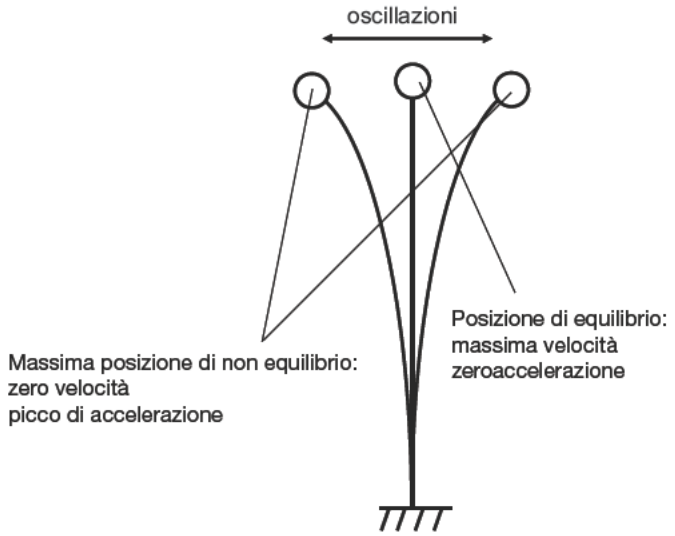
L'equazione 3 indica come il movimento del sistema è indotto dal sistema di forze non equilibrate e che l'equilibrio statico dello stesso richiede l'annullarsi di queste forze nodali residue.

Il processo è definito alle differenze finite e non richiede accelerazioni o spostamenti iniziali per iniziare in quanto questi sono definite dalle forze residue.

SMORZAMENTO

Al sistema viene applicato uno smorzamento per prevenire le oscillazioni continue. Lo smozamento può essere di due tipi:

- Smorzamento viscoso
- Smorzamento cinetico: l'oscillazione viene interrotta ad ogni picco di energia cinetica in analogia con il pendolo semplice.



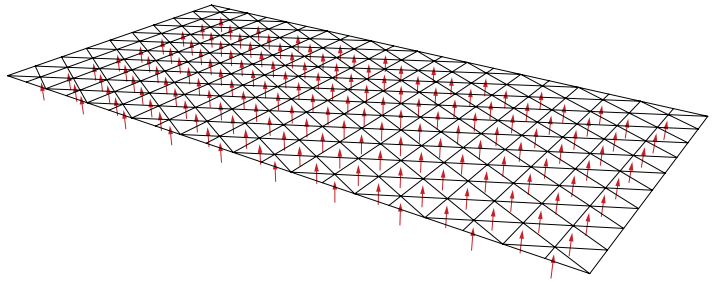
VANTAGGI

Il metodo del rilassamnto dinamico utilizza direttamente i termini di rigidezza nodale senza dover riassemblare la matrice di rigidezza ad ogni iterazione quindi si hanno vantaggi in termini di velocità e stabilità di convergenza dell'algoritmo. Attraverso questo metodo si possono condurre analisi non lineari con un notevole risparmio in termini di tempo-macchina.

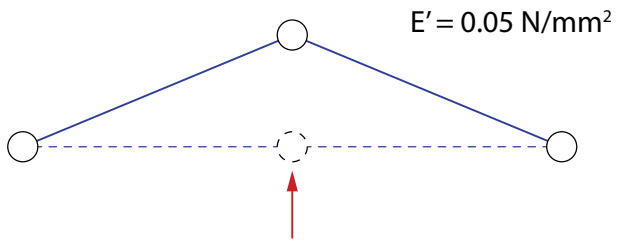
Prestress Force Form-Finding

Il Prestress Force Form-Finding è una metodologia utilizzata dallo studio RFR per la progettazione della copertura della corte dell'abbazia di Neumunster. Il processo si può riassumere nei seguenti punti:

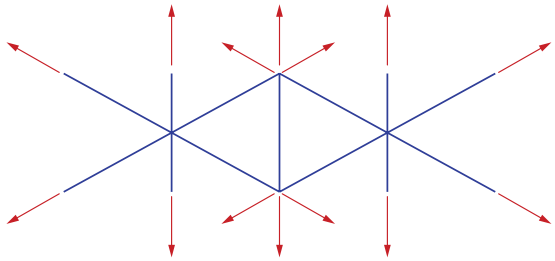
1. Definizione della griglia strutturale piana e dei carichi nodali



2. Definizione degli elementi e di un modulo di elasticità ridotto

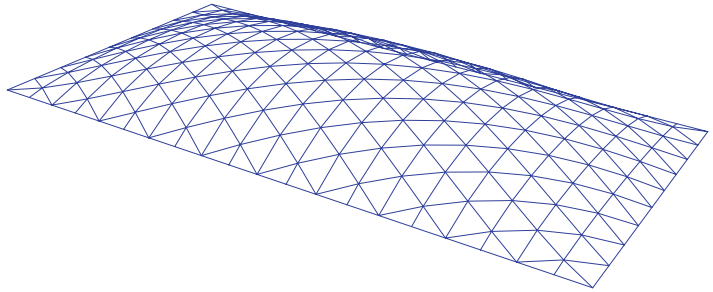


3. Definizione della pretensione negli elementi



4. Analisi della struttura attraverso il solutore *GSARelax* del software *Oasys GSA*

5. Generazione della nuova struttura e della condizione di carico che permette l'equilibrio

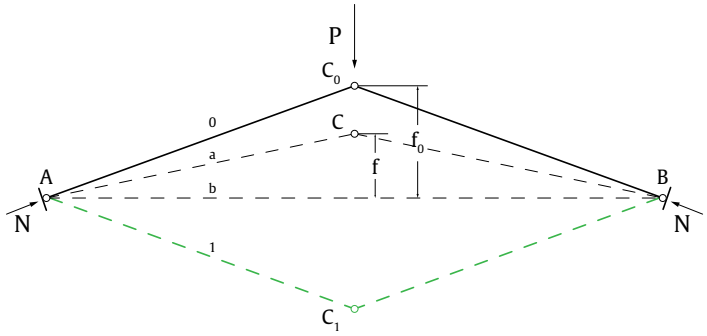


Le grid shells sono strutture che fanno fronte ai carichi esterni principalmente attraverso sforzi di compressione. Pertanto la verifica della stabilità risulta, spesso, il parametro dominante per la progettazione strutturale di queste opere.

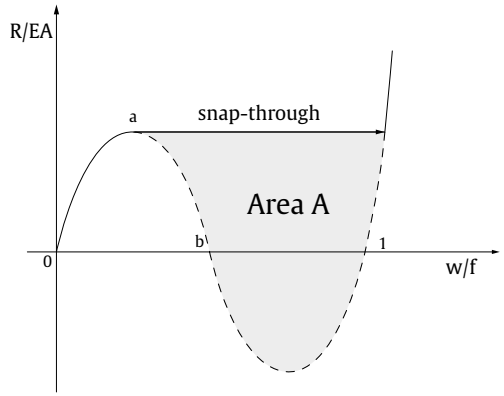
Le tipologie di collasso per instabilità delle grid shells si possono riassumere come segue [Bulenda Th., Knippers J., 2001]:

- instabilità del singolo elemento: si instabilizza un solo elemento senza coinvolgere il resto della struttura;
- instabilità locale: il fenomeno di snap-through;
- instabilità globale: si instabilizza tutta la struttura. Questa modalità di collasso è paragonabile al buckling globale di un guscio continuo;
- combinazioni dei precedenti modi: le grid shell sono ottimizzate nei confronti dei carichi che portano a diverse modalità di collasso. Questo vuol dire che varie modalità di collasso possono avvenire per lo stesso livello di carico. Si deve quindi prestare attenzione anche all'interazione di modi che di per sé non sono pericolosi ma che combinandosi possono portare al collasso della struttura.

SNAP THROUGH



Lo snap through è un fenomeno di instabilità locale che porta ad una configurazione "ribaltata" della struttura. Il passaggio tra la geometria iniziale e quella ribaltata avviene in maniera repentina con un rilascio dell'energia cinetica immagazzinata che si trasforma in un'azione impulsiva che coinvolge gli elementi adiacenti al nodo dove lo snap through si è manifestato.

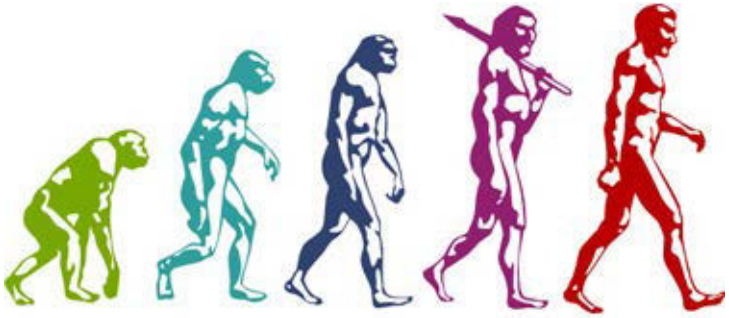


Se l'energia rilasciata è elevata, si accentua l'effetto dinamico che coinvolge i nodi adiacenti, trasformando l'instabilità locale in una più pericolosa instabilità globale che può portare al collasso dell'intera struttura o di una larga parte di essa [Gioncu V., Lenza P., 1993].

- IL FORM FINDING CON PRESTRESS UNIFORME GENERA UNA STRUTTURA OTTIMALE PER IL BUCKLING?

- ESISTE UN MODO PER INIZIALIZZARE IL PRESTRESS IN MODO DA OTTENERE UNA STRUTTURA OTTIMA DAL PUTNO DI VISTA DELL' INSTABILITÀ?

- È POSSIBILE SVILUPPARE UNA PROCEDURA CHE CONSENTA DI CONIUGARE FORM-FINDING E BUCKLING PER LA PROGETTAZIONE DI GRIDSHELLS?



Non è la specie più forte a sopravvivere, né la più intelligente, ma quella più pronta al cambiamento.

[Charles Darwin]

Un algoritmo genetico è un algoritmo di ottimizzazione basato su un metodo euristico ispirato al principio della selezione naturale. I principi fondamentali su cui si basa la selezione naturale sono:

- il principio della variabilità dei caratteri tra gli individui di una popolazione;
- il principio dell'adattamento, secondo il quale gli individui che presentano caratteri vantaggiosi ai fini della sopravvivenza sono i più adatti (*fit*) all'ambiente;
- il principio dell' ereditarietà dei caratteri tra genitori e figli.

L'idea che sta alla base di un algoritmo genetico è quella di, partendo da un insieme iniziale di possibili soluzioni (individui) per un determinato problema, selezionare le migliori e ricombinarle fra di loro in modo tale che esse evolvano verso un punto di ottimo (massimo o minimo).

Per formulare un problema in termini genetici si deve:

- determinazione dello schema di rappresentazione;
- determinazione della funzione di fitness
- determinazione dei parametri e delle variabili per controllare l'algoritmo;
- determinazione del criterio di arresto dell'algoritmo.

TERMINOLOGIA

DEFINIZIONE	BIOLOGIA
codice genetico	CROMOSOMA
componente di un cromosoma	GENE
possibili varianti di un gene	ALLELE
posizione di un gene nel cromosoma	LOCUS
informazione genetica completa	GENOMA
insieme dei geni contenuti in un genoma	GENOTIPO
insieme di tutte le caratteristiche osservabili in un individuo	FENOTIPO
ricombinazione dei geni durante la riproduzione	CROSSOVER
alterazione casuale di un cromosoma	MUTAZIONE
grado di adattabilità di un individuo	FITNESS

Schema di rappresentazione

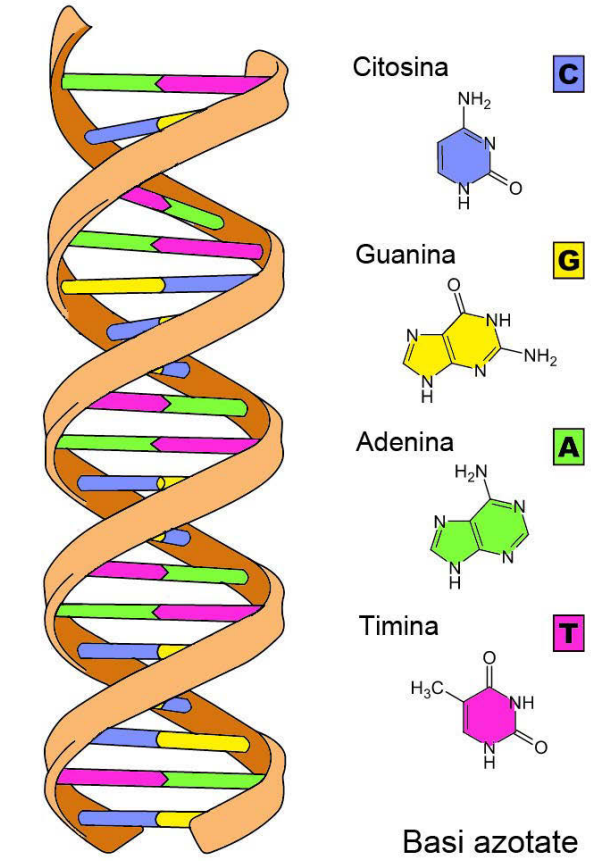
Lo schema di rappresentazione non è altro il processo che porta alla definizione delle variabili del problema, inclusa la definizione di un possibile spazio di ricerca della soluzione.

CODIFICA BINARIA

CROMOSOMA [A]	0	1	1	0	1	0	0	1	0	1	0	0
CROMOSOMA [B]	1	1	0	0	0	1	1	1	0	0	1	1

CODIFICA VALORI REALI

CROMOSOMA [A]	1.235	5.324	0.455	2.329	2.454	6.764	3.103	3.216
CROMOSOMA [B]	A B D J E I F J D H D I E R J F D L K F G T N L T Y							
CROMOSOMA [C]	(back), (back), (right), (forward), (left)							



DNA
Acido Desossiribonucleico

Definizione Fitness Function

La fitness function è la funzione “obiettivo” che l’algoritmo genetico deve massimizzare o minimizzare. Ad ogni individuo avrà una propria funzione di fitness attraverso la quale sarà valutata la sua “bontà”.

Si supponga, per esempio, che la funzione di fitness F dipenda dal vettore p composto da n variabili:

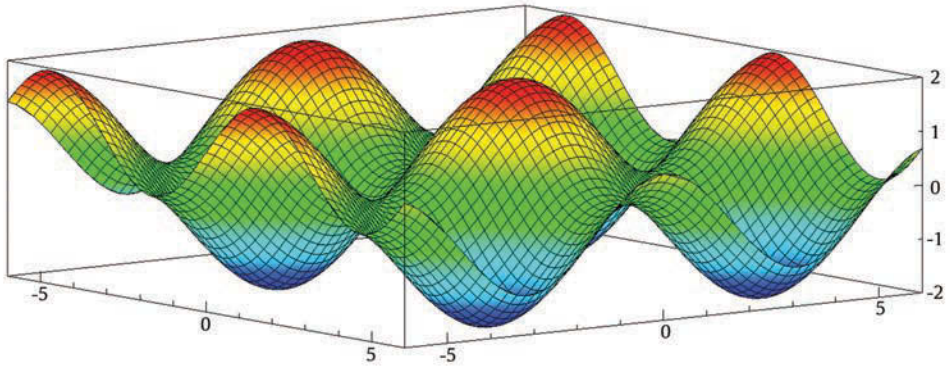
$$F = f(\mathbf{p}),\ \text{con}\ \mathbf{p} = (p_1,...,p_n),\ \text{per}\ p_n^{\min} \le p_n \le p_n^{\max}$$

L’ottimizzazione farà evolvere le soluzioni di **p** massimizzando (o minimizzando) il valore della fitness function *F*:

$$\max[\min]F = f(\mathbf{p}),\ \text{in}\ p_n^{\min} \le p_n \le p_n^{\max}$$

FITNESS LANDSCAPE

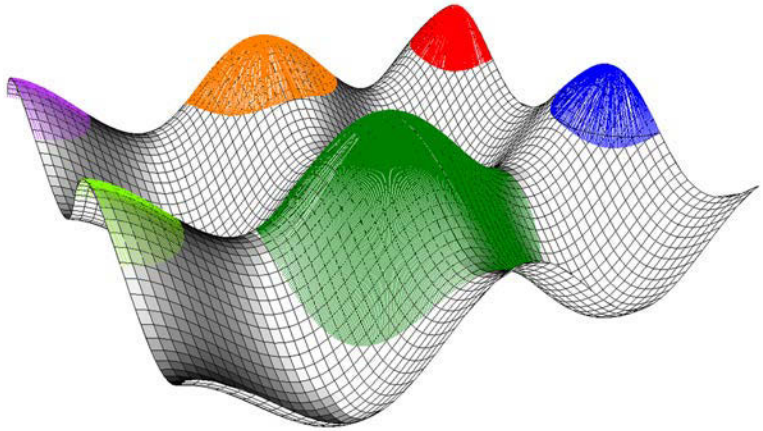
Il fitness landscape è la rappresentazione grafica della funzione di fitness.



fitness landscape della funzione f(x,y) = sin(x) - cos(y)

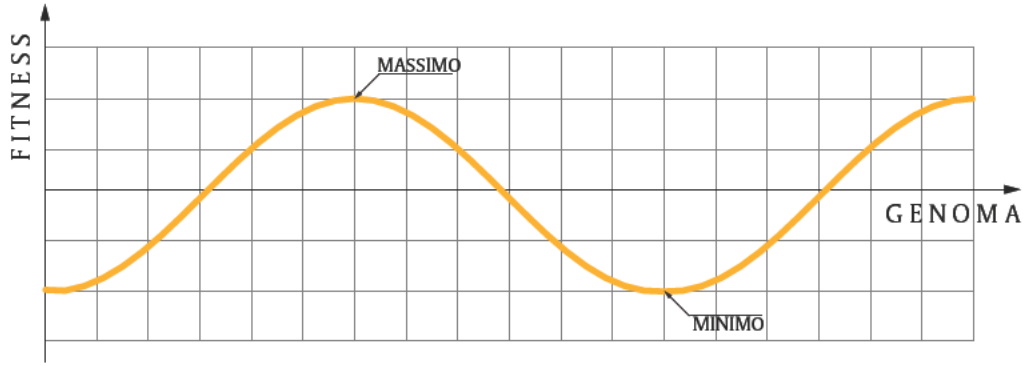
BACINO DI ATTRAZIONE

Per un sistema dinamico, un bacino di attrazione è il più grande intorno U di un attrattore x_0, tale che ogni orbita x(t), che parte da un punto interno a U, evolve verso x_0. Nel caso specifico, gli attrattori sono rappresentati dai picchi del fitness landscape.

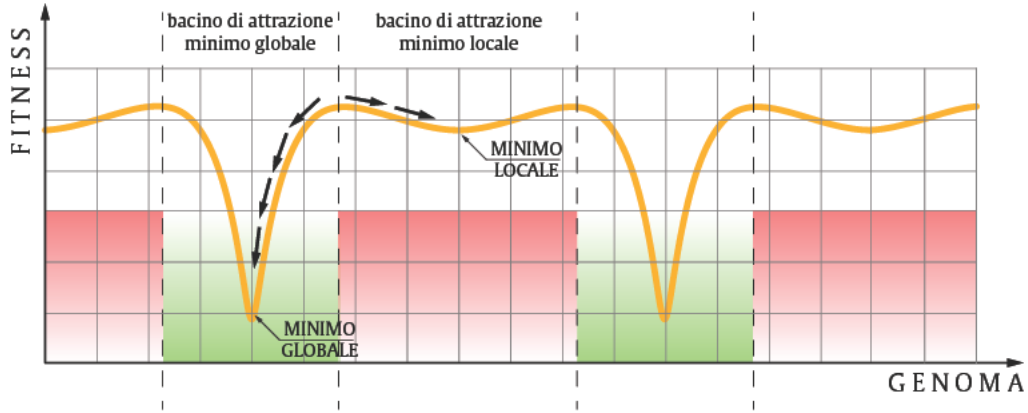


FITNESS SCALING

E' necessario scalare la funzione di fitness per creare delle "biodiversità" che consentano un'esplorazione più grande possibile dello spazio di ricerca.



In questo caso le soluzioni raggiungono un ottimo globale senza il pericolo di incappare in una soluzione locale.



Il ladscape presenta una soluzione locale con un bacino di attrazione doppio rispetto all'ottimo. È necessario quindi introdurre una diversità nelle soluzioni che eviti una convergenza prematura verso un ottimo locale.

ESEMPIO NUMERICO

FITNESS	ADATTABILITÀ
1100	17%
1200	18.5%
1300	20%
1400	21.5%
1500	23.%

La probabilità che gli individui vengano selezionati per la riproduzione è, in pratica, la stessa. C'è il pericolo di avere una convergenza prematura dell'algoritmo.

METODI DI FITNESS SCALING

- Linear scaling
- Sigma truncation scaling
- Power law scaling

Parametri di controllo

POPOLATION SIZE	20 - 30
NUMBER OF GENERATIONS	non specificato
CROSSOVER RATE	0.8 - 0.95 (80%- 90%)
MUTATION RATE	0.005 - 0.01 (0.5% - 1%)

Criteri di arresto

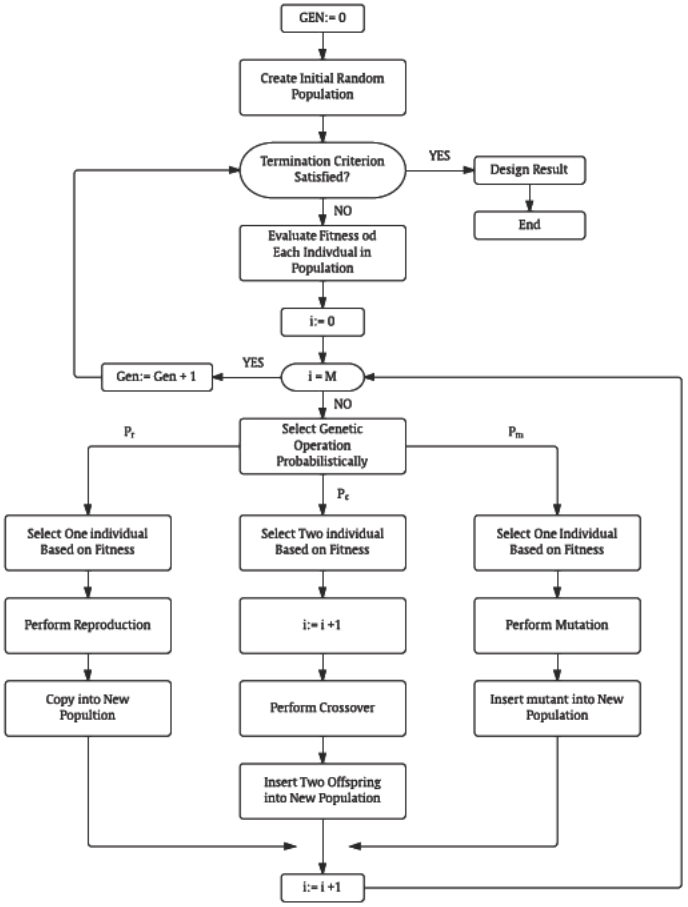
TEMPO DI EVOLUZIONE	L'algoritmo si arresta quando il tempo di evoluzione, imposto dall'utilizzatore, viene raggiunto.
NUMERO DI GENERAZIONI	L'algoritmo si arresta quando il numero massimo di generazioni viene raggiunto.
CONVERGENZA DELLA POPOLAZIONE	L'algoritmo si arresta quando una popolazione converge. La convergenza di una popolazione può essere stimata in vari modi come ad esempio verificando l'uguaglianza, in termini di massima minima e media fitness, fra la popolazione corrente e la popolazione allo step successivo oppure controllando che la fitness media della popolazione corrente sia inferiore ad una percentuale della fitness massima definita dall'utente.
CONVERGENZA DELLA FITNESS	L'algoritmo si arresta, ad esempio, quando i valori massimi minimi e medi della fitness function per la popolazione corrente si eguagliano.
CONVERGENZA DEL VALORE MIGLIORE DELLA FITNESS	L'algoritmo si arresta, ad esempio, quando il valore "grezzo" (non scalato) della funzione di fitness è uguale ad un valore definito dall'utente.

Procedura di un GA

Dopo aver formulato il problema in esame in termini genetici, la procedura convenzionale per un algoritmo genetico può essere riassunta nei seguenti step principali:

- generare una popolazione iniziale in maniera random;
- ripetere i seguenti passi per tutti gli individui della popolazione fino a che non si raggiunge il risultato cercato:
  1. valutare la fitness function per ogni individuo della popolazione;
  2. generare una nuova popolazioni di individui applicando i seguenti tre operatori genetici:
     i. selezionare gli individui migliori per la riproduzione della nuova popolazione;
     ii. ricombinare il codice genetico degli individui selezionati in modo da creare un nuovo individuo;
     iii. applicare una mutazione random al codice genetico dei nuovi individui
- l'individuo migliore di ogni generazione (ovvero di ogni iterazione) rappresenta la soluzione ottimale o una sotto-soluzione ottimale al problema.

```
1   PROCEDURE Ga_procedure
2       BEGIN
3           READ the GA's parameters
4           SET t to 0
5           INIT the population P(t) with random individual
6           FOR each <individual> in the P(t)
7               Evaluate individual performance
8           ENDFOR
9           WHILE terminating condotion not met ITERATE
10              Sort the individual in P(t) in basis of their fitness function
11              Select the individuals with a selection criteria
12              Crossover parentnts to form offspring and copy them in <incubator>
13              Mutate offspring
14              FOR each <offspring> in <incubator>
15                  Evaluate offspring
16              ENDFOR
17              SET P(t+1) by replacing the wrost individuals with the offspring
18              INCREMENT t
19          ENDWHILE
20          DISPLAY the best individual
21      END
```



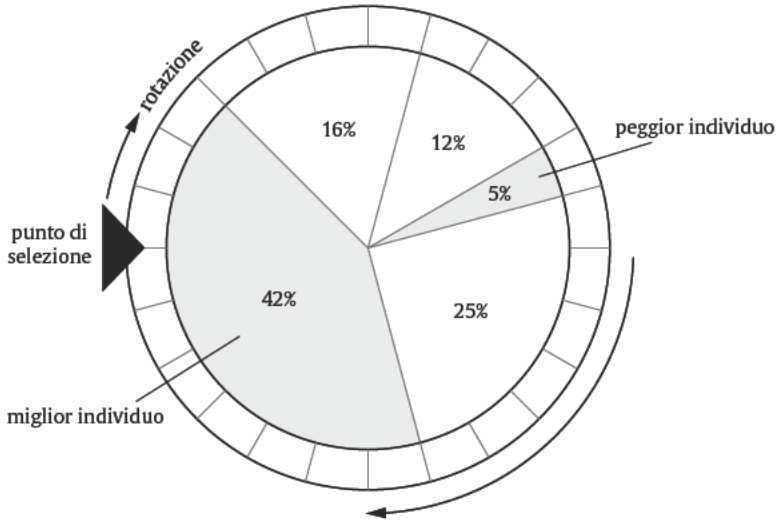
Operatori genetici

SELEZIONE

PROPORTIONATE ROULETTE WHEEL SELECTION

Questa selezione assegna ad ogni individuo una possibilità di sopravvivere proporzionale alla propria fitness. Le singole probabilità corrispondono ad uno spicchio di roulette come mostrato in Fig.9. La roulette viene quindi fatta ruotare per in maniera casuale per un numero finito di volte che, generalmente, è uguale al numero degli individui in una generazione o, al limite, pari a due ovvero uguale al numero minimo necessario per generare un nuovo individuo. Chiaramente l'individuo migliore ha una maggiore possibilità di essere scelto (lo spicchio della roulette è più grande) ma in questo modo si dà una chance anche a tutti gli altri aumentando così la varietà genetica e quindi l'esplorazione dello spazio di ricerca.

```
1   PROCEDURE Roulette_Wheel_Selection
2       BEGIN
3           SET <fitness_sum> equal to 0
4           FOR each <individual> in the <population>
5               SUM the <fitness_sum> to the fitness of this individual
6           ENDFOR
7
8           SET i equal to 0
9           FOR each <individual> in the <population>
10              SET <probability> equal to the <individual> fitness devide by <fitness_sum>
11              SET <list_propability[i]> equal to probability
12              SET i equal to i plus 1
13          ENDFOR
14
15          FOR each <individual> in the <population>
16              SET <rand> equal to a random number in the range [0;1]
17              SET <p> equal to 0
18              WHILE <list_probability> is minus then <rand> DO
19                  p = p + 1
20              ENDWHILE
21              COPY the individual <population[p]> in the current position of the population
22          ENDFOR
23      END
```



Operatori genetici

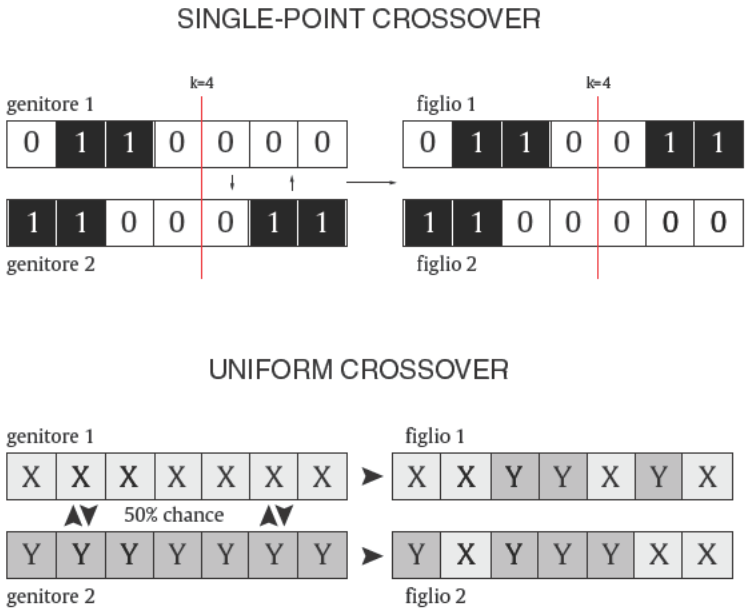
RIPRODUZIONE

UNIFORM CROSSOVER

Questo tipo di crossover è idoneo per un cromosoma codificato con valori reali. Infatti per un codice binario il locus del gene, cioè la sua posizione, è molto importante. Si prenda, ad esempio, un cromosoma binario 111000. Questo è estremamente diverso rispetto a 000111. In un cromosoma con valori reali, invece, la poszione è in generale irrilevante e quindi un cromosoma 3-4-11-35-6 è lo stesso di un cromosoma 35-11-4-6-3. Per lo uniform crossover, ogni allele di entrambi i genitori, è scambiato con l'allele corrispondente dell'altro genitore, con un fattore di probabilità pari, generalmente, a 0.5. In questo modo ogni allele di ciascun genitore ha il 50% di possibilità di essere rimpiazzato dal corrispondente dell'altro. Il processo viene ripetuto due volte per la generazione di ogni figlio.

```
1   PROCEDURE One_Point_Crossover
2       BEGIN
3           SET <probability> of swapping, often set to 0.5
4           FOR each <reprodiction_operation> in the <polpulation>
5               SELECT randomly <parent_1> in the <population>
6               SELECT randomly <parent_2> in the <population>
7               SET <u> as a random number in range [0,1]
8               FOR each <gene> in the <chromosome> of the <parent_1>
9               IF <u> is smaller than <probability>
10                  COPY <parent_1_gene> in the same position into <child_1_gene>
11                  COPY <parent_2_gene> in the same position into <child_2_gene>
12              ELSE
13                  COPY <parent_1_gene> in the same position into <child_2_gene>
14                  COPY <parent_2_gene> in the same position into <child_1_gene>
15              ENDIF
16          ENDFOR
17      END
```

SINGLE-POINT vs UNIFORM CROSSOVER

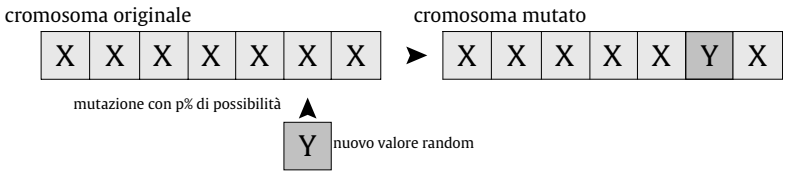


Operatori genetici

MUTAZIONE

L'operatore di mutazione provoca una variazione genetica nel cromosoma degli individui di una popolazione. Al contrario dell'operatore di crossover, che è considerato un costruttore di nuovi candidati alla soluzione finale, la mutazione è considerata com un distruttore della configurazione esistente. Per questa regione, gioca un ruolo secondario nel processo genetico. Infatti le mutazioni sono inserite allo scopo di evitare, ad esempio, che tutti gli individui siano uguali e quindi che che la possibilità di evoluzione si arresti in un ottimo locale.

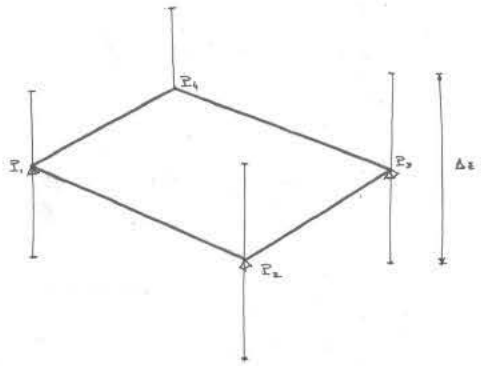
```
1   PROCEDURE One_Point_Crossover
2       BEGIN
3           SET <probability> of swapping, often set to 0.5
4           FOR each <reprodiction_operation> in the <polpulation>
5               SELECT randomly <parent_1> in the <population>
6               SELECT randomly <parent_2> in the <population>
7               SET <u> as a random number in range [0,1]
8               FOR each <gene> in the <chromosome> of the <parent_1>
9               IF <u> is smaller than <probability>
10                  COPY <parent_1_gene> in the same position into <child_1_gene>
11                  COPY <parent_2_gene> in the same position into <child_2_gene>
12              ELSE
13                  COPY <parent_1_gene> in the same position into <child_2_gene>
14                  COPY <parent_2_gene> in the same position into <child_1_gene>
15              ENDIF
16          ENDFOR
17      END
```

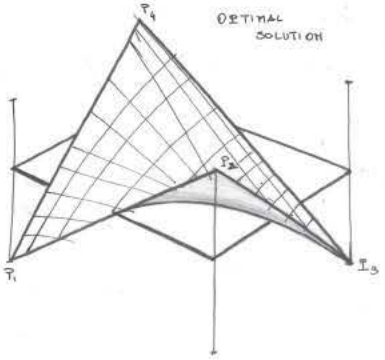


Semplice Algoritmo Genetico

DEFINIZIONE DEL PROBLEMA

Ricercare, partendo da una geometria quadrata piana, muovendo i vertici lungo l'asse z entro un intervallo stabilito, la geometria che ha il perimetro maggiore. L'intervallo di variabilità della posizione verticale dei quattro vertici è pari alla lunghezza del lato della superficie piatta. La lunghezza di un lato della geometria piana è pari a 8 m. La soluzione è nota e si tratta di un iperboloide parabolico.





date le coordinate dei quattro punti della geometria piatta $P_i = (x_i, y_i, z_i)$ per $i = (1, 2, 3, 4)$, il perimetro della geometria è pari alla somma della distanze fra due punti adiacenti ovvero:

$$p = (d_{1,2} + d_{2,3} + d_{3,4} + d_{4,1})$$

essendo $d_{i,j}$ la distanza fra due punti vicini pari a:

$$d_{i,j} = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2 + (z_j - z_i)^2} \text{ per } i = (1,2,3,4) \text{ e } j = (2,3,4,1)$$

Il perimetro è quindi funzione della coordinata z dei punti:

$$p = f(z_i) \text{ per } i = (1,2,3,4)$$

Se si indica con **z** il vettore delle coordinate z dei quattro punti, il problema può essere scritto come:

$$\max p = f(\mathbf{z}) \text{ con } \mathbf{z} = (z_1, z_2, z_3, z_4)$$

con z_i che può variare fra:

$$-\frac{l}{2} \le z_i \le \frac{l}{2} \text{ con } i = (1,2,3,4) \quad \text{con } l = 8 \text{ m.}$$

```python
def ga_engine():
    genome = G1DList.G1DList(4)
    genome.initializator.set(Initializators.G1DListInitializatorReal)
    genome.setParams(rangemin = range_min, rangemax = range_max)
    genome.mutator.set(Mutators.G1DListMutatorRealGaussian)
    genome.crossover.set(Crossovers.G1DListCrossoverUniform)
    genome.evaluator.set(eval_function)

    ga=GSimpleGA.GSimpleGA(genome)
    ga.setGenerations(generation)
    ga.setPopulationSize(population)
    ga.setElitism(elitism)
    ga.selector.set(Selectors.GRouletteWheel)
    #ga.terminationCriteria.set(GSimpleGA.ConvergenceCriteria)
    ga.setMultiProcessing(True)
    pop=ga.getPopulation()
    pop.scaleMethod.set(Scaling.SigmaTruncScaling)

    ga.evolve(1)
    best=ga.bestIndividual()
    print best
    i = 0
    best_points = []
    for point in arrPoints:
        moved_points =[point[0],point[1],point[2] + best[i]]
        best_points.append(moved_points)
        i = i + 1
    best_surface = rs.AddSrfPt(best_points)

if __name__=="__main__":
    ga_engine()
```

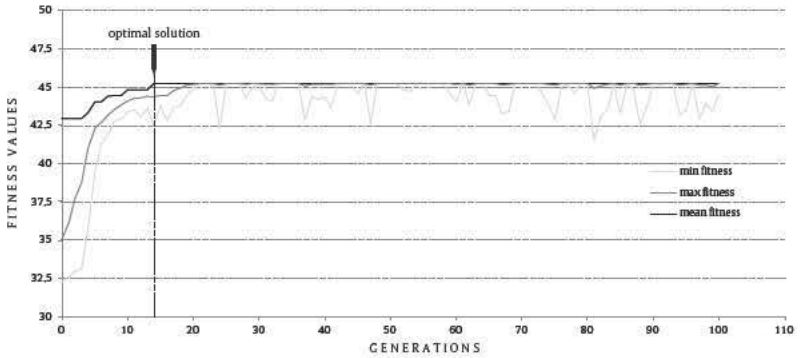

```python
def ga_engine():
    genome = G1DList.G1DList(4)
    genome.initializator.set(Initializators.G1DListInitializatorReal)
    genome.setParams(rangemin = range_min, rangemax = range_max)
    genome.mutator.set(Mutators.G1DListMutatorRealGaussian)
    genome.crossover.set(Crossovers.G1DListCrossoverUniform)
    genome.evaluator.set(eval_function)

    ga=GSimpleGA.GSimpleGA(genome)
    ga.setGenerations(generation)
    ga.setPopulationSize(population)
    ga.setElitism(elitism)
    ga.selector.set(Selectors.GRouletteWheel)
    #ga.terminationCriteria.set(GSimpleGA.ConvergenceCriteria)
    ga.setMultiProcessing(True)
    pop=ga.getPopulation()
    pop.scaleMethod.set(Scaling.SigmaTruncScaling)

    ga.evolve(1)
    best=ga.bestIndividual()
    print best
    i = 0
    best_points = []
    for point in arrPoints:
        moved_points =[point[0],point[1],point[2] + best[i]]
        best_points.append(moved_points)
        i = i + 1
    best_surface = rs.AddSrfPt(best_points)

if __name__=="__main__":
    ga_engine()
```

Semplice Algoritmo Genetico

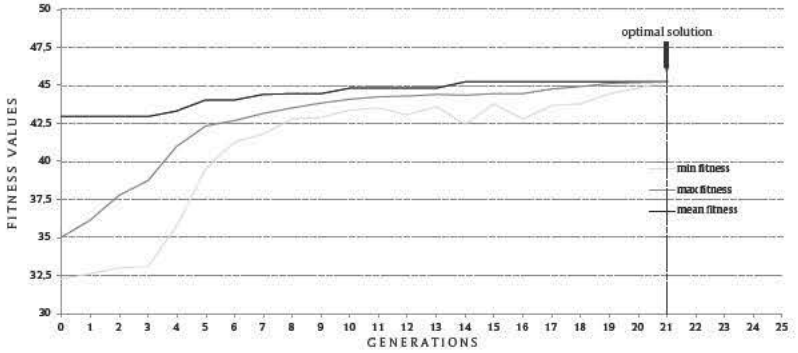
GA IN PYEVOLVE

CODIFICA	Numeri Reali
GENERAZIONI	100
POPLAZIONE	21
TIPO DI MUTAZIONE	Gaussian Real Mutation [31
PERCENTUALE DI MUTAZIONE	0.03 (3%)
TIPO DI CROSSOVER	Uniform Crossover
PERCENTUALE DI CROSSOVER	0.50 (50%)
N. DI INDIVIDUI ELTIE	1
TIPO DI SCALA DELLA FITNESS	Sigma Trunc Scaling
TIPO DI CONVERGENZA	Numero generazioni/Convergenza popo-lazione

CRITERIO DI ARRESTO: NUMERO DELLE GENERAZIONI

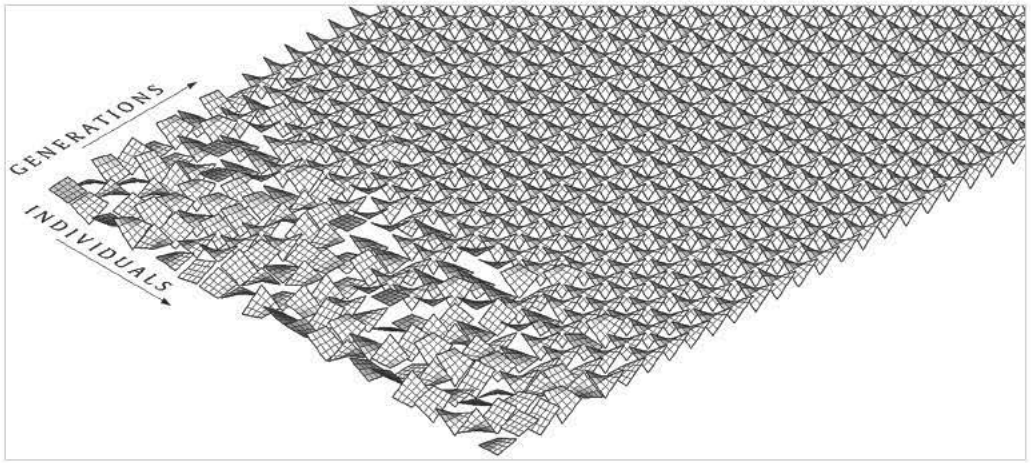


CRITERIO DI ARRESTO: CONVERGENZA POPOLAZIONE



Sono stati sviluppati due algoritmi diversi con la biblioteca PyEvolve:

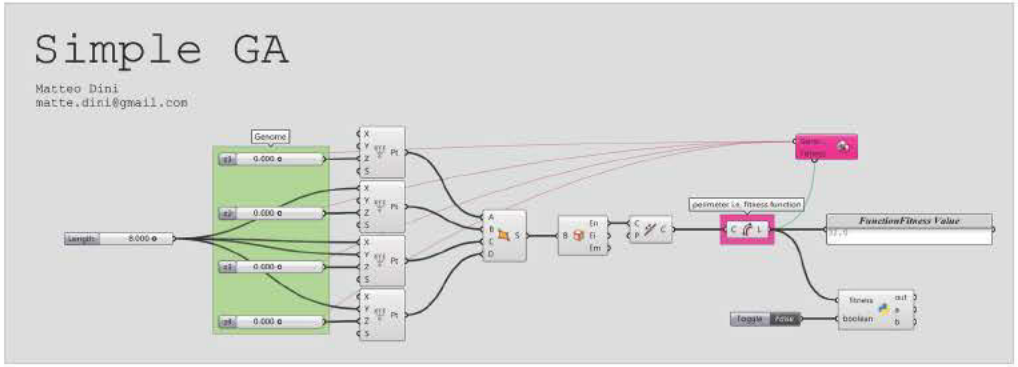
- GA_1: algoritmo semplice
- GA_2: algoritmo più raffinato che permette la visualizzazione di tutta l'evoluzione

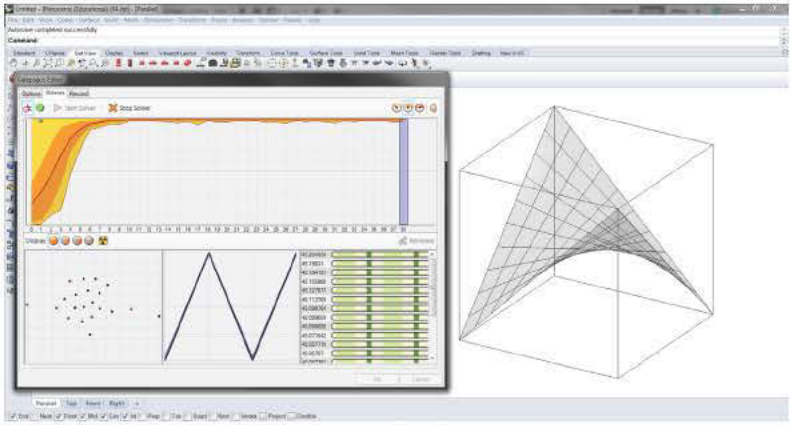


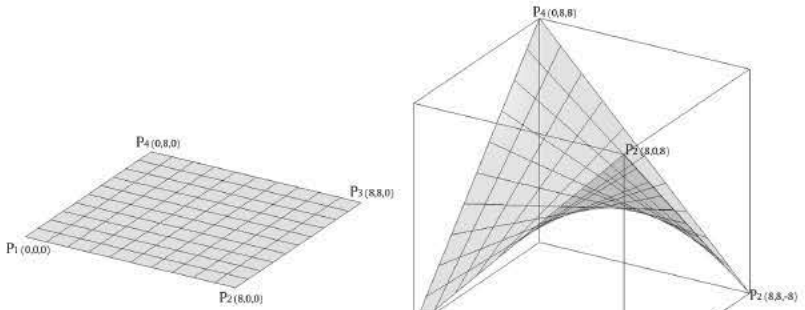
Semplice Algoritmo Genetico

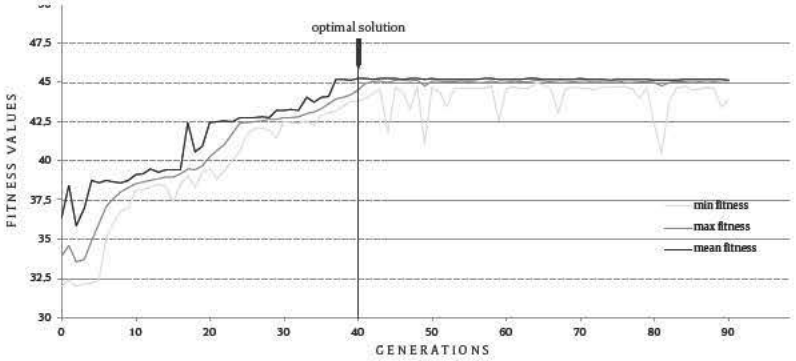
GA IN GRASSHOPPER/GALAPAGOS

FITNESS	Maximize
MAX. STAGNANT	50
POPULATION	30
INITIAL BOOST	2
MAINTAIN	2%
INBREEDING	75%









Semplice Algoritmo Genetico

CONFRONTO GALAPAGOS vs. PYEVOLVE

ALGORITMO GENETICO	TEMPO DI ESECUZIONE [s]	CRITERIO DI ARRESTO
Galapagos	5.540	stagnant iterations (50)
PyEvolve GA_1	0.670	numero generazioni (100)
PyEvolve GA_2	3.50	numero generazioni (100)
PyEvolve GA_1	0.230	convergenza popolazione
PyEvolve GA_2	1.51	convergenza popolazione

Form-Finding e Buckling

DEFINIZIONE DEL PROBLEMA

Il problema studiato consiste nel ricercare la forma, attraverso un processo di form-finding, che sia ottimizzata nei confronti del buckling.

Le diverse geometrie vengono generate cambiando il prestress negli elementi che compongono la struttura. I valori che può assumere il prestress varia da 0.5 kN a 10 kN. Per poter confrontare le geometrie, l'altezza massima è stata posta pari a 3 metri. Questo comporta la necessità di scalare il prestress fino a che la struttura non raggiunge l'altezza ricercata. Una volta trovata la forma le proprietà degli elementi vengono cambiate impostando le sezioni reali e come materiale l'acciaio e non più il materiale utilizzato per il form finding con il modulo di elasticità ridotto. La tipologia dei vincoli interni ed esterni viene scelta in base all' analisi che si vuole effettuare. Si procede quindi all' analisi di buckling con carichi nodali pari a quelli che hanno generato la struttura (1kN) che fornisce il valore del più basso carico critico di collasso. L'algoritmo genetico deve massimizzare questo valore.

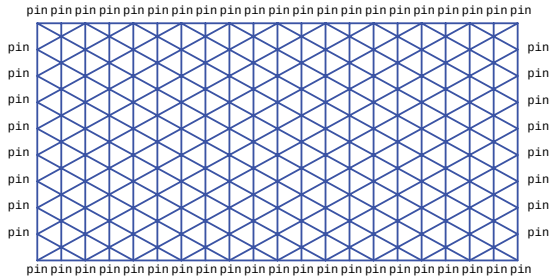
Se indichiamo con p = (p1,p2,...,pn) il vettore del prestress virtuale per n elementi strutturali, l'ottimizzazione fa evolvere le soluzioni per p massimizzando:

$$\max b(\mathbf{p}) \text{ con } \mathbf{p} = (p_1, p_2, ..., p_n)$$

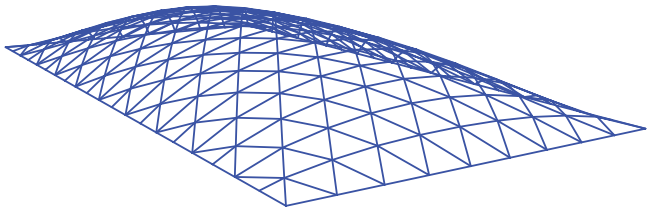
In termini genetici si può dire che:

$\max b(\mathbf{p}) \text{ con } \mathbf{p} = (p_1, p_2, ..., p_n)$ è la FITNESS FUNCTION

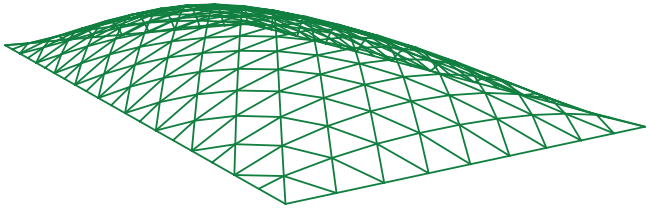
il vettore p = (p1,p2,...,pn) è il GENOMA



form-finding



struttura per il buckling



LOAD FACTOR

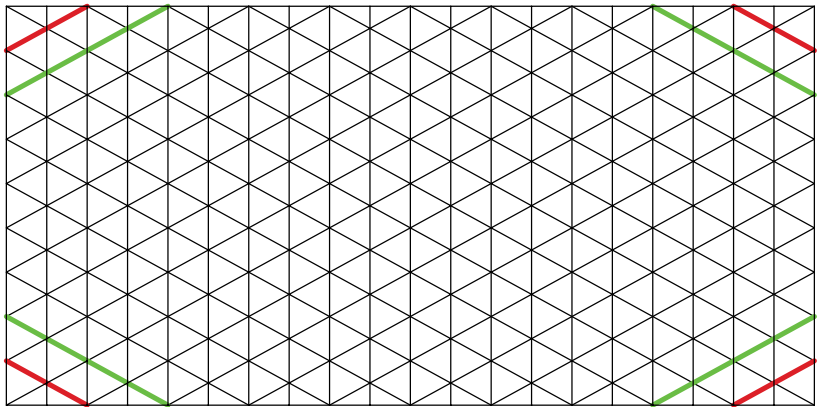
PARAMETRI DELL'ALGORITMO GENETICO

CODIFICA	Numeri Reali
GENERAZIONI	50
POPLAZIONE	10
TIPO DI MUTAZIONE	Gaussian Real Mutation
PERCENTUALE DI MUTAZIONE	0.05 (5%)
TIPO DI CROSSOVER	Uniform Crossover
PERCENTUALE DI CROSSOVER	0.50 (50%)
N. DI INDIVIDUI ELTIE	1
TIPO DI SCALA DELLA FITNESS	Sigma Trunc Scaling
TIPO DI CONVERGENZA	Numero generazioni/Convergenza popolazione

MODELLAZIONE DELLA STRUTTURA IN GSA

La modellazione della maglia strutturale piana è stata fatta in GSA in modo tale da facilitare l'esecuzione delle procedure dell'algoritmo. Le caratteristiche principali del modello sono:

- CARICHI NODALI PER IL FORM-FINDING: 1 kN
- DEFINIZIONE DELLE SEZIONI PER IL FORM FINDING E DI UN MATERIALE CON E = 0.05 N/mm^2
- DIVISIONE DEGLI ELEMENTI IN GRUPPI SIMMETRICI PER L'ASSEGNAZIONE DEL PRESTRESS

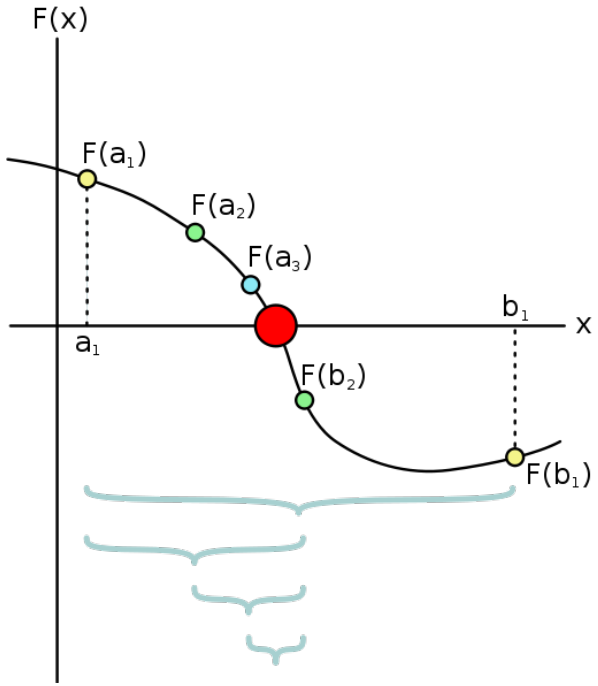


- DEFINIZIONE DELLE SEZIONI PER GLI ELEMENTI PER L' ANALISI DI BUCKLING
- CARICHI NODALI PER IL BUCKLING: 1 kN

ALGORITMO DI BISEZIONE

L'algoritmo consente di trovare la radice di una funzione compresa in un intervallo [a,b] con a e b di segno opposto.

```
INPUT : Function f, endpoint values a, b, tolerance TOL, maximum iterations NMAX
CONDITIONS : a < b, either f(a) < 0 and f(b) > 0 or f(a) > 0 and f(b) < 0
OUTPUT : value which differs from a root of f(x)=0 by less than TOL

N = 1
WHILE N = NMAX
  c = (a + b)/2
  IF (f(c) = 0 OR (b – a)/2 < TOL THEN { solution found
    OUTPUT (c)

  IF sign(f(c)) = sign(f(a)) then a = c else b = c
  N = N + 1 increment step counter

OUTPUT ("Method failed.") max number of steps exceeded
```



ALGORITMO DI BISEZIONE: RICERCA DELL' ALTEZZA

Per la ricerca dell'altezza si utilizza l'algoritmo di bisezione in modo da scalare il prestress degli elementi fino a che non ri raggiunge l'altezza cercata di 3 metri.

1. Definizione dell'intervallo *[a,b]*
2. Form-finding con un prestress pari a:

   p * a e **p** * b
3. Calcolo dell'altezza massima per le due geometrie trovate:

   h(a) e *h(b)*
4. Calcolo di *f(a)* e *f(b)* come:

   $f(a) = h(a) - h$

   $f(b) = h(b) - h$
5. Se $f(a)*f(b)>0$ si modifica l'intervallo e si riparte dal punto 2
6. Calcolo di *c*:

   $c = 0.5 * (a + b)$
7. Calcolo di *h(c)* con un prestress **p** * c
8. Calcolo di *f(c)*:

   $f(c) = h(c) - h$
9. Se $f(c) < Tol$ allora c è il valore del prestress che mi consente di avere un'altezza di 3 metri altrimenti:

   se $f(c) * f(b) > 0$ pongo $a = c$ e riparto dal punto 7

   se $f(c) * f(a) > 0$ pongo $b = c$ e riparto dal punto 7

```python
def bisection(a, b, arrPrestress, target_heigth, path1, path2, tol = 5.0e-3):

    arrPs_a = []
    arrPs_b = []

    for ps in arrPrestress:
        arrPs_a.append(a * ps)
    for ps in arrPrestress:
        arrPs_b.append(b * ps)



    heigth_a = heigth(path1,path2,arrPs_a)
    fa       = heigth_a - target_heigth

    heigth_b = heigth(path1,path2,arrPs_b)
    fb       = heigth_b - target_heigth

    if fa == 0.0    : return path2
    if fb == 0.0    : return path2

    if fa * fb > 0.0:
        arrPrestress = []
        b            = b + 5
        for ps in arrPrestress:
            arrPs_b.append(b * ps)
        heigth_b = heigth(path1,path2,arrPs_b)

    n = ceil(log(abs(a - b) / tol) / log(2))

    for i in range(int(n)):
        c       = 0.5 * (a + b)
        arrPs_c = []
        for ps in arrPrestress:
            arrPs_c.append(c * ps)

        heigth_c = heigth(path1,path2,arrPs_c)
        fc       = heigth_c - target_heigth
        if abs(fc) < tol: return path2
        if fb * fc < 0.0: a = c
        else:             b = c

    return path2
```

CALCOLO DEL FATTORE DI BUCKLING

Il fattore di buckling viene calcolato su una geometria deformata. L'algoritmo per il calcolo è il seguente:

1. Calcolo del primo modo di buckling *EV1* per la condizione di carico che ha generato la forma

2. Calcolo della nuova condizione di carico pari a:

$$0.8 * EV1$$

3. Analisi non lineare con la nuova condizione di carico e definizone della nuova geometria deformata

4. Calcolo del primo modo di buckling *EV2* per la geometria deformata

5. Calcolo del fattore di buckling per la struttura pari a:

$$(0.8 * EV1) * (EV2)$$

Definizione delle variabili di progetto

Definizione della prima generazione random
$(p_1, p_2, \ldots, p_n)$

$i = 0$

$i = i + 1$

INPUT

Ricerca dell'altezza con l'algoritmo di bisezione

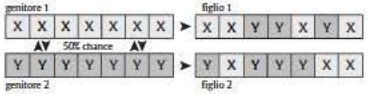
Cambio delle proprietà degli elementi della struttra

Calcolo del fattore di buckling

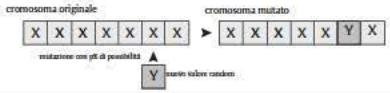
OUTPUT
Fitness Function

Fitness Proportionate Selection

Crossover



Mutazione



sintassi per connettere GSA con Python/Pyevolve

OTTIMIZZAZIONE///28

FLAT GEOMETRY	PRESTRESS DIRECTIONS: ONE DIRECTION	PRESTRESS DIRECTIONS: TWO DIRECTIONS	PRESTRESS DIRECTIONS: THREE DIRECTIONS	FORM FINDING SHAPE INTIAL: UNIFROM PRESTRESS	RESTRAINTS: PIN	RESTRAINTS: ENCASTRE	RELEASES: ENCASTRES	RELEASES: PINS DIRECTIONS1	RELEASES: PINS DIRECTIONS2	SECTIONS: EQUAL ALL WAYS	SECTIONS: DIRECTION 1: A#1=A/10 DIRECTION 2: A#2=A/10	SECTIONS: DIRECTION 1: A#1=A/10 DIRECTION 2: A#2=A	GEOMETRY INITIALIZATION: INITIAL SHAPE	GEOMETRY INITIALIZATION: LOAD FACTOR	BEST SHAPE: BEST SHAPE	BEST SHAPE: LOAD FACTOR	Δ	%
			■		■		■			■				19.04		22.32	3.28	17.3%
			■			■	■			■				20.99		22.81	1.82	8.61%
	■				■				■		■			2.08		7.52	5.44	261%
		■			■				■		■			2.22		5.74	3.54	159%
			■		■				■		■			5.33		6.67	1.34	25.1%
			■		■				■	■				14.92		16.48	1.56	10.5%
	■					■			■		■			2.67		9.27	6.60	247%
		■				■			■		■			2.36		6.85	4.49	190%
			■			■			■		■			5.41		6.90	1.49	27.5%
			■			■			■	■								
		■			■			■				■		6.29		8.99	2.70	43.6%
			■		■			■				■		7.49		10.43	2.94	39.2%
		■				■		■				■		7.58		10.16	2.58	34%
			■			■		■				■		8.49		12.52	4.03	47.5%

This part of matrix indicates how I put the prestress for the form-finding

This part of matrix indicates the internal and external conditions for the buckling analysis for the form found shape

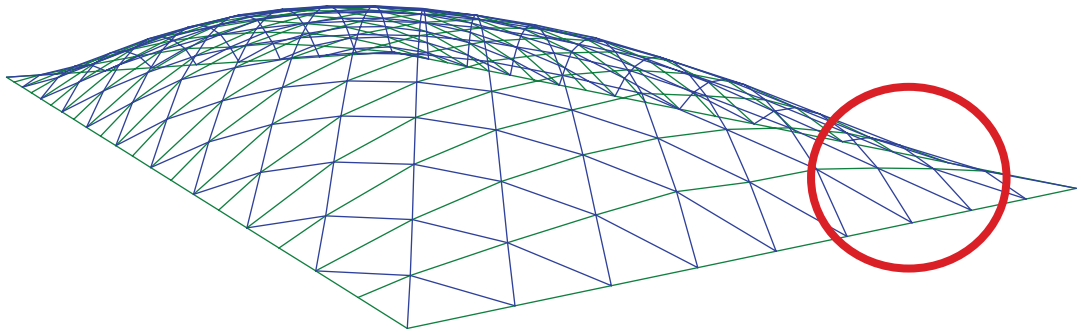
Risultati

- Il form finding effettuato con un prestress uniforme non genera la migliore geometria possibile nei confroti del bucklig

- Non esiste una relazione analitica tra i prestress per i vari elementi

- L'algoritmo genetico produce notevoli miglioramenti per il fattore di buckling delle geometrie

- Dai risultati si nota come gli individui migliori presentino un prestress maggiore per gli elementi in angolo. Questo produce una curvatura inversa dell'arco.



Futuri sviluppi

- Migliorare l'efficienza dell'algoritmo genetico in termini di tempo-macchina.
- Evitare la curvatura inversa degli archi agli angoli inserendo all'interno del GA il calcolo della curvatura della geometria come ulteriore parametro di controllo.
- Sviluppare un MOGA in grado di poter valutare separatamente funzioni di fitness diverse
- Effettuare ulteriori test con metodi di form-finding diversi e con una distribuzione di prestress negli elementi diversa
- Effettuare test con maglie strutturali diverse e su diversi perimetri
- Ricercare, se possibile, un metodo nuovo di form-finding che tenga in conto da subito del fattore di buckling della struttura