POLITECNICO DI TORINO


Tesi di Laurea Magistrale

In Ingegneria Civile

# IL DESIGN PARAMETRICO PER L'OTTIMIZZAZIONE DI UNA COPERTURA RETICOLARE

---

Modello numerico-sperimentale di ottimizzazione strutturale topologica, dimensionale e di forma

**Relatore:**

*Prof. Giuseppe Carlo Marano*

**Correlatore:**

*Laura Sardone*

**Candidato:**

*Alessandra Licari*

Anno Accademico 2020/2021

# Indice

**Sommario** ..... **3**

**1. Problemi di ottimizzazione** ..... **6**

1.1. Introduzione ..... 6

1.2. La ricerca dell'ottimo ..... 9

1.3. Algoritmi di ottimizzazione ..... 12

1.4. L'ottimizzazione strutturale ..... 16

1.4.1. Ottimizzazione dimensionale ..... 17

1.4.2. Ottimizzazione di forma ..... 20

1.4.3. Ottimizzazione topologica ..... 21

**2. Il design parametrico come nuovo approccio alla progettazione** ..... **25**

2.1. Introduzione ..... 25

2.2. Il design parametrico ..... 26

2.3. Le origini del design parametrico in architettura ..... 30

2.4. Vantaggi e svantaggi della progettazione parametrica ..... 39

2.4.1. I vantaggi del design parametrico ..... 40

2.4.2. Gli svantaggi del design parametrico ..... 43

2.4.3. Le distinzioni con metodi tradizionali CAD e BIM ..... 46

2.5. Il Computational design nei processi di progettazione ..... 51

2.5.1. La necessità di processi di design efficienti in termini di costi/tempi ..... 52

2.5.2. La disponibilità di nuove tecnologie ..... 53

2.5.3. L'impatto della progettazione computazionale sul ruolo degli ingegneri ..... 55

**3. Software e algoritmi parametrici** ..... **58**

3.1. Panoramica letteraria ..... 58

3.2. I principali software di design parametrico ..... 59

3.2.1. Grasshopper ..... 65

3.3. Gli algoritmi nei processi di design ..... 70

3.3.1. Il concetto di algoritmo ..... 71

3.3.2. La natura degli algoritmi e l'esplorazione della forma geometrica ..... 73

3.3.3. Introduzione agli algoritmi genetici ..... 78

**4. Ottimizzazione topologica, dimensionale e di forma: il caso studio** ..... **90**

4.1. Sviluppo di una copertura parametrica (*frame structure*) assistita da algoritmo ..... 91

4.2. Analisi FEM utilizzando Karamba3D ..... 96

4.3. Problema di ottimizzazione: il modello numerico ..... 103

4.3.1. SPEA-2 e HypE Reduction Algorithm: risoluzione del problema di ottimizzazione tramite l'uso di algoritmi genetici evolutivi ..... 105
4.3.2. Risultati..... 111
**5. Conclusioni .....122**
**6. Appendici.....124**
**7. Bibliografia .....129**
**8. Ringraziamenti .....132**

# Sommario

La crescita della complessità progettuale ed esecutiva nel settore delle costruzioni, oggi più che mai, rende la buona riuscita di un progetto fortemente dipendente dalle tecniche offerte dal campo dell'intelligenza artificiale. Grazie a tali tecniche, gli strumenti di calcolo sono in grado di interagire attivamente con il processo di progettazione indirizzando il progettista allo sviluppo di alternative che mostrano più rispondenza ai requisiti di progetto e alla ricerca di soluzioni sempre più orientate ad un comportamento ottimizzato della configurazione finale.
Per ottimizzazione si intende infatti un processo capace di individuare in modo automatico (totalmente o parzialmente) la soluzione progettuale "ottimale" che fornisce le prestazioni migliori, in relazione ad un determinato obiettivo da raggiungere e a vincoli di progettazione assegnati, massimizzando o minimizzando determinati parametri di performance. In ambito strutturale, essa trova un largo impiego nel miglioramento del comportamento, sotto l'effetto di carichi e sollecitazioni agenti, della struttura, e delle sue singole componenti, incrementando rigidezza e/o resistenza, riducendone peso e/o costo di produzione e, non ultimo per importanza, individuandone la disposizione ottimale del materiale da utilizzare senza però sacrificarne la resistenza richiesta alle sollecitazioni.

Nell'analisi di problemi complessi, come quelli della progettazione strutturale, nei quali interagiscono diverse variabili prestazionali, l'utilizzo di metodologie computazionali nel design può rappresentare un efficiente metodo di calcolo per la ricerca e l'analisi delle soluzioni di progetto "ottimali" capaci di garantire, da un lato, elevate prestazioni finali e, dall'altro, tempi e costi di progetto drasticamente ridotti.
Il *Computational Design* rappresenta infatti una delle frontiere più avanzate della progettazione architettonica, frutto di un approccio innovativo che evolve e modifica linguaggi e strumenti tradizionali e chiama i professionisti ad affrontare un complesso salto concettuale nell'approccio e nei processi che conducono allo sviluppo di un progetto. E non è difficile comprenderne il motivo, in quanto il tradizionale percorso concettuale e creativo tipico della disciplina viene affiancato e integrato da un metodo basato su calcoli e algoritmi finalizzati a offrire soluzioni a problematiche progettuali grazie a dati e parametri che, opportunamente elaborati (tramite algoritmi ad esempio), permettono di superare limiti e vincoli geometrici ottenendo soluzioni funzionali e originali.

L'elaborato di tesi si pone come obiettivo l'implementazione di tali algoritmi parametrici, come strumento di ottimizzazione nell'ambito dell'ingegneria strutturale, ai fini della progettazione "ottimale" di una copertura reticolare.

Il lavoro sarà strutturato in quattro capitoli, cercando di guidare il lettore, prima attraverso la comprensione del concetto di ottimizzazione strutturale e di design parametrico, supportato dall'enorme potenzialità di strumenti computazionali ed algoritmici, per poi discutere la parte applicativa che ha permesso di ottimizzare la struttura esaminata scegliendo come parametri di design, ovvero parametri da modificare opportunamente, l'altezza H della copertura e il numero dei frame contenuti rispettivamente lungo lo sviluppo della campata in direzione x ($n_x$) e in direzione y ($n_y$) oltre che la lunghezza delle campate $L_x$ e $L_y$.

Il capitolo 1 introduce alle basi teoriche e alle diverse tipologie di ottimizzazione discutendone le metodologie e i vantaggi offerti della sua applicazione in ambito strutturale.

Il capitolo 2 si occupa della descrizione del design parametrico secondo cui il punto di partenza del processo di design non sono le geometrie fisiche reali ma i parametri selezionati che, opportunamente sviluppati, generano soluzioni fisiche tanto più aderenti alle esigenze di partenza del progettista quanto più accurata è la loro definizione e selezione. Successivamente, viene posta l'attenzione sull'enorme potenzialità degli strumenti computazionali grazie ai quali, una volta definite una serie di regole (in forma di parametri) è possibile generare una forma in maniera quasi automatizzata e ottenere un numero virtualmente infinito di varianti dello stesso modello semplicemente agendo e modificando i parametri stessi. Infine, il capitolo discute i vantaggi e gli svantaggi offerti dalla modellazione parametrica facendo un confronto con i metodi tradizionali CAD e le metodologie BIM.

Il capitolo 3 discute invece gli strumenti applicativi del design parametrico, ovvero i software e gli algoritmi. Una volta introdotti i principali software parametrici oggi disponibili sul mercato, l'attenzione viene posta sul software *Grasshopper*, basato principalmente sul linguaggio di programmazione visuale, e su una particolare tipologia di algoritmi generativi definiti genetici (GA).

Il capitolo 4 infine, descrive la parte applicativa della tesi sullo sviluppo della copertura parametrica (sia in termini di modellazione geometrica che in termini di analisi strutturale) e sulla sua ottimizzazione discutendo i risultati ottenuti dall'implementazione dell'algoritmo in Grasshopper. In tal senso, il grande vantaggio offerto dalla modellazione parametrica e della

programmazione visuale ha consentito la convergenza alla configurazione finale della copertura reticolare come risultato di una doppia ottimizzare, in termini di spostamento massimo e massa (nonché costo), implementata tramite un algoritmo multi-obiettivo, e di una simultanea ottimizzazione delle sezioni trasversali degli elementi variando i parametri di design.

# 1. Problemi di ottimizzazione

La maggior parte dei problemi che riguardano l'ambito ingegneristico, economico, o in genere tutte le scienze esatte, possono essere rappresentati e studiati come problemi di ottimizzazione. L'ottimizzazione è quella disciplina che si occupa di determinare modelli utili nelle applicazioni, impiegando metodi efficienti per identificare una soluzione ottimale; questo spiega il grande interesse che oggi, sia da un punto di vista tecnico che scientifico, viene posto nei confronti dello studio e dello sviluppo di tali modelli capaci di garantire la complessa risoluzione di una vasta gamma di problemi.

## 1.1. Introduzione

Con la parola ottimizzazione, per definizione generale, si intende "il raggiungimento del risultato più vantaggioso possibile con i termini dati o in relazione a un determinato fine." Nel linguaggio matematico ottimizzare significa determinare il valore delle variabili di una funzione in modo che questa assuma il suo minimo o il suo massimo. In fase di progettazione infatti, è sempre consigliabile, o comunque utile, che il progettista tenda al perseguimento di una soluzione ottimale mediante l'utilizzo di specifici modelli, definiti modelli di ottimizzazione matematica, che consentono di trovare i valori di un set di parametri (variabili di progetto) in grado di massimizzare (o minimizzare) le funzioni di interesse, chiamate funzioni obiettivo. La formulazione di un modello di ottimizzazione può essere schematizzata attraverso quattro fasi principali:

(i) *Identificazione del problema*. Prima di tutto, è necessario identificare il problema che si vuole risolvere, comprendendo con chiarezza gli obiettivi del processo decisionale e i vincoli che la soluzione ottimale deve rispettare.

(ii) *Definizione delle variabili*. Occorre definire le variabili di decisione del modello, corrispondenti alle scelte che il processo decisionale comporta.

(iii) *Rappresentazione dell'obiettivo*. Bisogna definire la funzione obiettivo del modello, da minimizzare o massimizzare, dipendente dalle variabili di progetto che vengono fatte liberamente variare all'interno del processo di ottimizzazione stesso.

(iv) *Rappresentazione dei vincoli*. Per ultimo, è necessario identificare e rappresentare il set di vincoli del modello, se presenti, a cui la funzione obiettivo può essere

soggetta. Tali vincoli rappresentano infatti proprio quelle particolari condizioni che la soluzione del problema di ottimizzazione deve rispettare per poter essere ritenuta fattibile e devono essere espressi mediante equazioni e disequazioni nelle variabili di decisione.

Entrando nel dettaglio, data una funzione $f: IR^n \rightarrow IR$ un problema di Ottimizzazione, denominato anche problema di *Programmazione Matematica*, consiste nel determinare, se esiste, un punto di minimo (o massimo) della funzione $f$ tra tutti i punti dell'insieme *S*, insieme di tutte le $x$ possibili soluzioni del problema, e può essere sintetizzato, rispettivamente, nella forma:

$$\min_{x \in S} f(x)$$

$$\max_{x \in S} f(x)$$

La funzione $f(x)$ rappresenta la funzione obiettivo, $S$ è l'insieme delle soluzioni possibili (denominato ammissibile) e $x$ è il vettore delle variabili di progetto. Generalmente, il vettore $x$ è una variabile vettoriale n-dimensionale, esprimibile come $x = (x_1, x_2, \ldots, x_n)$, e quindi la funzione obiettivo $f(x)$ risulterà una funzione di n variabili reali $f(x_1, x_2, \ldots, x_n)$.

Gli elementi $x \in S$ si chiamano soluzioni ammissibili dell'insieme S e rappresentano una scelta alternativa ritenuta accettabile per lo specifico ambito di applicazione; un punto $x^* \in S$ che invece minimizza la funzione $f(x)$ soddisfacendo la relazione:

$$f(x^*) \leq f(x) \quad per\ ogni\ \ x \in S$$

viene detto *soluzione ottimale globale* o anche *soluzione di minimo globale (o assoluto)*. Il corrispondente valore della funzione obiettivo $f(x^*) = \min_{x \in S} f(x)$ viene detto valore di *ottimo globale* o di *minimo* di $f$ su $S$.

Vi sono tuttavia situazioni in cui le decisioni vengono confrontate mediante una funzione obiettivo che rappresenta un guadagno o comunque una caratteristica che deve essere massimizzata anziché minimizzata; in tal caso un problema di massimo può essere sempre ricondotto a un problema di minimo cambiando semplicemente il segno della funzione. I punti di massimo (ove esistono) del problema $\max f(x)\ x \in S$ infatti coincidono con i punti di minimo del problema $\min -f(x)\ \ x \in S$ e risulta quindi sempre valida l'espressione:

$$\max_{x \in S} f(x) = -\min_{x \in S}(-f(x))$$

Così facendo, la ricerca della soluzione "ottima" può essere sempre ricondotta alla ricerca del minimo della funzione obiettivo $f(x)$.

Come già accennato, la funzione obiettivo $f(x)$ da minimizzare, o massimizzare, è spesso soggetta a un set di vincoli. A differenza del caso appena visto, definito problema di *Ottimizzazione non Vincolata*, (in cui l'insieme ammissibile $S = IR^n$), una classe comune di problemi è quella in cui $S$ è un insieme chiuso in $IR^n$ e si parla in tal caso di *Ottimizzazione Vincolata*. L'insieme $S$ viene quindi descritto attraverso un insieme finito di vincoli di uguaglianza e diseguaglianza:

$$S = \{x \in IR^n \ : \ g_i(x) = 0, \ h_j(x) \leq 0 \}$$

in cui $g$: $IR^n \rightarrow IR^n$ e $h$: $IR^n \rightarrow IR^n$ sono vettori di funzioni assegnate che identificano i vincoli del problema stesso. Il problema di ottimo assume in tal caso la forma:

$$\min_x f(x)$$

$$soggetto\ a \begin{cases} g_i(x) \leq 0, & i = 1,2,\dots,m \\ h_j(x) = 0, & j = 1,2,\dots,n \end{cases}$$

I problemi vincolati mostrano un'applicabilità più elevata nella rappresentazione di processi decisionali complessi, dal momento che i sistemi reali prevedono solitamente diversi vincoli (fisici, tecnologici, logici), necessari per rappresentare tutte le condizioni che definiscono l'ammissibilità del problema stesso.

Un' ultima osservazione riguarda la caratterizzazione dei problemi di ottimizzazione relativa alle differenti assunzioni della funzione obiettivo e delle funzioni di vincolo. Il problema viene definito di ottimizzazione lineare se la funzione obiettivo $f(x)$ e le funzioni di vincolo sono lineari, viceversa viene definito di ottimizzazione non lineare se essi risultano non lineari. Questi ultimi, a loro volta, possono essere distinti in problemi di ottimizzazione intera, se le variabili del vettore $x$ sono vincolate ad assumere valori interi, e di ottimizzazione convessa se le funzioni $f(x)$, $g_i(x)$ e $h_j(x)$ risultano convesse. Tale caratterizzazione risulta infatti di grande rilevanza ai fini della ricerca della soluzione ottima del problema affrontato.

## 1.2. La ricerca dell'ottimo

Nei problemi di ottimizzazione non è detto che esista sempre una soluzione ottimale per il problema e, qualora esistesse, non è detto che essa è l'unica. Nella pratica ingegneristica, spesso il minimo globale risulta inoltre difficilmente ottenibile, o addirittura inesistente, a causa del verificarsi delle seguenti situazioni:

- l'insieme ammissibile $S$ è vuoto;
- l'insieme ammissibile $S$ non è vuoto ma la funzione obiettivo è illimitata inferiormente su $S$, cioè $\inf_{x \in S} f(x) = -\infty$ , e non esiste un valore minimo di $f$ su $S$;
- l'insieme ammissibile $S$ non è vuoto, la funzione obiettivo è limitata inferiormente su $S$, cioè $\inf_{x \in S} f(x) > -\infty$ , ma non esistono punti di minimo globale di $f$ su $S$.

In questi casi, in cui la ricerca di soluzioni globali può risultare complessa, per la limitatezza della funzione obiettivo o per la non chiusura dell'insieme ammissibile, può assumere particolare interesse la ricerca di soluzioni di tipo "locale".

Dato, se esiste, un intorno $U(x^*)$ di $x^*$ ,il punto $x^* \in S$ soddisfacente la relazione:

$$f(x^*) \leq f(x) \quad per\ ogni\ \ x \in S \cap U(x^*)$$

viene definito *soluzione locale* o anche *soluzione di minimo locale (o relativo*) mentre il corrispondente valore della funzione obiettivo $f(x^*)$ viene detto valore di *minimo locale* di $f$ su $S$.

Una soluzione locale $x^*$ di un problema di ottimizzazione deve soddisfare una condizione necessaria di ottimalità (CNO); ad esempio per la ricerca del minimo di un problema non vincolato essa consiste nel fatto che la derivata della funzione obiettivo f si deve annullare in $x^*$: $df(x^*)/dx = 0$. Tale condizione è di fondamentale utilità nei problemi di ottimizzazione; infatti, definito $\Omega$ l'insieme dei punti che soddisfa la CNO, la soluzione può essere cercata al suo interno piuttosto che sull'intero sistema ammissibile $S$ il quale risulta di solito molto più grande. Il generico punto ammissibile $x \in S$ può così non essere soluzione del problema se $x$ non è contenuto in $\Omega$, mentre può esserlo, ma non necessariamente, se $x \in \Omega$.
In generale, non è detto che una soluzione di minimo locale sia anche una soluzione di minimo globale; ovviamente un punto di minimo globale è necessariamente anche un punto di minimo locale, ma non è detto che valga il viceversa. Un modo per determinare un punto di minimo globale potrebbe consistere nel determinare, risolvendo le condizioni necessarie, tutti i punti

candidati scegliendo tra questi il migliore possibile. Questo approccio `e possibile solo in un numero limitato di casi particolarmente favorevoli, in cui la soluzione analitica delle condizioni di ottimo `è relativamente semplice.
Come illustrato nella seguente figura, a titolo di esempio, la funzione obiettivo presenta tre minimi locali di cui però solo la soluzione x* è un minimo globale.

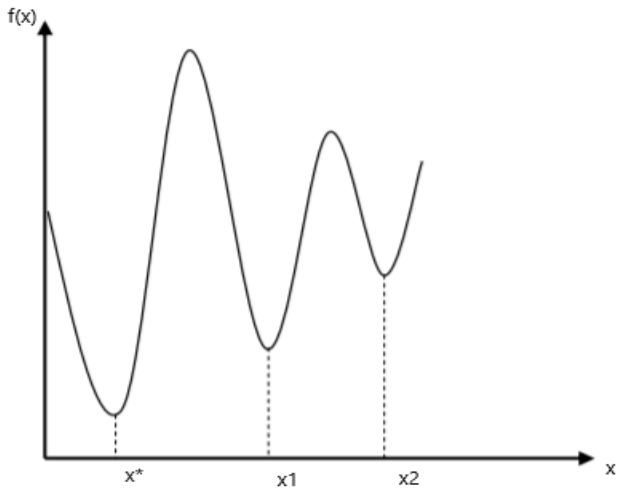



*Figura 1 – Funzione obiettivo e rappresentazione dei minimi*

Un ottimo può essere con certezza di tipo globale solo se il problema di ottimizzazione è convesso. Tale classe di problemi rivestono un ruolo importante nell'ambito dell'ottimizzazione non lineare e, affinché questo possa essere definito convesso, è necessario che la funzione obiettivo risulti essa stessa convessa all'interno del suo dominio di fattibilità. E' opportuno sottolineare che tutte le tecniche di programmazione non lineare possono al più determinare un minimo locale, o meglio punti che soddisfano le condizioni di ottimo locale e che potrebbero anche non risultare punti di minimo. Sotto opportune ipotesi di convessità, sia della funzione obiettivo $f$ sia dell'insieme ammissibile $S$, si verifica invece che ogni punto di minimo locale (un punto) di $f$ su $S$ è anche l'unico punto di minimo globale e quindi, un metodo per determinare un punto di minimo locale consente anche di determinare il minimo globale. Spesso però, tale ipotesi di convessità risulta troppo restrittiva e sono necessarie delle ipotesi più deboli per garantire che ogni punto di minimo locale sia anche globale.

I problemi di programmazione concava sono invece più "difficili" di quelli convessi; la principale difficoltà consiste nel fatto che essi presentano normalmente molti punti di minimo locale che non sono minimi globali. Tuttavia, la particolare struttura della funzione $f$ fornisce comunque informazioni importanti sui suoi punti di minimo globale.

Nel caso in cui il problema risulti non lineare non convesso è possibile determinare soltanto soluzioni ottimali locali, senza però garantire l'ottimalità globale. In questo caso, il risultato ottenuto è strettamente dipendente dal punto di partenza da cui si fanno partire le iterazioni, degli algoritmi numerici utilizzati, e per aumentare la probabilità di trovare un ottimo globale di può utilizzare un approccio definito Multiple Starting Point basato sull'utilizzo di diversi punti di partenza all'interno del dominio $S$; così facendo n potenziali punti possono risultare n diversi ottimi per il problema.

Si consideri, a titolo d'esempio, la seguente funzione $f(x)$ non convessa, con limiti $a \leq x \leq b$:

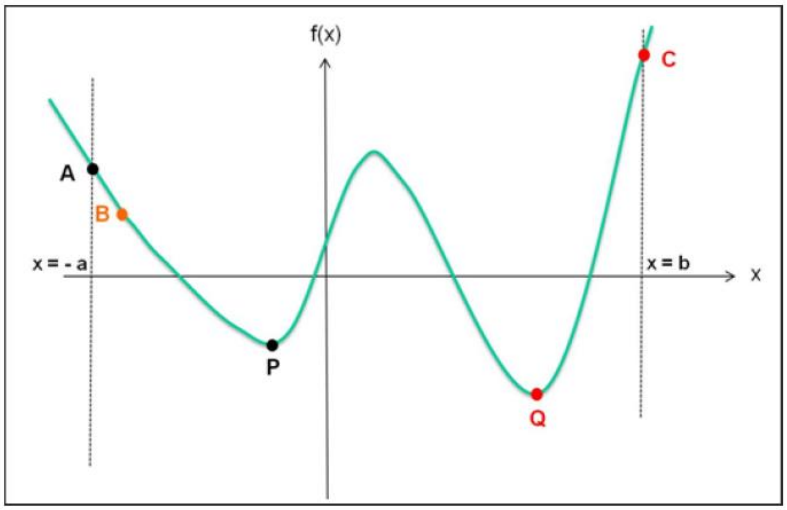



*Figura 2 – Esempio di funzione non convessa*

Se si conducesse la ricerca dell'ottimo a partire dal punto A, o analogamente dal punto B, si giungerebbe al punto P; se la ricerca venisse invece fatta partire dal punto C si arriverebbe al punto Q. Come è possibile osservare, variando il punto di partenza, non è garantita in ogni caso la convergenza ad una soluzione ottima ma si aumenta semplicemente la probabilità di trovarla.

## 1.3. Algoritmi di ottimizzazione

Nella pratica, la risoluzione dei problemi di ottimizzazione risulta complessa e non è possibile determinare una soluzione per via analitica; tale complessità è fortemente dipendente dal numero di variabili e di vincoli, ovvero dalla dimensione del problema, e dalla non linearità delle funzioni $f, g, h$ (anche solo di una tra queste) che caratterizzano il problema stesso.
Per la risoluzione di tali problemi assumono quindi un importante ruolo applicativo gli *algoritmi iterativi* cioè quei programmi di calcolo che consentono di determinare, data una approssimazione della soluzione $x^k$, una nuova approssimazione $x^{k+1}$ sfruttando una opportuna sequenza di operazioni; così facendo essi permettono di determinare una successione $\{x^k\}$ a partire da un'approssimazione iniziale $x^0$. Tali algoritmi, per come sono costruiti, presentano però una limitazione intrinseca poiché sono in grado di determinare solo quei punti del problema che soddisfano le *condizioni necessarie di ottimalità*, ovvero solo i punti dell'insieme $\Omega$ (insieme dei punti che soddisfa le CNO) definito *insieme bersaglio.* In tal senso, definito $X$ l'insieme delle soluzioni locali del problema (per cui vale sicuramente la relazione $X \subseteq \Omega \subseteq S$), l'efficienza dell'algoritmo può essere valutata in funzione della sua capacità di trovare punti di $\Omega$ piuttosto che su $X$.
La valutazione delle prestazioni di un generico algoritmo può quindi essere sintetizzata in due fondamentali aspetti:

- la *convergenza* all'insieme bersaglio dopo un certo numero di iterazioni *k,* ovvero la sua capacità di centrare, con la successione creata $\{x^k\}$, l'insieme $\Omega$;
- la *rapidità di convergenza* all'insieme bersaglio, ovvero la velocità con cui l'insieme $\Omega$ viene centrato dalla successione.

Per quanto concerne la convergenza, si può distinguere tra convergenza finita, se l'algoritmo converge a un punto di $\Omega$ dopo un numero finito di iterazioni (tipica dei problemi di Programmazione Lineare), o di convergenza asintotica se il numero di iterazioni $k \to \infty$, utilizzata in genere per la risoluzione di problemi di Programmazione non lineare.
Nella pratica, risulta però impossibile eseguire un numero di iterazioni infinite e quindi è importante prevedere un *criterio di arresto* per gli algoritmi a convergenza asintotica, cioè una regola che, dopo un numero finito di iterazioni, blocchi l'esecuzione dell'algoritmo stesso. Tale criterio di arresto risulta quindi importante nei problemi di ottimizzazione e, nella maggior parte dei casi, prevede di trovare, se non proprio un punto di $\Omega$, almeno una sua approssimazione accettabile. Un metodo potrebbe consistere nel fissare un limite ε sufficientemente piccolo e,

considerato l'insieme bersaglio $\Omega = \{\omega \in IR^n : \nabla f(w) = 0\}$, arrestare l'algoritmo al primo valore di *k* per cui si verifica che:

$$||\nabla f(x^k)|| < \varepsilon$$

Una prima caratterizzazione della convergenza dell'algoritmo si può avere, nel caso asintotico, in funzione del comportamento della successione $\{x^k\}$ per $k \to \infty$. Un primo caso si ha se $\lim_{k\to\infty} x^k = \omega \in \Omega$, ovvero se l'intera successione converge a un qualunque punto $\omega \in \Omega$; un secondo caso si ha invece se la successione $\{x^k\}$ ha almeno un punto di accumulazione $\omega \in \Omega$ e da essa si può estrarre una sottosuccessione $x^{k_j}, j = 1,2,..$.convergente a un punto $\omega \in \Omega$. Nei casi pratici, il secondo metodo, per quanto caratterizzato da requisiti meno restrittivi, può essere ritenuto soddisfacente e quindi, si ha che se $\{x^k\}$ ha almeno un punto limite, o di accumulazione in $\Omega$, è sempre possibile ottenere, per $k$ sufficientemente elevato, un punto $x^k$ capace di approssimare, con la precisione desiderata, un punto di $\Omega$.

Una seconda caratterizzazione della convergenza si può avere, inoltre, in relazione alla scelta dell''approssimazione iniziale $x^0$; se la convergenza ad un punto $\omega \in \Omega$ si consegue qualunque sia $x^0 \in IR^n$ si dice che l'algoritmo ha convergenza *globale*, per altri algoritmi invece si ha solo convergenza di tipo *locale*, ovvero la convergenza al punto $\omega$ è assicurata solo se $x^0 \in P$, con *P* intorno sferico aperto di $\omega$.

Per quanto concerne invece la rapidità di convergenza, si assume che la successione $\{x^k\}$ prodotta dall'algoritmo sia convergente ad un punto limite $\omega \in \Omega$ in modo che risulti $\lim_{k\to\infty} x^k = \omega$, e quindi che $\lim_{k\to\infty} || x^k - \omega|| = 0$. In tal caso, il metodo più comune per la valutazione di tale velocità di convergenza della successione si basa sull'analisi di come si riduce il termine $|| x^{k+1} - \omega||$ rispetto al termine $|| x^k - \omega||$, ovvero sull'analisi del rapporto:

$$\frac{|| x^{k+1} - \omega||}{|| x^k - \omega||}$$

Tale rapidità di convergenza può, a sua volta, risultare:

- *lineare*, se esiste un punto $c \in (0,1)$ tale che, per k abbastanza grande, si ha:

$$\frac{|| x^{k+1} - \omega||}{|| x^k - \omega||} \leq c$$

- *superlineare*, se si ha:

$$\lim_{k\to\infty} \frac{|| x^{k+1} - \omega||}{|| x^k - \omega||} = 0$$

- *quadratica*, se esiste $c > 0$ tale che, per k abbastanza grande, si ha:

$$\frac{||x^{k+1} - \omega||}{||x^k - \omega||^2} \leq c$$

Osserviamo che, essendo che se un algoritmo converge superlinearmente, converge anche linearmente; e se converge quadraticamente converge anche superlinearmente, la convergenza quadratica risulta più rapida della superlineare che a sua volta è più rapida della lineare.
Da un punto di vista pratico, le considerazioni appena fatte sulla velocità di convergenza non risultano però sufficienti a stabilire se un algoritmo ha prestazioni migliori di un altro, in quanto si fa riferimento solo al numero di iterazioni effettuate e non di quanto ognuna di esse sia onerosa da un punto di vista computazionale. Ad esempio, la singola iterazione di un algoritmo con rapidità lineare richiede calcoli più semplici rispetto agli altri due casi; e quindi diventa importante cercare un compromesso tra l'esigenza di limitare il numero delle iterazioni e quella di avere iterazioni troppo onerose.

Come già detto, i punti di minimo globale non sono caratterizzabili matematicamente in modo semplice e ciò si traduce, da un punto di vista teorico, in una difficile implementazione degli algoritmi numerici di ottimizzazione utilizzati per determinare tali punti di minimo. Infatti, tale complessità porta sia all'impossibilità di sfruttare il fatto che un punto prodotto dall'algoritmo non è un minimo globale e sia alla difficoltà di definirne dei criteri di arresto semplici e affidabili.
Al fine di superare tali limiti, in letteratura vengono proposti algoritmi che utilizzano due approcci differenti:

- algoritmi che utilizzano *informazioni globali*;
- algoritmi che utilizzano *informazioni locali*.

Gli algoritmi che utilizzano informazioni globali sul problema di ottimizzazione sono capaci di generare sequenze di punti con interessanti proprietà di convergenza. Sfortunatamente però, l'applicabilità di questa classe di metodi risulta limitata dal fatto che essi richiedono o ipotizzano "a priori" la conoscenza di informazioni sul problema tra cui il valore ottimo della funzione obiettivo, il numero di minimi globali e la convessità o concavità del particolare problema.
La seconda tipologia di algoritmi, al contrario, non necessita di tali informazioni sul problema ma solo di grandezze e quantità facilmente ottenibili durante le iterazioni dell'algoritmo stesso,

tra cui, ad esempio, i valori della funzione obiettivo $f$ e delle funzioni di vincolo nei vari punti prodotti dall'algoritmo. Essi utilizzano ad esempio metodi di interpolazione o di estrapolazione che spesso, per cercare di avere una convergenza ai minimi globali, cercano di estrarre delle informazioni sul comportamento globale della funzione $f$ sul dominio ammissibile $S$ andando a campionare, cioè valutare, $f$ in un numero sufficientemente grande di punti appartenenti all'insieme stesso. In funzione del tipo di campionamento effettuato è possibile distinguere tali metodi in deterministici e probabilistici.

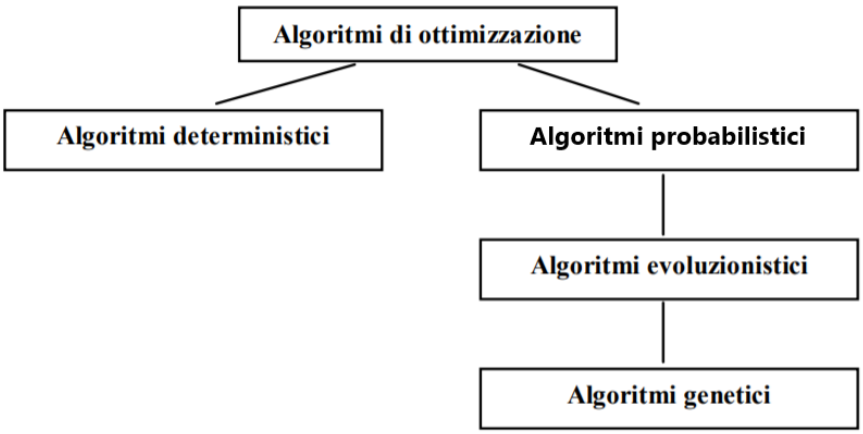



*Figura 3 - Classificazione degli algoritmi di ottimizzazione*

Mentre negli algoritmi deterministici i punti in cui viene valutata la funzione $f$ vengono determinati utilizzando le informazioni sulla funzione ottenute durante le iterazioni dell'algoritmo, nei probabilistici i punti sono dei vettori aleatori distribuiti uniformemente o secondo delle leggi dipendenti dalle informazioni estratte dall'algoritmo.

Tra gli algoritmi di tipo probabilistico emergono i metodi di tipo *multistart*, i metodi detti di "*simulated annealing"* e metodi che usano *"popolazioni" di punti*. Quest'ultima tipologia di metodi rientra nella categoria dei vari metodi di ottimizzazione globale di tipo eucaristico, ovvero quei metodi per cui non è possibile stabilire nessuna convergenza teorica e che nascono quindi dalla necessità di affrontare problemi di ottimizzazione reali. Come già visto, le proprietà di convergenza di un metodo sono infatti dipendenti dal fatto che, al crescere del numero delle iterazioni, la funzione obiettivo $f$ è valutata sull'intero insieme ammissibile; nella maggior parte dei casi tale approccio, anche se solo in maniera approssimata, risulta però impossibile da seguire e quindi questa categoria di metodi cerca di affrontare tale limite basando il suo approccio sui seguenti step:

- effettuare inizialmente un campionamento finito della funzione obiettivo $f$ su un insieme, detto *popolazione*, iniziale di punti in modo da ottenere informazioni sul comportamento globale della funzione obiettivo stessa sull'insieme ammissibile;
- migliorare iterativamente i punti che costituiscono tale popolazione attraverso la sostituzione dei punti in cui il valore della funzione obiettivo è maggiore, con dei nuovi punti ottenuti attraverso delle minimizzazioni locali "approssimate";
- ripetere iterativamente tale processo di miglioramento finché la popolazione non risulta "concentrata" nell'intorno dei minimi globali della funzione obiettivo.

Tra i metodi che usano popolazioni di punti le classi più note e, da un punto di vista pratico, più utilizzate sono:

- *i metodi "controlled random search";*
- *gli algoritmi genetici;*
- *gli algoritmi evolutivi.*

La prima classe di metodi utilizza una famiglia di punti, scelti inizialmente in modo randomico, e ad ogni iterazione, cerca di cambiare un solo punto di tale popolazione attraverso processi di minimizzazione. La seconda classe, nel determinare i nuovi punti, cerca di trarre inspirazione dai processi evolutivi biologici; le variabili del problema vengono codificate spesso attraverso stringhe di binari e i nuovi punti vengono generati attraverso la ripetizione di due operazioni genetiche su tali stringhe di cui verrà ampiamente discusso nel Capitolo 2. La principale differenza di questa classe di metodi che utilizzano popolazioni di punti rispetto ai metodi controlled random search, risiede nel fatto che ad ogni iterazione, invece di cambiare solo un punto della famiglia, si tenta di cambiarne un certo numero, spesso fissato. Infine, gli algoritmi evolutivi, che rientrano nella sottoclasse degli algoritmi genetici, aggiornano ad ogni iterazione effettuata tutti i punti della famiglia considerata.

## 1.4. L'ottimizzazione strutturale

In passato, la ricerca di strutture più efficienti veniva effettuata tramite procedure a tentativi, il recente avvento di tecniche avanzate di ottimizzazione strutturale (a partire da algoritmi di ottimizzazione innovativi per software) ha permesso di risparmiare notevolmente sui costi e ottenere risultati in un precesso di progettazione ottimale.

In ambito ingegneristico, per ottimizzazione strutturale si intende "la ricerca di forma, dimensione e tipologia ottimale di un determinato componente o di una struttura al fine di

aumentarne la resistenza o la rigidezza oppure ridurne il peso, la freccia massima o il costo di produzione". Nella sua eccezione più generale, fa parte dell'ottimizzazione strutturale anche l'ottimizzazione dei materiali; in base alla proprietà prestazionali dei materiali stessi, l'ingegnere deve selezionarli ed eventualmente progettarli o modificarli per una specifica applicazione.

I motivi che spiegano il perché l'ottimizzazione è ampiamente applicata nell'ambito della progettazione strutturale sono molteplici; il più importante risiede nel fatto che il processo di design delle strutture è molto simile al processo di ottimizzazione. Infatti, se si considera a titolo d'esempio di voler ridurre il peso di una struttura, tale riduzione può essere considerata equivalente alla funzione obiettivo di ottimizzazione e il problema viene ricondotto alla minimizzazione della funzione stessa. Inoltre, diverse condizioni di progettazione per il design strutturale possono essere facilmente trasformate in vincoli di ottimizzazione da soddisfare, e il problema viene facilmente ricondotto ad uno di ottimizzazione vincolata, con vincoli rigidi o labili rispettivamente espressi da funzioni di eguaglianza e diseguaglianza.

L'ottimizzazione strutturale, in funzione della caratteristica geometrica scelta come variabile di progetto, può essere classificata in tre differenti famiglie di ottimizzazione, che fanno riferimento a diversi aspetti della progettazione:

- Ottimizzazione dimensionale (o *size*);
- Ottimizzazione di forma (o *shape*);
- Ottimizzazione topologica (o *topology*)

### 1.4.1. Ottimizzazione dimensionale

La prima classe di ottimizzazione, in genere utilizzata per strutture di tipo reticolare, è la più semplice tra tutte e prevede di variare la dimensione del componente strutturale. Si tratta di un vero e proprio dimensionamento della struttura che, dato un dominio di progetto, una topologia (forma e architettura) e una configurazione geometrica iniziale e, dato inoltre, un set di proprietà iniziali degli elementi (area, spessore, inerzia) permette di definire la combinazione ottimale delle proprietà di ciascun elemento della struttura andando a cambiare la dimensione, ovvero la proprietà fisica degli elementi stessi. La principale caratteristica di un problema di ottimizzazione dimensionale è che il dominio di progetto, non solo è noto a priori, ma rimane fissato durante tutto l'intero processo. Un tipico problema $size$, è illustrato in figura 4. Esso è applicato a una trave elastica soggetta a note condizioni di vincolo e di carico, e il suo obiettivo

potrebbe essere quello di determinare l'andamento "migliore" del diametro degli elementi della trave, ovvero quell'andamento del diametro che minimizza (o massimizza) una specifica grandezza fisica come la cedevolezza, la deflessione o lo stress di picco.

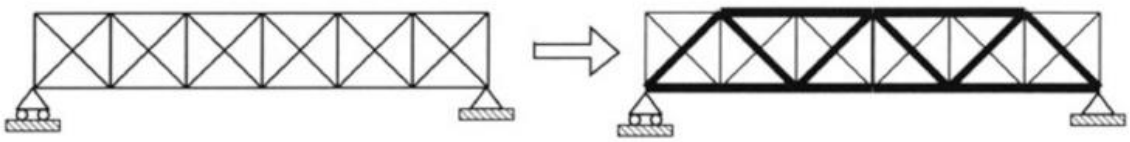


*Figura 4 – Ottimizzazione dimensionale*

Durante il processo di ottimizzazione è importante che esso garantisca, allo stesso tempo, che vengano soddisfatte delle condizioni sulle variabili di progetto, nel nostro caso il diametro degli elementi reticolari, e sulla risposta del sistema, come potrebbe essere lo stress in uno specifico punto.

Generalmente l'ottimizzazione dimensionale è eseguita dopo l'ottimizzazione della forma e l'ottimizzazione topologica poiché essa mira a trovare i valori ottimali dei parametri che definiscono le sezioni trasversali degli elementi strutturali. In particolare, nel caso di ottimizzazione di elementi *plates* (piaste) o *shell* (gusci), il loro spessore variabile o costante può essere adottato come variabile di progetto, o insieme di variabili di progetto, come illustrato nella seguente figura:

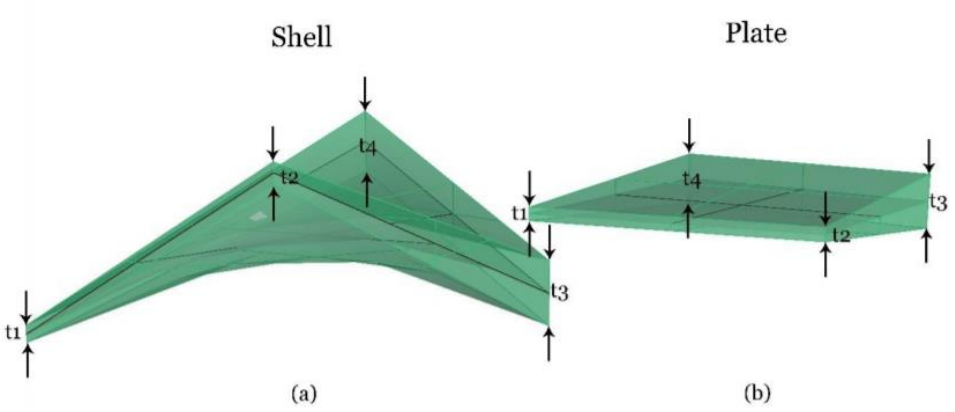



*Figura 5 - Esempio delle variabili di progetto per un elemento (a) guscio e (b) piastra*

A tal proposito, è prassi comune ottimizzare lo spessore variabile di coperture a guscio sottile al fine di migliorarne il comportamento strutturale (principalmente dipendente dalla loro forma) in termini di robustezza, rigidità e stabilità, garantendo una distribuzione delle tensioni interne più uniforme possibile per evitare effetti flessionali indesiderati.

Nel caso di ottimizzazione dimensionale delle strutture a telaio, ad essere assunte come variabili di design sono, generalmente, le aree della sezione trasversale dei membri. Raramente invece si decide di assumere singoli parametri che caratterizzano le sezioni trasversali degli elementi, come ad esempio i diametri delle sezioni circolari trasversali o gli spessori delle sezioni cave (figura 6), poiché essi determinerebbero un aumento significativo delle variabili stesse di progetto.

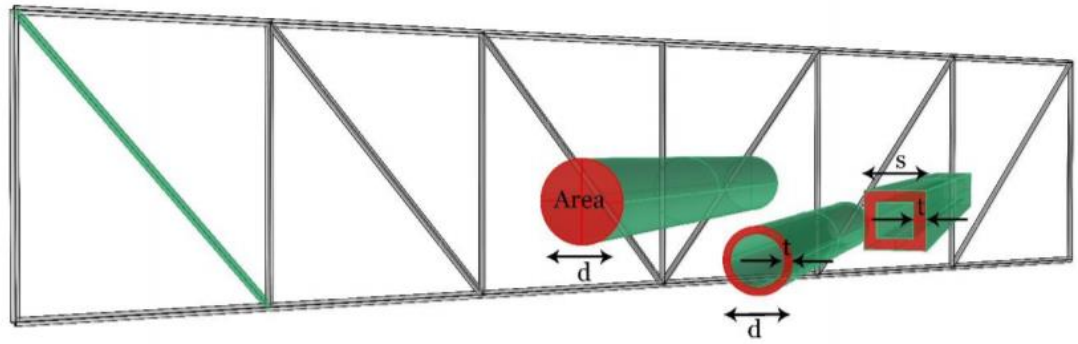



*Figura 6 - Esempio delle variabili di ottimizzazione dimensionale per elementi truss o strutture a telaio*

Inoltre, al fine di ridurre il numero necessario di variabili di progettazione, è possibile raggruppare gli elementi delle strutture a telaio in più gruppi in base alle loro funzioni strutturali o di alcune considerazioni geometriche (condizioni simmetriche). Nell'ottimizzazione di strutture discrete, quali le strutture a telaio, potrebbe invece essere conveniente adottare variabili di progetto discrete, cioè caratterizzate da un serie di valori isolati, i cui parametri vengono presi da un elenco di sezioni trasversali commerciali. Dal momento che, la risoluzione di un problema di ottimizzazione con variabili di progettazione discrete risulta più complesso che risolvere problemi simili con variabili continue, le variabili possono essere impostate come continue in un secondo momento arrotondando al numero intero più vicino.

### 1.4.2. Ottimizzazione di forma

L'ottimizzazione di forma (*shape*), è una sezione particolare dell'ottimizzazione strutturale che mira invece a trovare la forma ottimale degli elementi della struttura da ottimizzare date specifiche condizioni al contorno e dato un dominio di progetto e una configurazione geometrica iniziale. L'obiettivo è quindi quello di determinare la migliore forma, sconosciuta, del contorno del dominio, la quale dunque assume la valenza di variabile di progetto; in generale le variabili di progetto da considerare possono essere ad esempio la distribuzione di spessore in corrispondenza di un elemento della struttura, il diametro di un foro, il raggio di curvatura o qualunque altra misura caratteristica del dominio. Essa può quindi partire da una configurazione geometrica iniziale e arrivare, attraverso procedure di rigonfiamento o assottigliamento degli elementi, ad una geometria ottimizzata, come mostrato caso nella figura seguente:

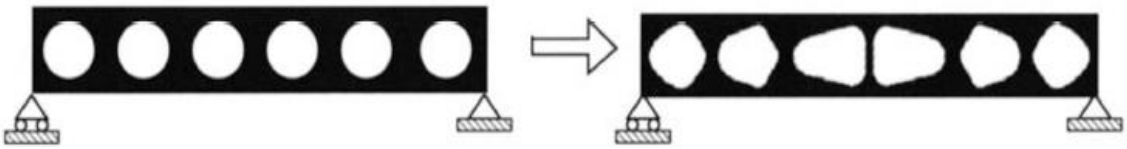


*Figura 7 – Ottimizzazione di forma*

Anche le coordinate nodali di una struttura al continuo o al discreto (opportunamente discretizzate in linee o elementi di superficie) possono essere direttamente assunte come variabili di progetto, come mostrato in figura 8:

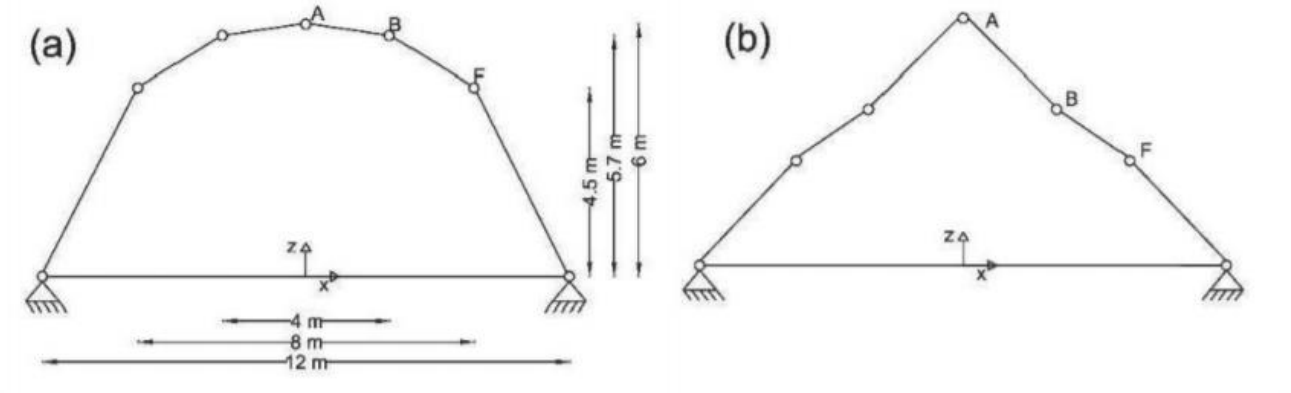



*Figura 8 - Ottimizzazione di forma di una struttura discreta*

Il processo per trovare la forma ottimale di una struttura per alcuni requisiti di progettazione, come il rispetto di determinate condizioni limite e carichi, proprietà dei materiali, sollecitazioni ammissibili e spostamenti, è anche chiamato "ricerca di forme". Prima dell'avvento di tecniche computazionali avanzate, ingegneri e architetti impiegavano modelli fisici, definiti sospesi, nella ricerca delle forme ottimali delle strutture; nel 1675 lo scienziato inglese R.Hooke scoprì la relazione tra una catena sospesa (che assume la forma di una catenaria sotto il suo peso proprio capace di resistere a forze di tensione) e un arco a compressione scoprendo che la forma delle corda, sotto un insieme di carichi e soggetta a forze di pura trazione, se irrigidita e capovolta, corrisponde a una "linea di spinta" di forze di compressione per un arco che supporta lo stesso insieme di carichi. La forma così ottenuta della corda tesa e dell'arco rovesciato compresso definiscono una forma funicolare per questi carichi. Antonio Gaudì e altri professionisti, si servirono successivamente di tali modelli rovesciati, o sospesi, e caricati per gravità come strumenti di rilevamento della forma per la progettazione di strutture a guscio, convalidando così la legge di inversione di Hooke. Anche se i modelli fisici possono sempre essere considerati validi per la ricerca di forme, sia di strutture bidimensionali sia tridimensionali, le tecniche più evolute sono state sviluppate sfruttando nuovi metodi informatici per la statica grafica affiancati da tecniche di "parametrizzazione" di cui verrà ampiamente discusso nel successivo capitolo.

### 1.4.3. Ottimizzazione topologica

L'ottimizzazione topologica, nota anche come OT, è infine la più generale forma di ottimizzazione strutturale la cui applicazione, nei vari campi dell'ingegneria, può portare a notevoli miglioramenti sia in termini di costo che di qualità di design. Essa consiste, dato un dominio di progetto (volume), nell'andare a ricercare la migliore distribuzione spaziale del materiale all'interno dello spazio di design, determinando quindi posizione, numero e interconnessione dei vuoti e dei pieni al suo interno come illustrato nella seguente figura:

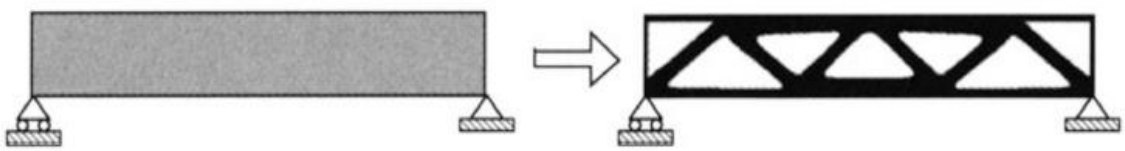


*Figura 9- Ottimizzazione topologica (OT)*

Nel metodo OT, la ricerca del design ottimo implica quindi la valutazione su tutto il dominio della migliore distribuzione di una densità del materiale fittizia ρ normalizzata rispetto alla densità nominale $\rho_0$ del materiale preso in considerazione, la quale può assumere valori compresi tra 0 e 1, a cui corrispondono lo 0% (vuoto) e il 100% (pieno) di $\rho_0$ . E' importante sottolineare che, la densità a cui si fa riferimento è quella di tipo strutturale, ovvero non al peso o alla massa dell'elemento ma a quanto esso contribuisce a dare rigidezza all'intero componente. In tal senso, tale algoritmo di ottimizzazione, consente, attraverso un processo iterativo, di aggiungere o sottrarre tale densità al dominio di progetto in funzione di un parametro indicativo spesso rappresentato dallo stato di sforzo o dall'energia di deformazione dell'elemento; in funzione di quanto l'elemento è sollecitato o caricato di energia esso viene quindi reso più o meno partecipe alla resistenza dell'intero componente.

In generale, esistono due diverse strategie di ottimizzazione topologica:

- OT di massima omogeneizzazione
- OT di minimizzazione della cedevolezza

La prima strategia consiste, definito un limite di sollecitazioni, nell'andare a garantire che all'interno del dominio il materiale sia soggetto tutto alla stessa sollecitazione, ovvero la sollecitazione limite, e rappresenta la classica teoria utilizzata per l'ottimizzazione della trave incastrata caratterizzata da un dominio a cuneo.

La seconda strategia, prevede come obiettivo la minimizzazione della cedevolezza, ovvero la massimizzazione della rigidezza globale della struttura, andando a definire un valore obiettivo di peso. Poiché banalmente la soluzione a tale problema sarebbe quella che prevede materia in modo continuo su tutto il dominio di progetto, è necessario introdurre un vincolo sulla quantità totale di materiale che può occupare il dominio stesso; quindi partendo ad esempio da un dominio pieno andare ad imporre una riduzione percentuale in peso da raggiungere. Un problema di ottimizzazione simile risulta quindi di grande interesse pratico- ingegneristico poiché, opportunamente tarato, permette di ridurre il peso di una struttura preservandone le sue caratteristiche in termini di rigidezza ai carichi esterni.

La formulazione matematica del problema di minimizzazione della cedevolezza fa uso del Principio dei Lavori Virtuali e assume la forma:

$$\min_{\rho} C(\rho) = U^T f = U^T K U$$

$$soggetto\ a \begin{cases} KU = f \\ V(\rho) < V_0 \\ 0 \leq \rho_e \leq 1 \end{cases}$$

In cui $K$ è la matrice di rigidezza globale, $U$ e $f$ rappresentano rispettivamente il vettore degli spostamenti e il vettore dei carichi esterni e $\rho = [\rho_1, \rho_2, \ldots \rho_n]$ è il vettore delle variabili di progetto dato dalle densità dei materiali. L'espressione $V(\rho) < V_0$ rappresenta il vincolo sulla quantità di materiale, $V_0$ è il limite ammissibile (volume di progetto) e $V(\rho) = \sum_{e=1}^{N} \rho_e v_e$ con $N$ il numero di elementi che discretizzano il dominio, ovvero le variabili di progetto, e $v_e$ il volume del generico elemento finito *e.*

A causa della loro complessità, i problemi OT vengono spesso risolti numericamente utilizzando algoritmi e procedure iterative. Esistono diversi approcci tra cui l'uso di algoritmi genetici, grazie ai quali è possibile la rimozione del materiale inefficiente tra un'iterazione e l'altra durante il processo di ottimizzazione. Senza entrare nella trattazione specifica, iterazione dopo iterazione, vengono infatti aggiornate le densità dei materiali associate ad ogni elemento finito, modellato tramite analisi FEM, attraverso diversi metodi come l'MMA, Method of Moving Asymptotes, basato sui risultati ottenuti da *analisi di sensitività.*

A titolo d'esempio, si riporta in Figura 10 un flusso computazionale di ottimizzazione topologica:

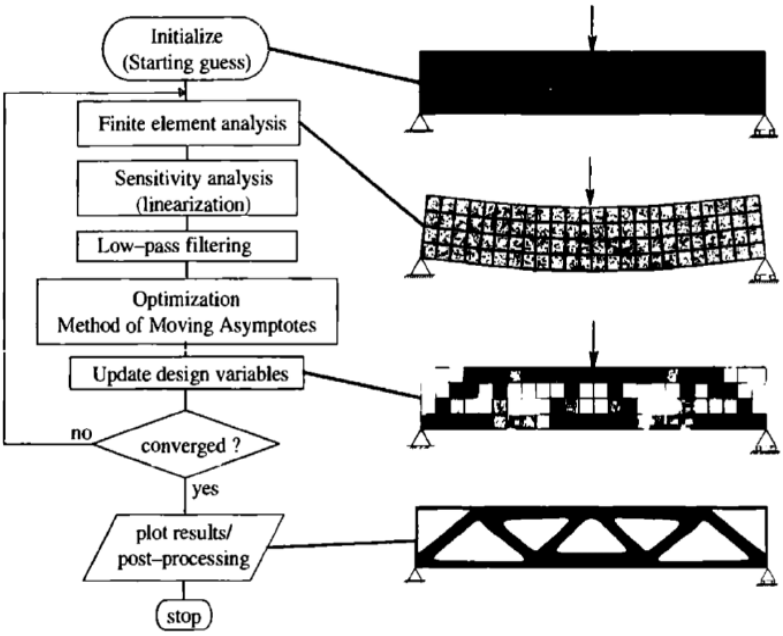



*Figura 10 - Flusso di ottimizzazione topologica*

Il flusso è articolato in cinque steps: viene prima di tutto definito il design iniziale caratterizzato da una distribuzione del materiale, o densità, omogenea, attraverso il metodo FEM viene modellato ogni elemento calcolando spostamenti e cedevolezza della struttura, si esegue l'analisi di sensitività, si aggiornano le variabili di design tramite il metodo MMA, si controlla, infine, la convergenza del risultato valutando se, tra l'ultimo passo e il precedente, la variazione di cedevolezza è trascurabile o inferiore ad un limite fissato.

Nel caso delle travature reticolari, l'ottimizzazione topologica assume una forma discreta (nota anche come ottimizzazione topologica del traliccio, o TTO) e l'obiettivo diventa quello di ottimizzare la connettività tra i nodi, le cui coordinate sono note e fisse, di una griglia. Questa tipologia di problemi di ottimizzazione può essere convenientemente formulato mediante il cosiddetto "ground structure method" (MP Bendsoe and Sigmund 2003); in questo approccio, il layout di una struttura reticolare può essere trovato consentendo un certo insieme di connessioni tra i punti nodali di una griglia fissa.

Infine, vale la pena sottolineare che l'ottimizzazione strutturale topologica può essere vista come una categoria dell'ottimizzazione di forma, anche se soggetta a vincoli progettuali (ad esempio i nodi fissi in caso di TTO).

# 2. Il design parametrico come nuovo approccio alla progettazione

## 2.1. Introduzione

Nel capitolo precedente è stato ampiamente discusso come le tecniche di ottimizzazione e le innovazioni tecnologiche siano in grado di supportare la progettazione e il come la ricerca delle prestazioni ottimali di una struttura è la base della fase di design, una volta definiti gli obiettivi che si vogliono perseguire, le quantità che possono essere variate per raggiungere tali obiettivi e le limitazioni alle possibili soluzioni raggiunte.

Tuttavia, all'interno di un processo di ottimizzazione risulta spesso indispensabile, o comunque utile, una definizione parametrica del problema progettuale ovvero la selezione appropriata di un set di variabili di progetto da cui dipenderanno la funzione obiettivo e le funzioni di vincolo del problema di ottimizzazione stesso (figura 11).

Soprattutto se il problema è di tipo *shape*, l'ottimizzazione delle strutture di grandi dimensioni (continue o discrete) caratterizzata da un numero elevato di nodi potrebbe richiede un numero elevato di variabili di progetto; in tal senso potrebbe quindi risultare vantaggioso adottare funzioni di forma parametriche, a seconda di un piccolo numero di parametri che possono essere assunti come variabili di ottimizzazione della forma per il problema considerato.

La tecnica di "parametrizzazione" di una forma da ottimizzare, attraverso opposite funzioni di forma, facilita infatti la modifica della forma in esame semplicemente variando i valori di pochi parametri e riducendo, notevolmente, il numero richiesto di variabili di progettazione della forma.

Il capitolo seguente ha lo scopo di introdurre al concetto del design parametrico e ai performanti strumenti di progettazione che rivestono un ruolo fondamentale nella fase preliminare di un processo di ottimizzazione strutturale, definiti appunto "parametrici". Essi vengono implementati tramite opposite piattaforme software in cui il designer è chiamato ad inserire una serie di parametri che vengono successivamente elaborati dal software stesso in una forma creata sulla base di tali valori; motivo per cui il design computazionale viene generalmente considerato come lo strumento che rende possibile l'utilizzo dei metodi parametrici all'interno dei processi di design. Già da questa sintetica definizione è intuibile la misura del salto di approccio che il *computational desig* offre rispetto ai metodi tradizionali, determinando il

passaggio da una logica di rappresentazione dell'oggetto architettonico a una sua simulazione mediante algoritmi.

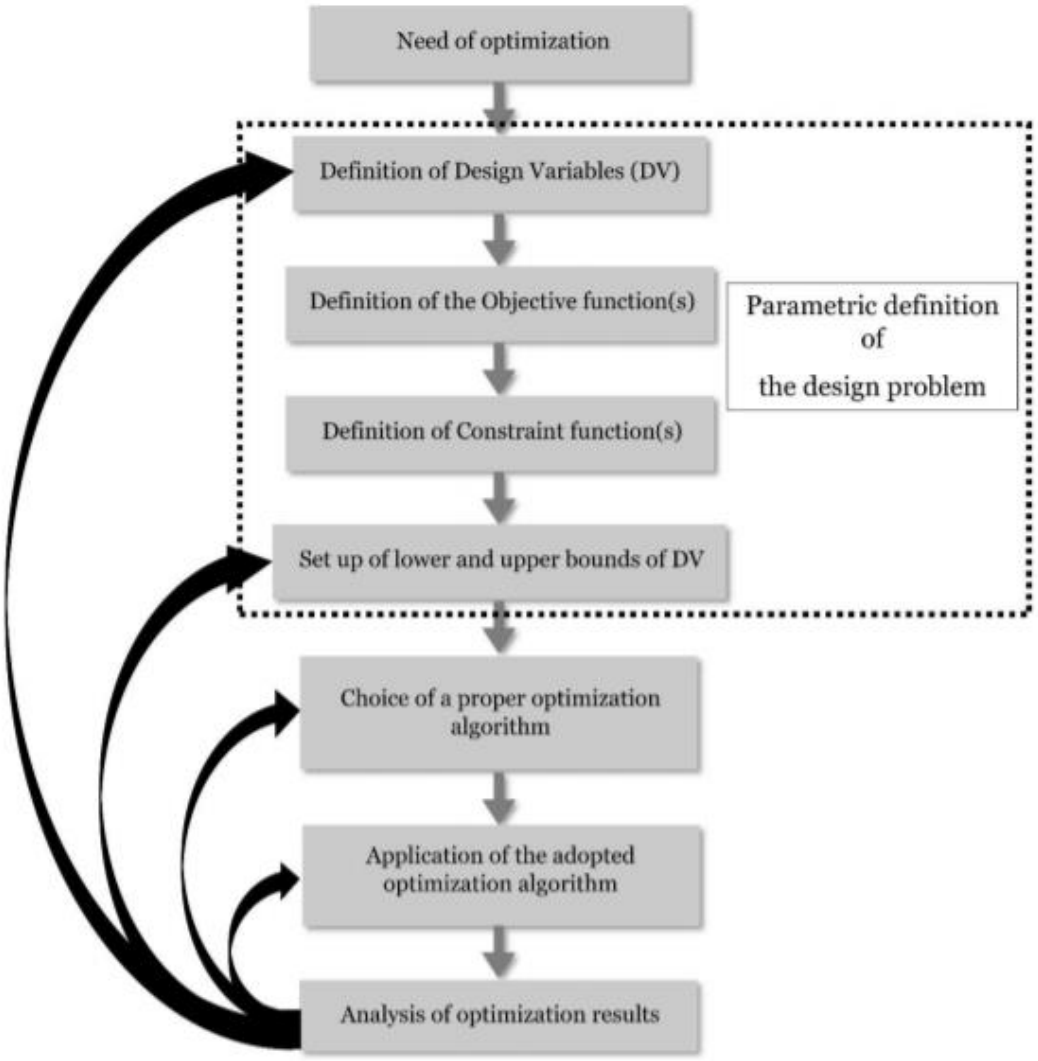



*Figura 11 – La "parametrizzazione" all'interno di un processo standard di ottimizzazione strutturale*

## 2.2. Il design parametrico

Negli ultimi anni il settore della progettazione strutturale ha focalizzato il suo interesse sulla realizzazione di geometrie irregolari, dalla forma sempre più complessa, utilizzando tecniche e materiali innovativi, specialmente nell'ambito della progettazione di opere in cui la forma svolge un ruolo importante come le coperture a superficie sottile o reticolare. Per queste particolari tipologie la complessità del comportamento strutturale, sotto un sistema di forze esterne applicate, e la difficoltà di eseguirne le analisi, ne limitano spesso la progettazione stessa

portando il progettista a valutare l'utilizzo di sistemi di disegno assistito parametrico che consentono di esplorare soluzioni progettuali differenziate nell'ambito di uno stesso concept.

L'approccio tradizionale alla progettazione strutturale prevede di utilizzare degli strumenti basati sul metodo agli elementi finiti (FEM) i quali permettono di ottenere risultati validi e accurati in termini di analisi sulla forma strutturale ma non consentono di modificarne facilmente e velocemente la forma, di visualizzare in tempo reale il modello e di analizzarlo; cosa molto utile quando si vuole rigenerare un nuovo modello in fase di progettazione strutturale come conseguenza di modifiche effettuale sul disegno architettonico.

Una soluzione per ovviare a questo problema è quella di servirsi di strumenti appropriati e versatili, da applicare sia nella fase di progettazione concettuale sia nella fase di analisi, basati su una modellazione parametrica che consente di apportare velocemente, e in maniera esatta, modifiche al progetto o di studiare simultaneamente, e in maniera relativamente semplice, una serie di possibilità o alternative partendo da dei presupposti numerici (dimensionali, quantitativi o formali) assegnati. Questo approccio rappresenta una modalità particolarmente interessante, specialmente se applicata nel campo del design e della progettazione di prodotti, poiché permette di concatenare una serie di modifiche e di automatizzare dei processi tutte le volte che si interviene su un determinato parametro (variabile iniziale di progetto) da modificare, ottenendo facilmente degli aggiustamenti del modello considerato e rendendo possibile l'inserimento delle proprietà dei materiali, dei vincoli di progettazione e delle logiche di composizione in parametri.

Prima di tutto è necessario chiarire brevemente cosa si intende per parametrico. Nell'ambito del design di tipo parametrico il risultato non è improntato sulla forma bensì è affidato alle relazioni tra le parti ovvero tra i dati (parametri) e le operazioni matematiche (componenti) le quali possono essere intese come le "regole del gioco" che seguiranno le informazioni iniziali per configurarsi in una forma finale. Definito quindi un insieme di dati e definito l'algoritmo matematico, ovvero il modo in cui tali parametri devono interagire tra loro, essi vengono inseriti nel software generando come output una forma, un modello tridimensionale "elastico" capace di aggiornarsi in qualsiasi momento al variare dei parametri stessi, rimanendo comunque coerente con quanto stabilito in partenza. La grande potenzialità offerta dalle tecniche parametriche risiede quindi nel fatto che esse consentono di capire e controllare il comportamento di un sistema complesso (ad esempio di un edificio) pianificandone in modo opportuno la risposta.

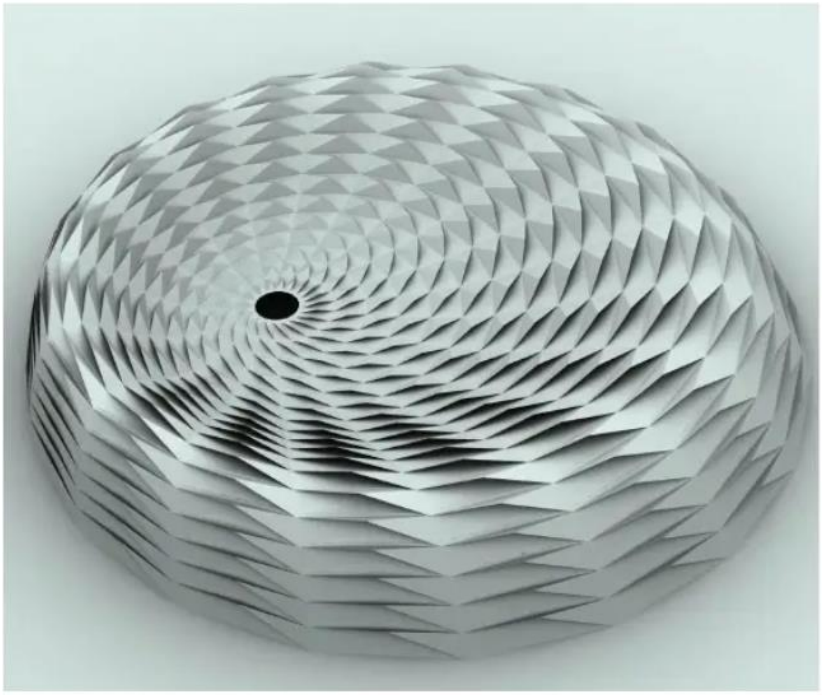


*Figura 12 – Guscio realizzato tramite tecniche parametriche (IaaC, 2007 – 08)*

La progettazione parametrica è quindi un tipo di modellazione tridimensionale che si differenzia dalla modellazione solida classica proprio perché si basa sulla messa in relazione delle componenti, o parti del modello, tra di loro o con numeri o tramite caratteristiche che vengono appunto definiti parametri. A differenza di un approccio design classico, in cui si ha una richiesta specifica a cui si risponde con un iter progettuale fatto da un'idea di base, uno sviluppo e un risultato (soluzione a cui si aspira), nel design parametrico la soluzione risiede nel processo stesso; il compito del progettista non è quindi quello di ottenere un risultato finale previsto già dal principio ma quello di progettare solo le condizioni, "regole" affinché tale risultato finale si possa generare, rappresentando qualcosa di prevedibile ma non di previsto. Il progettista interviene sul modello (rappresentazione al computer di un progetto) modificando i parametri per cercare diverse soluzioni al problema in questione e il modello risponde alle modifiche adattandosi o riconfigurandosi ai nuovi valori dei parametri senza cancellarli o ridisegnarli. Si parte quindi non dalla modellazione delle geometrie reali ma dalle logiche e regole che le definiscono; la forma, dunque, in un progetto parametrico non è definitiva, né definita a-priori piuttosto è incognita e rimane tale durante le fasi iniziali di impostazione dei parametri per poi essere svelata al progettista soltanto alla fine del processo come risultato ottimale.

I seguenti tre punti sono generalmente caratteristici del processo di progettazione parametrica:

- I designer progettano le regole e definiscono le relazioni logiche per la creazione dei modelli 3D

Questo rappresenta come già detto una delle principali differenze tra la modellazione computazionale tradizionale e il design parametrico in cui le regole diventano procedure di progettazione di base nella configurazione 3D. La progettazione dell'insieme di regole e delle loro relazioni logiche sta diventando l'obiettivo principale del pensiero progettuale in cui i progettisti impostano le variabili e i flussi di dati digitali, regolando i valori dei parametri e rivedendo di conseguenza le regole.

- I progettisti possono cambiare e modificare il loro design in qualsiasi fase del processo di progettazione

Nei processi di progettazione parametrica, il sistema di design è differenziato e correlato così come, nel modello parametrico, tutte le procedure e le attività di progettazione sono correlate tra loro e chiaramente distinte. Questo permette di avere un processo di progettazione aperto e flessibile in cui i designer possono tornare indietro in qualunque momento e rivedere parametri o regole da modificare per ottenere un diverso risultato.

- È possibile sviluppare alternative di progettazione in parallelo in qualsiasi fase

I progettisti spesso considerano un numero relativamente limitato di soluzioni alternative; il processo di progettazione parametrico consente invece, una volta implementate le regole e stabiliti i parametri, di avere un numero illimitato di alternative di design che possono essere generate, velocemente e facilmente, in modo parallelo.

Come è semplice intuire, la correlazione tra la progettazione parametrica e lo sviluppo tecnologico è strettissima ed è solo grazie all'avvento di programmi computazionali di modellazione 3D che diventa possibile calcolare rapidamente le soluzioni di analisi strutturale visualizzando con estrema precisione e in tempo reale le modifiche effettuate sui modelli, spaziando dalle più semplici travi fino alle più complesse tipologie di membrane. Anche per la progettazione è senza dubbio iniziata l'era virtuale in cui, grazie al design parametrico, è possibile implementare sia metodi matematici sia algoritmi nel campo dell'Intelligenza Artificiale che, una volta definiti una serie di regole, in forma di parametri, e numeri da inserire nel software, danno origine ad una forma in maniera quasi automatizzata e permettono di ottenere un numero virtualmente infinito di varianti dello stesso modello una volta modificati.

Per i motivi già esposti, e per molti altri ancora, è facile intuire come tale approccio parametrico, e automatizzato tramite tecniche computazionali, possa rappresentare oggi un nuovo modo di pensare di supporto al progettista che fino a poco tempo fa poteva contare solo sull'esperienza e sull'intuizione per risolvere i problemi di programmazione anche più semplici; i nuovi software, parametrici o non, sono uno strumento innovativo ormai parte integrante del processo di ottimizzazione che permettono a ingegneri e architetti di capire nel dettaglio il comportamento delle superfici che trasportano carichi, o la generazione di nuove e complesse forme architettoniche.

Nell'ottica di questo nuovo modo di percepire la programmazione, la metodologia usata per condurre un design ottimizzato può essere ricondotta ai seguenti due steps:

- Un design parametrico, ovvero lo studio per un numero minimo di parametri che meglio descrivono il modello in tutte le sue caratteristiche;
- Lo studio della configurazione ottimale al fine di determinare la struttura che meglio risponde alle esigenze del problema considerato.

## 2.3. Le origini del design parametrico in architettura

Nonostante la manipolazione di parametri è un argomento molto vivo all'interno della progettazione contemporanea, in architettura esso non rappresenta un concetto del tutto nuovo e, per quanto possa sembrare un tipo di ragionamento che ben si accosta all'uso di software specializzati, la nascita di tale approccio risale a ben prima dell'avvento del computer.

Il termine parametrico ha origini nel campo della matematica, ma c'è un dibattito su quando i progettisti hanno iniziato ad utilizzare questa parola nella realizzazione delle loro opere; il matematico Robert Stiles sostiene che la nascita dell'architettura parametrica è da attribuire all'architetto italiano Luigi Moretti che, già dal 1940, parla di architettura parametrica definendola, nei suoi scritti, come lo studio di sistemi architettonici che hanno l'obiettivo di "definire le relazioni tra le dimensioni dipendenti dai vari parametri". Moretti usa il progetto di uno stadio come esempio, spiegando come la forma dello stadio possa derivare da diciannove parametri tra cui, ad esempio, gli angoli di visuale e il costo del calcestruzzo (Moretti 1971, 207). Le versioni dello stadio furono presentate durante la sua mostra di Architettura Parametrica alla XII Triennale di Milano nel 1960 e, nei cinque anni successivi egli realizza il Watergate Complex che "si ritiene essere il primo grande lavoro di progettazione a fare un uso

significativo di strumenti computazionali per l'applicazione dei principi parametrici" (Livingston 2002).

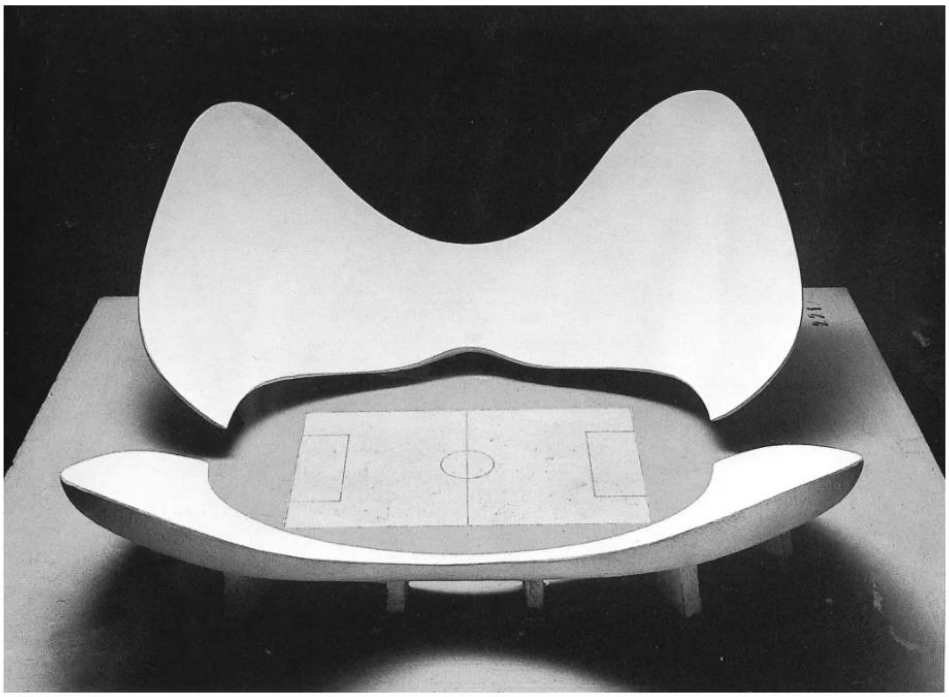


*Figura 13 - Un modello dello stadio realizzato da Luigi Moretti, esibito durante la mostra di Architettura parametrica nel 1960*

In quegli anni Moretti annuncia l'arrivo di una nuova architettura, anticipando considerazioni sul rapporto tra forma e struttura che verranno approfonditi e ampliati negli anni successivi.

Restando nell'Italia degli anni '70, un'altra importante innovazione si ha con il progetto del ponte sul Basenzo dell'ingegnere Sergio Musmeci grazie al quale si assiste ad una *inversione del processo progettuale*; egli abbandona infatti l'idea di forma generata a priori come punto di partenza del processo di progettazione affermando che quest'ultima può invece esserne vista come il punto di arrivo: "La forma diventa l'incognita, non le tensioni".

I primi esempi dell'utilizzo di "Parametrico" per descrivere modelli tridimensionali risalgono però a quasi cento anni prima degli scritti di Moretti; un esempio ne è il documento sul disegno di figure cristalline di James Dana del 1837 in cui spiega i passaggi generali per disegnare una vasta gamma di cristalli e le disposizioni per le variazioni utilizzando un linguaggio con parametri, variabili e rapporti. Ad esempio, in uno dei suoi passaggi, Dana dice al lettore di inscrivere un piano parametrico su un prisma: "Se il piano da introdurre fosse 4P2 il cui rapporto

parametrico è 4:2:1, dovremmo individuare allo stesso modo 4 parti di $e$, 2 di $\bar{e}$ e 1 di $\ddot{e}$ "; con questa citazione egli descrive la relazione parametrica tra tre parametri del piano (4:2:1) e la rispettiva divisione delle rette $e, \bar{e}, \ddot{e}$ (figura 14).

Il resto del documento possiede affermazioni simili che spiegano come vari parametri filtrano attraverso lunghe equazioni per modificare il disegno dei cristalli; tali equazioni cristalline sviluppate da Dana, assomigliano a quelle che sarebbero state utilizzate dagli architetti 175 anni dopo per sviluppare modelli parametrici nel campo architettonico.

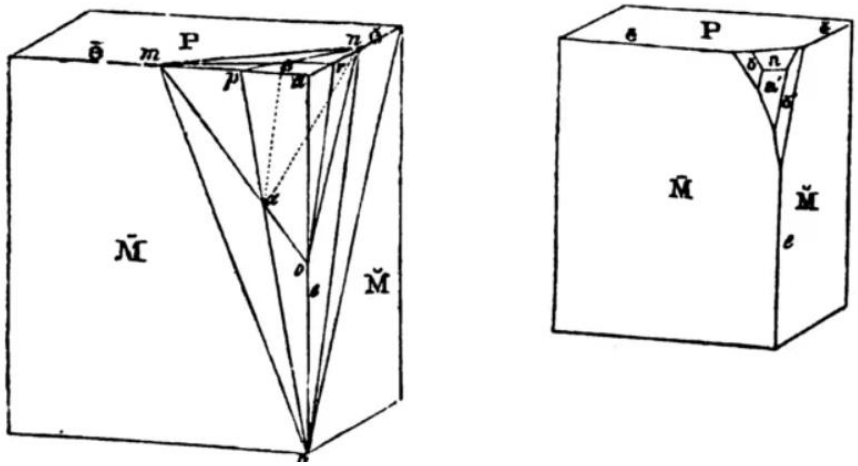



*Figura 14 - Istanze dei disegni in cristallo di James Dana che mostrano l'impatto della modifica del rapporto di smussamento del bordo (Dana 1837, 43)*

Il termine "parametrico" usato da Dana, o dai matematici oggi, ero inteso come ciò che la *Concise Encyclopedia of Mathematics* definisce "un insieme di equazioni che esprimono un insieme di quantità come funzioni esplicite di un numero di variabili indipendenti, note come parametri (Weisstein 2003)". Tale definizione tecnica-matematica, che può essere intesa come l'origine del termine parametrico, stabilisce due criteri:

1) Un'equazione matematica esprime "un insieme di quantità" con un numero di parametri;
2) I risultati (insieme delle quantità) sono correlati ai parametri tramite "funzioni esplicite".

Un esempio di equazione parametrica è rappresentato dalle formule che definiscono una curva catenaria:

$$x(a,t) = t$$
$$y(a,t) = a\cos h\left(\frac{t}{a}\right)$$

Le due formule soddisfano il criterio di un'equazione parametrica poiché esprimono un insieme di quantità (in questo caso una quantità *x* e una quantità *y*) in termini di un numero di parametri (*a*, che controlla la forma della curva e *t* che invece controlla la posizione del punto lungo la curva); inoltre, i risultati (*x* e *y*) sono correlati ai parametri (*a* e *t*) tramite funzioni esplicite ovvero non c'è ambiguità nelle relazioni tra queste variabili.

A parte i disegni dei cristalli parametrici di Dana, esistono molti altri esempi, risalenti ai primi anni del Novecento, basati sulla concezione matematica delle espressioni parametriche.

Indipendentemente dal fatto che Antonio Gaudì conoscesse o meno questi lavori precedenti, basati sulla definizione delle geometrie tramite equazioni parametriche, egli ha fatto certamente uso di tali metodi specialmente nella fase finale della sua progettazione architettonica basata, quasi esclusivamente, sulla realizzazione di superfici rigate matematiche (elicoidi, paraboloidi e iperboloidi) parametricamente associate a linee rette, booleani, rapporti e archi di catenarie.

L'uso delle equazioni parametriche può infatti essere visto in molti aspetti dell'architettura di Gaudì, ma è forse meglio illustrato dal suo uso del modello a catena sospesa.

*Figura 15 – Modello parametrico a catena sospesa*

Tale modello ha tutte le caratteristiche di un'equazione parametrica, vi sono infatti una serie di parametri indipendenti (lunghezza della stringa, posizione del punto di ancoraggio, peso del *birdshot*) e una serie di risultati (le differenti posizioni dei vertici dei punti sulle stringhe) che derivano dai parametri utilizzando funzioni esplicite, in questo caso leggi del movimento. Modificando i parametri indipendenti del modello parametrico, Gaudì ha generato versioni della Cappella *Colònia Guell* avendo la certezza che la struttura risultasse sempre in pura compressione.

Rispetto al precedente utilizzo delle equazioni parametriche da parte di scienziati e matematici, l'innovazione fondamentale del modello a catena sospesa è che esso è in grado di calcolare in modo automatico i risultati parametrici; piuttosto che calcolare manualmente i risultati delle formule parametriche della curva catenaria, Gaudì è infatti stato in grado di derivare la loro forma attraverso la forza di gravità che agiva sulle corde. Questo metodo di calcolo, successivamente ampliato dall'architetto Frei Otto, rendeva più facile l'esplorazione della forma sia vincolando a forme strutturalmente solide sia derivando automaticamente queste forme ogni volta che i parametri del modello venivano aggiornati.

Negli anni successivi, la digitalizzazione ha facilitato i calcoli non possibili con i modelli parametrici fino ad allora proposti; allo stesso modo in cui Gaudì e Otto usavano le leggi fisiche per accelerare il calcolo delle espressioni parametriche, l'architetto Ivan Sutherland cercò di utilizzare i computer (in un'epoca in cui essi funzionavano in modalità batch) per accelerare il calcolo di qualsiasi espressione parametrica. L'obiettivo di Sutherland era quello di creare "un sistema che permettesse ad un uomo e un computer di conversare" creando il primo programma di progettazione interattiva assistita chiamato *Sketchpad*. Grazie all'utilizzo e alla potenza di calcolo del computer TX-2, il designer poteva realizzare linee e archi che potevano poi essere messi in relazione tra loro con quelli che egli stesso chiamava vincoli atomici ovvero vincoli che avevano tutte le caratteristiche di un'equazione parametrica; ognuno di essi presentava infatti un insieme di risultati espressi come funzione esplicita di un numero di parametri indipendenti. A differenza dei modelli precedenti, queste equazioni parametriche non sono vincolate a leggi fisiche e, la novità offerta da Sketchpad stava nel fatto che i progettisti, oltre che esplorare le variazioni introdotte modificando i parametri, erano anche liberi di modificare le relazioni del modello ottenendo in modo automatico un ricalcolo e un ridisegno della geometria; pertanto, il controllo del designer sul software, così come per la maggior parte dei programmi di modellazione parametrica, non avviene solo attraverso i parametri del modello ma anche tramite le relazioni ad esso sottostanti.

Vent'anni dopo, un periodo in cui i computer stavano diventando economici al punto da consentire ad alcune persone di possederne uno ad uso personale, AutoCAD è stato rilasciato e rapidamente ha acquisito un ruolo dominante nel crescente settore della progettazione assistita da computer; erano finite le curve e le geometrie autoreplicanti poiché queste vennero sostituite in AutoCAD da comandi che consentivano al progettista di disegnare esplicitamente linee bidimensionali sullo schermo ma solo diciotto versioni dopo, AutoCAD 2010, il programma è stato ampliato con funzionalità parametriche e dichiarato dal comunicato stampa "una nuova capacità rivoluzionaria".

Nel 1988, la *Parametric Technology Corporation*, fondata appena qualche anno prima, ha distribuito quello che sarebbe diventato il primo software parametrico di successo commerciale chiamato Pro/ENGINEER nato con l'obiettivo di creare un sistema sufficientemente flessibile da incoraggiare l'ingegnere a considerare facilmente una varietà di progetti e in cui il costo delle modifiche apportate al design fosse il più vicino possibile allo zero. Con il nuovo software, gli utenti potevano associare parti della geometria insieme utilizzando varie equazioni parametriche, in modo esattamente analogo a quanto era concesso fare in Sketchpad, ma a differenza di quest'ultimo, la geometria era tridimensionale (piuttosto che bidimensionale) e le modifiche potevano propagarsi su più disegni diversi creati da utenti diversi.

A pari passo con lo sviluppo tecnologico, gli ex sviluppatori della *Parametric Technology Corporation* fondarono la *Revit Corporation* aspirando alla creazione del "primo modellatore parametrico per architetti e ingegneri per la progettazione di edifici"; sebbene Revit utilizzi equazioni parametriche per effettuare revisioni automatiche del modello, a differenza dei software di modellazione Sketchpad e Pro/ENGINEER, le relazioni parametriche vengono nascoste dietro un'interfaccia.

Da quel momento in poi, la modellazione parametrica si è fatta strada nei progetti anche attraverso le interfacce di scripting, consentendo ai progettisti di "scrivere" codici al fine di automatizzare parti del software. Lo script, caratterizzato da parametri di input, funzioni esplicite e output, può essere visto come la moderna definizione del concetto matematico di parametrico; osservando che i sistemi parametrici si basano principalmente su principi algoritmici (poiché un algoritmo prende un valore o un insieme di valori con input, esegue una serie di passaggi computazionali che trasformano gli input e infine produce un valore o un insieme di valori come output) gli script sono intrinsecamente parametrici e le interfacce, accessibili ormai nella maggior parte dei pacchetti software, sono predisposte alla creazione di modelli parametrici.

Lo scorso decennio ha infine visto l'emergere di un nuovo tipo di interfaccia di scripting, definita visiva, ampiamente utilizzata oggi nell'ambito della progettazione; la programmazione visiva implica la rappresentazione dei programmi non più in forma testuale ma tramite diagrammi di flusso. Essa si basa quindi su grafici che mappano il flusso di relazioni dei parametri attraverso funzioni definite dall'utente per poi concludersi, solitamente, con la generazione della geometria finale; le modifiche ai parametri o alle relazioni del modello provocano la propagazione delle modifiche stesse, attraverso le funzioni esplicite, per ridisegnare automaticamente la geometria. In quanto tali, la programmazione visiva rappresenta ancora un ulteriore metodo per la creazione di un modello parametrico a oggi ampiamente utilizzato soprattutto durante la fase di progettazione.

Come è facilmente intuibile, dal momento in cui i computer hanno iniziato a supportare ingegneri e architetti, simulando lo spazio e le articolazioni geometriche, sono diventati uno strumento fondamentale nel processo di design grazie ai quali la geometria computazionale ha potuto dare vita a strumenti come gli algoritmi generativi; sebbene infatti i software 3D aiutano a simulare quasi tutto lo spazio visualizzato, è la nozione di algoritmo generativo che rende oggi strumenti del design, come la "progettazione parametrica", così potente nel campo dell'architettura.

Solo nell'ultimo decennio quindi la modellazione parametrica è passata dall'essere un metodo matematico impiegato da Gaudì e altri ingegneri all' essere una parte regolare della pratica architettonica; l'esprimere le intenzioni di progettazione tramite parametri e funzioni esplicite richiede sicuramente un modo di pensare diverso da quello a cui la maggior parte dei designer è sempre stato abituato secondo cui è il progettista, in quanto tale, a decidere gli obiettivi da perseguire, i parametri, per arrivare alla definizione di una forma univoca totalmente controllata e conosciuta in ogni suo minimo dettaglio. In un modo di pensare parametrico, o algoritmico, a cui la maggior parte dei professionisti si sta ormai approcciando, al designer viene richiesto di distaccarsi completamente dalla necessità di controllare la forma per delegare allo strumento digitale parte del suo lavoro e focalizzarsi sull'aspetto decisionale. Ovviamente questo non implica che egli perda le sue capacità di controllo all'interno del processo; la scelta dei parametri rimane infatti di competenza del designer e, data l'importanza della loro definizione (per poter poi dare al software gli input necessari all'elaborazione), nella progettazione integrata alla programmazione, che sia essa visiva o no, il fulcro del processo si sposta dalla creazione dell'elaborato finale alla scelta dei parametri e del codice (compresi i relativi algoritmi).

Mentre in passato la realizzazione di metodologie sostanzialmente nuove di progettazione con gli strumenti tradizionali era soltanto una prerogativa di pochi liberi pensatori visionari, come Frei Otto o Antoni Gaudi, ora, grazie al metodo parametrico e alla produzione digitale, questa si è trasformata in un campo di ricerca fiorente nell'ambito del design. Le nuove possibilità digitali di progettazione e produzione a livello internazionale hanno infatti permesso di realizzare affascinanti esperimenti di costruzione che reinterpretano nei modi più disparati l'interazione di forma, funzione, materiale e produzione. I progetti pilota che sono stati implementati mostrano già oggi il cambiamento che potranno subire in futuro le città grazie all'impiego del metodo parametrico.
Si riportano di seguito, a titolo d'esempio, alcune tra le principali opere simbolo dell'architettura parametrica.

Il primo esempio è il *Padiglione Philips*, ideato da Corbusier e Xenakis intorno agli anni '60, in cui il disegno parametrico (tridimensionale) sostituisce la classica modellazione solida e le forme geometriche parametriche mettono in relazione la staticità di una normale struttura alla sinuosità dell'ambiente circostante.

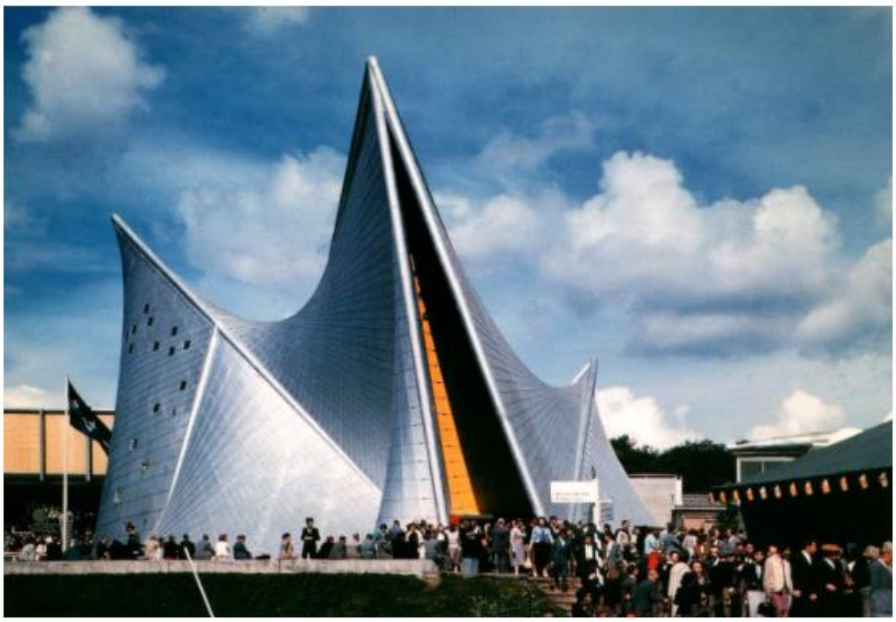


*Figura 16 -Padiglione Philips di Corbusier e Xenakis, 1958*

Il secondo esempio è il Palazzetto dello Sport a Roma, relativo all'ingegnere italiano Pier Luigi Nervi, le cui opere diventano sin da subito emblema di un nuovo modo di pensare l'ingegneria civile basato sulla connessione tra progettazione statica e richiamo artistico.

L’ultimo esempio è invece relativo all’opera di Zaha Hadid e Patrick Schumacher; completata nel 2013 essa rappresenta un’icona tra le strutture parametriche dell’ultimo decennio il cui processo di costruzione, caratterizzato da ricche e complesse forme sinusoidali, è stato notevolmente semplificato dall’utilizzo di algoritmi computazionali.

*Figura 17 – Palazzetto dello sport di Pierluigi Nervi, Roma*

*Figura 18 – Heydar Aliyev Center di Zaha Hadid, Baku*

## 2.4. Vantaggi e svantaggi della progettazione parametrica

Questo paragrafo pone l'attenzione sui vantaggi e gli svantaggi del design parametrico e la sua distinzione rispetto ai metodi CAD tradizionale e al Building Information Modeling (BIM).

Prima di analizzare nel dettaglio i lati positivi e negativi della progettazione parametrica, viene però presentata una breve analisi basata sulle risposte a un questionario che forniscono informazioni su cinque elementi, vale a dire finanza, gestione dei dati, tempo di apprendimento, problemi di utilizzo e cambiamenti negli approcci alla progettazione, chiarendo ciò che gli architetti pensano relativamente ai cinque punti precedenti (Figura 19).

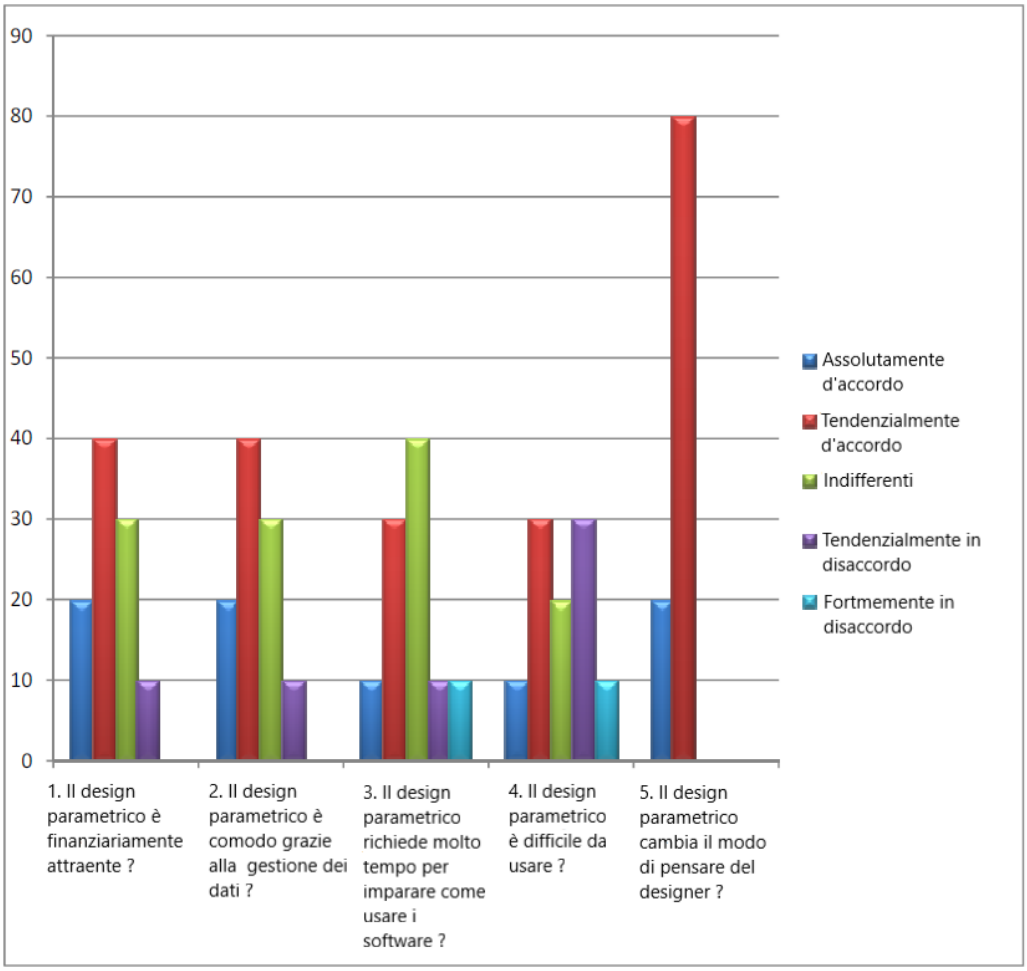



*Figura 19 - La valutazione della progettazione parametrica basata su cinque categorie: finanza, gestione dei dati, tempo di apprendimento, problemi di utilizzo e cambiamenti agli approcci dei designers; la comparazione è effettuata mediante una scala Likert a cinque punti.*

Come il grafico illustra chiaramente: per la domanda cinque, quasi tutti gli intervistati credono che il design parametrico cambi il modo in cui pensano i progettisti. In termini di difficoltà dell'utilizzo di programmi parametrici (domanda quattro), le risposte dei partecipanti sono

fermamente uguali; inoltre, mentre un terzo di loro non è d'accordo sul fatto che il design parametrico sia difficile da usare, un terzo concorda con questa posizione. Il resto (20%) non ha una invece una posizione distinta. In termini di finanza, gestione dei dati e tempo di apprendimento, la scala Likert è più focalizzata sugli elementi intermedi (tendono a non essere né d'accordo né in disaccordo) piuttosto che sui due estremi, il che probabilmente mostra che gli architetti sono in qualche modo ambivalenti nel riconoscere l'oggetto in questione come merito parametrico.

Questa breve panoramica delle opinioni degli intervistati fornisce un'altra serie di domande basate sull'idea che il design parametrico offre una visione distinta del design, concentrandosi su alcuni aspetti fondamentali del processo di progettazione e gestendoli in modo diverso. Le sezioni successive stabiliranno una comprensione più profonda dei vantaggi, degli svantaggi e delle distinzioni della progettazione parametrica rispetto ad altre metodologie.

### 2.4.1. I vantaggi del design parametrico

I motivi per cui un tipico studio di architettura utilizza un approccio parametrico possono essere diversi: mentre alcune aziende seguono una strategia competitiva semplicemente tenendosi al passo con i software più recenti, come i pacchetti parametrici, la maggior parte tende a vedere i parametri attraverso l'obiettivo di funzionalità ovvero come miglioramento delle opportunità di progettazione.

In generale, la capacità di esplorare rigorosamente più alternative di progettazione e quindi di vedere migliori soluzioni emergenti dai problemi di progettazione è indicata come il principale vantaggio della progettazione parametrica. Inoltre, a differenza del CAD tradizionale, che dipende ancora in larga misura dallo schizzo o dalla modellazione fisica, alcuni dei vantaggi della progettazione parametrica sono stati visti, sin dall'inizio, nell'esplorazione delle possibilità di progettazione; è noto infatti che, quando gli architetti pensano specificamente alle strutture a forma libera, il design parametrico offre loro "una grande opportunità per esplorare forme più eccitanti".

Tuttavia, esistono grandi vantaggi anche nelle fasi successive del processo di progettazione per l'automazione della documentazione di costruzione e dei livelli superiori di controllo architettonico nella produzione; in modo conciso, la progettazione parametrica può colmare il divario tra la progettazione e la produzione delle strutture. Roland Hudson, architetto e

ricercatore di la progettazione parametrica descrive le opportunità che la progettazione parametrica può offrire:

*"Vedo specificamente opportunità che la progettazione parametrica presenta nell'assistenza e nello sviluppo della conoscenza di un problema di progettazione, formalizzando o acquisendo tale conoscenza in modo esterno, sviluppando strategie per la costruzione, conducendo indagini di progettazione e, infine, implementando un metodo di documentazione di costruzione. Tutti possono e devono essere avvicinati simultaneamente e ciclicamente durante la fase di progettazione fino a quando il processo converge su un approccio che abbraccia tutti".*

Attraverso la ricerca nella letteratura, una delle questioni fondamentali che viene spesso citata è la capacità di creare relazioni tra oggetti, utilizzando equazioni per definire la geometria associativa. Robert Woodbury indica che la definizione delle relazioni non era stata precedentemente considerata come parte del pensiero progettuale, poiché le attività definite convenzionali nel design erano "aggiungi e cancella"; mentre ora, i progettisti hanno due capacità extra, ovvero 'relazionare' e 'riparare'. Per Woodbury, la "relazione" richiede un pensiero esplicito sul tipo di relazione, e la "riparazione" avviene dopo una cancellazione, quando le parti che dipendono da una parte cancellata sono nuovamente correlate alle parti che rimangono. Quindi, questi due atti hanno imposto cambiamenti fondamentali ai sistemi del passato ed è ragionevole considerarli come vantaggi della progettazione parametrica.

Oltre a questi problemi, gli architetti riconoscono molti altri elementi di differenziazione sulla progettazione parametrica rispetto ai metodi tradizionali di progettazione. Questi problemi possono essere suddivisi in tre classi, vale a dire: l'ottimizzazione del processo di progettazione, la capacità di realizzare una gamma di soluzioni per il problema e l'ingegnerizzazione del processo di progettazione in modo più efficiente.

- <u>L'ottimizzazione del processo di progettazione:</u>

Come già ampiamente discusso nel capitolo precedente, per ottimizzazione si intende letteralmente uno stato "*optimus*", o desiderabile, che soddisfa le condizioni del problema tra una serie di opzioni. I metodi di ottimizzazione possono essere suddivisi in approcci stocastici e deterministici. Mentre 'stocastico' (che deriva dal greco '*stokhastikos*' e significa mirare o tirare con un arco a un bersaglio) offre una caratteristica casuale, nel metodo deterministico di ottimizzazione non c'è casualità e si presume che i valori siano accurati e ogni azione determina con precisione lo step successivo; pertanto, la stessa routine di ottimizzazione ripetuta con le

stesse condizioni di partenza per lo stesso numero di iterazioni produrrà ogni volta lo stesso risultato.

In un certo senso, la progettazione parametrica utilizza entrambi questi metodi (sebbene sia spesso più deterministica che stocastica) nell'ottimizzazione dello stato di design, che include soluzioni di progettazione e percorsi che conducono ad esse, aiutando gli architetti a stabilire vincoli e misurare le cose in modo diverso; si può quindi affermare che, per quegli architetti che cercano un design ottimale, il design parametrico è l'ideale. Vikram Kaushal, architetto praticante e docente presso la Manchester School of Architecture (MSA), spiega che, nell'architettura commerciale, raggiungere questo punto ottimale è significativo e, la progettazione parametrica diventa quindi estremamente vantaggiosa in questo caso.

A causa della sua considerazione di parametri che possono controllare l'intero progetto, il progetto parametrico è finanziariamente ben definito, fornendo agli architetti un alto livello di controllo sul costo del loro progetto. Inoltre, può portare benefici anche ai clienti monitorando l'economia del design e di conseguenza può aiutare il progettista a minimizzare il rischio dello sviluppo “integrando quante più informazioni e parametri possibili in un'unica piattaforma che può abbracciare l'intero flusso di lavoro”.

- Una gamma di soluzioni per il problema di progettazione

Questo vantaggio è forse il più evidente tra le caratteristiche del design parametrico; impostando infatti collegamenti dinamici tra tutte le decisioni prese durante il processo e fornendo una logica associativa, gli architetti possono semplicemente cambiare un'opzione di progettazione con un'altra e produrre una gamma di soluzioni invece di fornire una sola risposta al progetto problema. In effetti, questo aspetto del parametrico ha origine dal concetto di 'relazione' che è stato discusso in precedenza e, sebbene sia un chiaro vantaggio a prima vista, è anche una sfida, perché richiede il chiarimento delle interdipendenze create da esigenze diverse. Matthew Smith spiega che, nella progettazione parametrica, il processo decisionale è molto più semplice proprio grazie alla vasta gamma di soluzioni disponibili:

*“Penso che una delle cose buone sia che ti fa pensare alle relazioni tra i diversi elementi del design, alla relazione nel contesto con vari parametri; non pensi a una soluzione fissa, stai pensando a 'cosa sono i fattori, le influenze e le cose che possono plasmare questa struttura o questo progetto '. Possono essere prese molte decisioni diverse. Come architetto, non stai aggiungendo i valori proprio come un lavoratore manuale; stai aggiungendo valore nel progettare e pensare al problema, non cambiando manualmente ogni riga; questo è un grande*

*vantaggio. Pensi al problema e indovina in un modo più olistico, pensi al problema e cerchi di risolverlo; fai una serie di soluzioni e risultati diversi, perché puoi variare le cose non manualmente".*

Sebbene tali capacità sembrino un grande vantaggio, i progettisti devono essere cauti, poiché un errore nella modifica di un parametro può avere un effetto a catena sul progetto e questo può essere ancora più problematico quando è qualcun altro, piuttosto che il designer originale, a modificare un qualunque parametro; pertanto, l'applicazione di metodi parametrici, in questo senso, richiede una visione manageriale superiore alle capacità dei professionisti in uno studio di architettura, un livello di gestione che probabilmente può essere raggiunto solo in studi di grandi dimensioni.

- Ingegnerizzazione del processo di progettazione

Uno dei notevoli vantaggi della progettazione parametrica è la sua capacità di consentire una gestione efficace del processo di progettazione. In altre parole, offre all'architetto la possibilità di ingegnerizzare il processo di progettazione offrendo i seguenti vantaggi:

- velocizzare l'intera attività di progettazione;
- eliminare azioni noiose e prendere molte decisioni diverse;
- aiutare anche gli architetti ad accelerare le trattative con il cliente.

Probabilmente, è vero che i clienti sono molto più interessati al risultato di un progetto che al suo processo; tuttavia, nelle fasi successive della progettazione, quando le alternative si presentano per competere tra loro, è inevitabile la necessità di avere una conversazione efficace con il cliente per finalizzare ciò che è stato progettato fino ad ora. Gli architetti ritengono che la progettazione parametrica possa migliorare le negoziazioni con il cliente perché può rendere l'ambiente di progettazione flessibile e pronto ad accogliere eventuali modifiche successive, consentendo agli architetti di rispondere rapidamente al cliente in diverse situazioni.

### 2.4.2. Gli svantaggi del design parametrico

Alla luce di portare una prospettiva diversa al design, gli architetti sostengono che il design parametrico porta anche degli svantaggi alla loro pratica. Alcuni difetti derivano dalla novità dei programmi per computer utilizzati nella progettazione parametrica; tuttavia, altri derivano dallo stesso approccio parametrico, che non solo cambia il modo in cui gli architetti progettano, ma altera anche notevolmente il ruolo di un architetto in un ufficio di progettazione.

Un problema che potrebbe essere correlato all'approccio parametrico in generale è che non risolve mai quali parametri sono necessari per la progettazione; in altre parole, in una certa misura non fornisce un quadro metodologico per la progettazione e, di conseguenza, gli architetti devono ancora elaborare la maggior parte delle parti del progetto nelle loro menti. Strategie computazionali come il metodo algoritmico cercano di colmare questa lacuna, sebbene in questi approcci il ruolo dell'architetto sia spesso soppiantato dal software.

Un altro problema deriva dal fatto che la maggior parte dei programmi parametrici sono stati progettati e collegati con "un allineamento del flusso di lavoro tradizionale in mente, ma consentono una maggiore riflessione sui processi". Inoltre, gli operatori di questi sistemi devono prevedere in anticipo tutte le direzioni del progetto per creare le geometrie e costruire le interrelazioni. I programmi parametrici dovrebbero essere progettati con un approccio parametrico e gli sviluppatori dei pacchetti parametrici hanno bisogno di una vera comprensione di questo approccio, dal momento che "progettano un progetto" in modo efficace.

L'altro difetto generale, non solo della progettazione parametrica stessa, ma di tutti i pacchetti software, è la necessità di disporre di software aggiuntivi per l'analisi dei moduli; sebbene i pacchetti parametrici abbiano cercato di colmare questa lacuna progettando una struttura per la generazione di moduli, la sfida rimane come connettere il software di generazione di moduli a un software di analisi dei moduli.

Oltre a questi problemi fondamentali, gli svantaggi della progettazione parametrica possono essere classificati in quattro categorie principali, raffiguranti forse il motivo per cui la progettazione parametrica è ancora ai margini o meno conosciuta tra gli architetti nonostante la sua preminenza discussa: complessità inutile con troppe informazioni, il problema della paternità, vincolare la creatività con una struttura reattiva e difficoltà di apprendimento e formazione.

- Complessità inutile con troppe informazioni

Uno dei problemi di fondo, che a prima vista sembra rappresentare una seria sfida per la progettazione parametrica, è la complessità dei pacchetti parametrici. Secondo Aish e Woodbury, la modellazione parametrica "può richiedere uno sforzo aggiuntivo, può aumentare la complessità delle decisioni di progettazione locale e aumentare il numero di elementi a cui prestare attenzione nel completamento dell'attività. Inoltre, gli architetti sostengono che non è necessario disporre di una struttura così laboriosa per un problema di progettazione, poiché ciò

rende solo l'attività di progettazione più complicata; è come usare una specie di mazza per rompere un dado, non è necessaria tutta quella potenza per progettare.

Questa complessità porta anche a un altro problema: la richiesta di computer più potenti. Questa esigenza è particolarmente evidente nelle aziende piccole e modeste, poiché la maggior parte dei pacchetti parametrici sono di dimensioni piuttosto voluminose. Inoltre, a causa della capacità limitata dei sistemi stand-alone, gli architetti hanno bisogno di condividere le proprie informazioni tra loro attraverso una rete interna che, ancora una volta, dipende fortemente dai computer utilizzati nel processo di progettazione per supportare queste caratteristiche. Gli architetti richiedono quindi sistemi informatici aggiornati che siano in grado di soddisfare le loro attuali esigenze; nonostante ciò, alcuni dei problemi non possono nemmeno essere risolti disponendo di sistemi più avanzati, perché emergono da un difetto strutturale, come il problema della terminazione dell'algoritmo. I modellatori parametrici (come Grasshopper) non consentono i loop nei loro modelli e questo forse mostra ancora una volta i punti deboli nel corpo degli attuali pacchetti parametrici.

- Il problema della paternità

Il più delle volte gli architetti sono definiti gli autori di un progetto e, in un processo di progettazione parametrica, alcuni parametri sono variabili per definizione. Questa variabilità può essere automatizzata e controllata dal sistema; ad esempio, un programma potrebbe essere incaricato di generare un numero qualsiasi di variazioni, in modo casuale o in funzione di alcuni fattori esterni. "Alcuni parametri possono essere scelti, a un certo punto, da qualcuno diverso dall'autore originale, e possibilmente senza il suo consenso". Questo problema culmina nell'approccio algoritmico, in cui lo scripting è il segno distintivo del design, e può sorgere confusione sulla proprietà delle forme algoritmiche e sulla questione della proprietà intellettuale. Anche in questo caso, il problema si verifica quando l'autore della sceneggiatura originale potrebbe non essere l'unico autore del prodotto finale e, di conseguenza, potrebbe non determinare tutte le sue caratteristiche finali. L'unica differenza tra l'aspetto parametrico e algoritmico in questo senso è che nell'approccio algoritmico una "decisione progettuale può essere presa da un processo algoritmico non inteso dal progettista". Tuttavia, si potrebbe dire che la paternità è una preoccupazione seria che dovrebbe essere aggiunta tra le tante sfide della progettazione parametrica.

- Vincolare la creatività con una struttura reattiva

Uno dei confini netti tra CAD tradizionale e progettazione parametrica è l'idea di stabilire vincoli nella progettazione; anche se questo può essere assunto come un vantaggio della progettazione parametrica, allo stesso tempo può svolgere un ruolo negativo. Gli architetti sostengono che i vincoli a volte limitano la creatività dei designer. Inoltre, a causa della struttura reattiva sviluppata dagli ingegneri del software, gli architetti si ritrovano di nuovo vincolati da una preimpostazione di parametri. Sebbene alcuni di questi parametri siano flessibili, offrono comunque condizioni limitate poiché, l'ideale si ottiene quando il progettista è in grado di avere una conversazione interattiva con il programma parametrico.

- Difficoltà di apprendimento e formazione

Sottolineando la complessità dei pacchetti parametrici, risiede un'ulteriore questione legata al problema dell'istruzione e della formazione; sebbene ci siano grandi vantaggi nell'utilizzo di programmi parametrici, la loro integrazione in una pratica è una sfida piuttosto grande. I professionisti dovrebbero imparare come usarli correttamente e come comprenderne la gestione complessiva. Con molta probabilità, questo è il motivo per cui il numero di coloro che conoscono il software parametrico è notevolmente inferiore rispetto agli altri professionisti in un gruppo di progettazione.

Secondo Robert Woodbury, la padronanza dei parametri richiede che gli architetti siano in parte designer, in parte informatici e in parte matematici, cosa più comunemente vista tra i giovani designer; egli si riferisce a sei abilità, tutte richieste per la padronanza parametrica: concepire il flusso di dati, strategie divide et impera, nominare, pensare con astrazione, pensare matematicamente e pensare algoritmicamente. Sebbene la pedagogia parametrica sembri una trappola che forse può essere gradualmente risolta nel tempo, è attualmente una sfida per molti architetti praticanti e questo è il motivo per cui in molti progetti, in particolare edifici residenziali, è spesso preferito l'utilizzo del CAD tradizionale; l'uso del design parametrico è forse visto di più nei progetti culturali, educativi e commerciali, o in quegli edifici che vengono presentati per la prima volta a un concorso di architettura.

### 2.4.3. Le distinzioni con metodi tradizionali CAD e BIM

La sezione finale di questo capitolo cerca di chiarire le distinzioni tra pacchetti parametrici e CAD tradizionale e Building Information Modeling (BIM).

Prima di passare a quanto dicono gli architetti su questo tema di esplorazione, vale la pena confrontare queste due categorie (progettazione parametrica e CAD tradizionale) in termini di

aspetti che vengono solitamente discussi come principali differenziazioni. La Figura 20 illustra un confronto statistico basato sui seguenti sette elementi: migliorare il flusso di lavoro nel processo di progettazione, offrire un approccio più rigoroso, offrire più alternative, facilitare soluzioni impreviste, richiedere meno tempo per l'esplorazione all'interno del processo di progettazione, offrire un livello di controllo più elevato in produzione e apportare vantaggi competitivi.

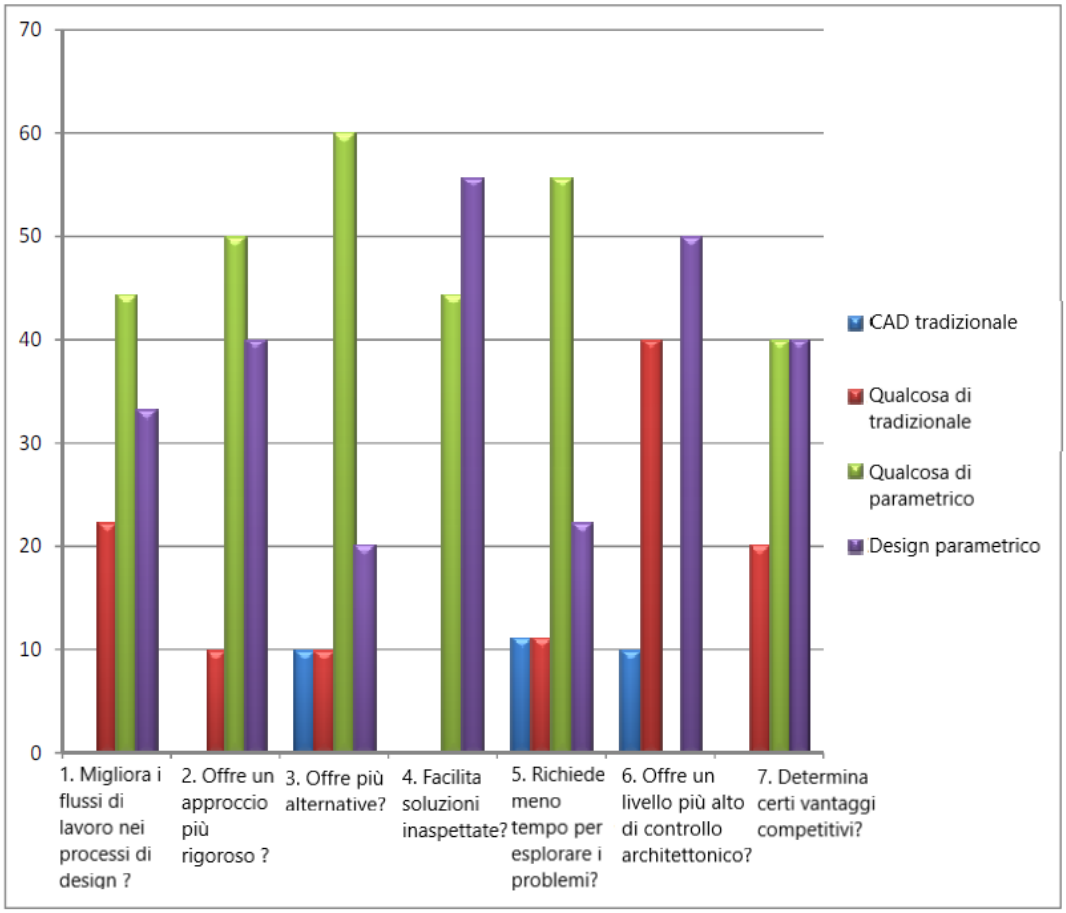



*Figura 20 - Comparazione statistica tra design parametrico e CAD tradizionale*

Agli intervistati è stato chiesto di selezionare una delle opzioni, coprendo una gamma che va dalla progettazione parametrica al CAD tradizionale; in termini di categorie a cui si riferiscono queste sette domande, le domande possono essere suddivise in tre classi: le domande da una a cinque riflettono le preoccupazioni sul processo di progettazione, la domanda sei richiede informazioni sulla produzione e la produzione e infine, la settima domanda chiede se ci sono vantaggi competitivi nella progettazione parametrica rispetto al CAD.

Come mostra il grafico, in quasi tutte le domande, gli intervistati vedono la superiorità del design parametrico rispetto al CAD tradizionale. In termini di facilitare soluzioni inaspettate

(domanda quattro), il design parametrico ha un ruolo molto distintivo e tutti gli architetti intervistati lo hanno riconosciuto chiaramente; nella domanda sei, tuttavia, nonostante la convinzione comune che la progettazione parametrica offra livelli più elevati di controllo architettonico, gli intervistati hanno attribuito più o meno lo stesso livello di importanza sia alla progettazione parametrica che al CAD tradizionale. Tuttavia, a parte questa domanda, il grafico mostra chiaramente che gli architetti sono in grado di distinguere tra CAD tradizionale e progettazione parametrica. Tuttavia, per esprimere le distinzioni in modo più specifico, verranno qui discussi alcuni aspetti assiali facendo riferimento sia a fonti secondarie sia alle dichiarazioni degli intervistati.

In primo luogo, dovrebbe essere argomentato il concetto di impostazione dei vincoli, che è la distinzione intrinseca dei parametrici dal CAD. In passato, gli architetti dovevano pensare ai parametri nella loro mente e poi trasferirli sullo schermo del computer. Tuttavia, la maggior parte di questi parametri è ora impostata all'interno del pacchetto e, di conseguenza, gli architetti non devono concentrarsi su di essi in anticipo.

A questo si aggiunge il fatto che, i pacchetti CAD tradizionali non avevano la capacità di supportare le relazioni tra gli elementi del design; i programmi parametrici invece consentono al progettista di applicare catene di dipendenza e associatività agli oggetti di progettazione ogniqualvolta richiesto. Inoltre, forniscono un livello superiore di gestione di tutti i disegni di un progetto, come piante, sezioni e prospetti, mentre nel CAD tradizionale tutto era separato. Concentrandosi sulla proprietà associativa, Woodbury offre una spiegazione chiara affermando che in precedenza, negli strumenti di progettazione convenzionali, la creazione di un modello era facile, ma apportare modifiche era difficile e noioso. Quindi, gli strumenti potrebbero ostacolare l'esplorazione del design; anche l'eliminazione di una parte diventa semplice, poiché le parti erano indipendenti. In confronto, la progettazione parametrica richiede che il progettista si concentri sulla logica che lega insieme i componenti del progetto. Sebbene ciò possa richiedere tempo, il risultato è che il sistema si occupa di mantenere il design coerente con le relazioni e quindi aumenta la capacità del progettista di esplorare le idee riducendo la noia delle rielaborazioni.

Una caratteristica negativa della progettazione parametrica rispetto al CAD consiste nel trasformare la posizione di un architetto in un gestore di fogli di calcolo. Questo cambiamento non è realmente compreso dal resto del settore o dai clienti nel loro insieme; dopo aver esaminato una serie di studi di architettura, si potrebbe affermare che nella pratica odierna sono attribuiti ai progettisti parametrici compiti diversi che non esistevano quando si lavorava con

strumenti CAD tradizionali solo dieci anni fa. Pertanto, gli architetti si vedono con un'identità trasformata, che consecutivamente porta a questioni complesse.

Infine, la progettazione parametrica è più vicina alla realtà di un progetto e dalla sua costruzione; tornando alle prime versioni di pacchetti CAD, la maggior parte degli architetti considerava quella generazione di software come un'estensione della tecnica di disegno su un pezzo di programma; ora, i programmi paramedici offrono un modo diverso di concettualizzare, che in sostanza è più vicino alla realtà delle costruzioni. Le Impostazioni di vincoli geometrici e fisici all'interno della progettazione, nonché la considerazione di questioni come i parametri di costo, aiuta gli architetti ad avvicinarsi sempre più a ciò che un progetto necessita per il suo design.

Di estremo interesse risulta anche il confronto tra il design parametrico e la metodologia BIM.

La maggior parte degli architetti pensa infatti che il design parametrico e il BIM siano gli stessi, o almeno abbaiano le stesse origini. Sebbene entrambi lavorano con i parametri, esistono due fattori principali che li separano l'uno dall'altro.

Il primo elemento è relativo alla capacità di scripting che offrono tutti i programmi parametrici, che consente all'architetto di avere una sorta di dialogo con il programma del computer. Simile al CAD, la struttura BIM è rigida e talvolta troppo elaborata per pensare alla forma architettonica; non consente agli architetti di godere del tipo di libertà che viene loro concessa lavorando con i pacchetti parametrici. I pacchetti BIM non consentono quindi la funzionalità di scripting, che ancora una volta limita i progettisti nella scelta dei comandi procedurali standard.

Sottolineando approcci innovativi alla progettazione, la maggior parte dei pacchetti parametrici sono stati sviluppati come risultato di aspirazioni verso nuove forme, mentre il BIM si occupa più di facilitare una progettazione che risulti il più efficiente possibile. Il BIM è un ottimo strumento dal punto di vista commerciale, perché consente agli architetti di condividere e modellare l'intero progetto in cui tutto viene aggiornato quando i progettisti cambiano qualcosa, che in questo modo è abbastanza simile a programmi parametrici come Grasshopper. Tuttavia, programmi come Grasshopper vengono normalmente utilizzati per la ricerca di moduli o la generazione di moduli piuttosto che creare un database per gli elementi di progettazione.

In un certo senso, il BIM è uno "strumento di costruzione" piuttosto che uno "strumento di architettura" e offre la possibilità di riunire tutti gli stakeholder di un progetto per facilitare la collaborazione durante tutto il processo di progettazione. In altre parole, il BIM funge da

collegamento tra i diversi attori, mentre i pacchetti parametrici riguardano più come facilitare il processo di acquisizione della forma.

Per concludere, in confronto ai programmi parametrici, il BIM è piuttosto giovane e forse questo è il motivo per cui può sembrare misterioso a coloro che non sono strettamente legati al mondo dei programmi per computer. Tuttavia, sembra che sia più comprensibile per quegli architetti che hanno una conoscenza di base in CAD, per i quali i programmi parametrici sembrano estremamente peculiari e problematici da imparare e con cui lavorare.

## 2.5. Il Computational design nei processi di progettazione

Oggi, all'interno dell'architettura, gli strumenti digitali, dall'apprendimento automatico alle tecnologie di fabbricazione, dall'intelligenza artificiale ai Big Data, stanno diventando sempre più onnipresenti e pervasivi all'interno della nostra quotidianità. Il crescente interesse per l'impatto che queste tecnologie negli ultimi anni stanno avendo, ha rapidamente ampliato l'uso di questi strumenti soprattutto all'interno del processo di progettazione al punto di parlare del design computazionale come uno dei principali strumenti alla base dei processi di progettazione, sempre più in crescita e con un'impareggiabile capacità di controllo delle informazioni relazionate ai modelli 3D.

Il design computazionale, come dice la parola stessa, è basato sull'uso di strumenti informatici all'interno del processo di design e offre al progettista la possibilità di manipolare idee, concetti e processi tramite computer, nonché di controllare la conversione delle proprietà di oggetti e inputi in soluzioni e alternative differenti con l'obiettivo di ottimizzare i requisiti funzionali del problema di design.

Esso permette quindi di applicare le strategie computazionali ad un processo compositivo e può essere inteso, in prima battuta, come il prodotto dell'unione di due discipline: il design e la computazione; una simbiosi capace di unire quei i fattori creativi, funzionali, simbolici e produttivi che portano alla realizzazione di un oggetto di design, con gli algoritmi alla base della computazione, che traducono la complessità della realtà in un susseguirsi di elementi semplici e li trasforma in dati da poter rielaborare sottoforma di algoritmi capaci di gestire e risolvere specifici problemi.

La grande potenzialità del design computazionale sta nella possibilità di concatenare una serie di dati e rendere automatiche una serie di modifiche che altrimenti richiederebbero tempo, energie e denaro. Questa rapidità di operare permette infatti di studiare simultaneamente diverse soluzioni per un'unica richiesta**;** l'architettura è quindi il campo ideale dove esprimere tutto il potenziale di questi strumenti computazionali.

Una volta definito progettualmente l'oggetto, come ad esempio una copertura di un padiglione, se ne scrive, prima di tutto, la definizione parametrica precisando parametri dimensionali, quantitativi, materici, di gestione delle forme ecc. Agendo su di essi, si generano poi forme e idee anche molte diverse tra loro; in tal modo si ottengono spesso volumetrie e geometrie che,

con i comuni strumenti CAD, sarebbe quasi impossibile, o comunque molto dispendioso dal punto di vista di tempo ed energie, rappresentare in 2D o 3D.

Il ruolo di un ingegnere è quello di realizzare l'ambizione dei suoi clienti utilizzando tecnologie intelligenti capaci di soddisfare la crescente domanda in modo più efficiente e offrendo degli standard più elevati. All'interno di un generico processo di progettazione, i fattori chiavi dell'utilizzo di strumenti digitali sono:

- La necessità di un processo di progettazione efficiente in termini costo/ tempo;
- La disponibilità della nuova tecnologia;
- Le crescenti richieste di prestazioni del risultato del progetto

Di conseguenza, è semplice intuire come l'uso dei metodi basati sul calcolo non solo domina il modo in cui si progetta, si costruiscono edifici e le loro componenti, ma determina anche il modo in cui si collabora, si scambiano informazioni e si organizza il processo di costruzione stesso.

### 2.5.1. La necessità di processi di design efficienti in termini di costi/tempi

L'introduzione del *Building Information Modeling (BIM)* e dei software di progettazione parametrica nel settore delle costruzioni ha stimolato lo scambio interdisciplinare di informazioni di progettazione influenzando, già a partire dalla fine degli anni '90, il processo di progettazione architettonica in modo integrato. Sulla base di obiettivi e vincoli predefiniti, grazie a tali strumenti è oggi possibile trovare modifiche ottimali al progetto e rendere evidenti le conseguenze di tali modifiche evitando costosi errori, o peggio, rielaborazioni.

L'automazione dei flussi di lavoro è considerata estremamente efficace in quanto permette di ridurre al minimo i tempi (e quindi i costi) degli ingegneri e allo stesso tempo di ridurre al minimo il rischio di errori del progettista. L'approccio algoritmico o parametrico dei flussi di lavoro computazionali (automatizzazione), tuttavia richiede una nuova mentalità nel campo della progettazione; tipicamente quando si lavora con algoritmi, è necessario eseguire passaggi sequenziali affinché l'approccio sia efficace. Prima di tutto, bisogna pensare a cosa si vuole ottenere e quali sono le condizioni al contorno per il risultato desiderato; in secondo luogo, è necessario andare a definire le regole e quindi impostare il flusso di lavoro logico individuando quali variabili devono avere influenza sul progetto e razionalizzando le associazioni tra questi parametri. Infine, è necessario "insegnare" questo flusso di lavoro al computer mediante programmazione (codice) e premere invio. L'uso di algoritmi all'interno della pratica

ingegneristica aumenta notevolmente l'efficienza del lavoro dei progettisti automatizzando le attività ripetitive, come ad esempio il calcolo dei rinforzi dei pilastri; un algoritmo intelligente è infatti capace di generare una grande varietà di layout di rinforzo (che potrebbe essere desiderabile dal punto di vista della riduzione del materiale da usare), utilizzando la stessa logica programmata. Questo approccio di progettazione parametrica consente una generazione più efficiente di soluzioni alternative e offre l'opportunità di rispondere alle variazioni di progettazione lungo il percorso; fintanto che l'obiettivo e la logica non cambiano (e di conseguenza non infrangono le regole implementate nel codice) l'algoritmo creato è in grado di supportare il progetto nel tempo.

### 2.5.2. La disponibilità di nuove tecnologie

La nuova tecnologia consente di superare i limiti precedenti e quindi di migliorare la gestibilità del progettista, inspirando gli ingegneri e gli architetti a trovare soluzioni precedentemente impensabili.

La storia insegna che spesso, da un punto di vista ingegneristico, sono necessarie nuove tecnologie per risolvere un problema tecnico specifico, dopodiché i progettisti fanno un uso riconoscente delle conoscenze e delle esperienze acquisite per elaborare nuove applicazioni per questa specifica tecnologia. Per esempio, le descrizioni matematiche (algoritmi) che consentono agli ingegneri di Renault e Citroen di definire le forme curve dei loro nuovi progetti di automobili, hanno portato alla fine all'uso diffuso di tali funzioni parametriche negli strumenti CAD (*computer-aided-design*) che hanno inspirato architetti come Frank O'Gehry, Zaha Hadid o molti altri a progettare i loro edifici a forma libera.

Per i designer di oggi, i progetti geometricamente complessi, ai quali gli ingegneri del passato dovevano dedicare gran parte della loro vita, sono ora considerati a portata di mano grazie a dei semplici clicks. Strumenti di progettazione computazionale avanzati come *Rhinoceros* e *Autodesk3D*, consentono oggi di progettare strutture tridimensionali di forma complessa che erano considerate troppo difficili o dispendiose in termini di tempo per comunicare o modificare con tradizionali disegni bidimensionali o modelli su piccola scala.

La disponibilità di una potenza di calcolo computazionale sufficiente e di un software di analisi avanzato, come l'analisi agli elementi finiti (FEA), ci consente di utilizzare la simulazione digitale per ottenere anche informazioni sul comportamento strutturale di qualsiasi struttura.

Rimanendo nell'ambito dell'ingegneria civile, ad esempio, per il progetto di ristrutturazione di un edificio della facoltà *Delf University of Technology*, è stata realizzata una struttura a traliccio, che si estende tra le aule a sbalzo, utilizzando specifici algoritmi di ottimizzazione strutturale capaci di destinare il materiale solo ai luoghi in cui esso è più efficace.

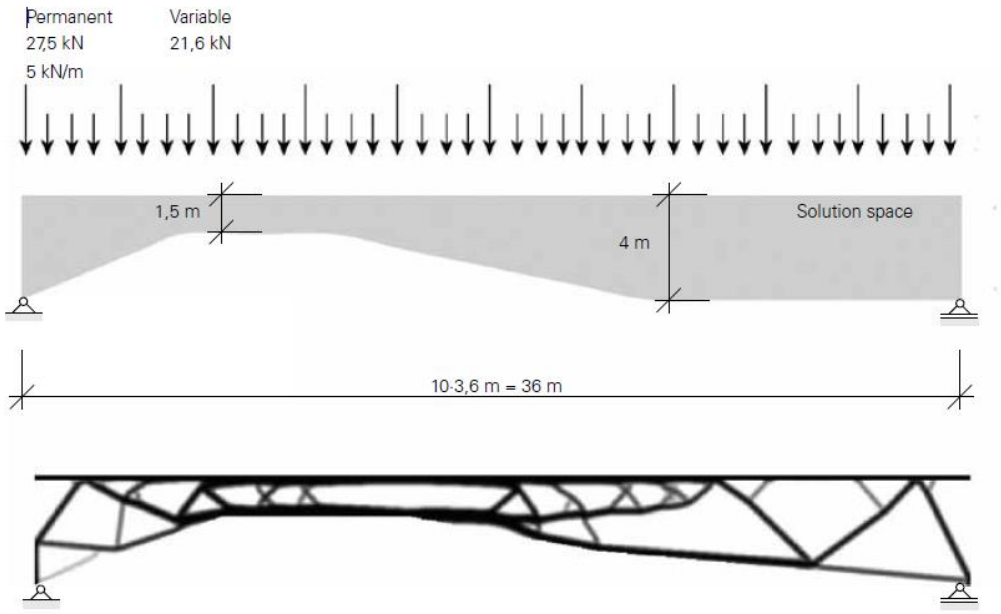



*Figura 21 – Generazione del layout di una struttura a traliccio utilizzando specifici algoritmi*

Anche le crescenti richieste di prestazioni degli edifici e degli ambienti costruiti, ad esempio ambizioni di sostenibilità, richiedono un processo integrato per raggiungerle. Di conseguenza, la complessità dei flussi di lavoro di progettazione aumenta in modo significativo e crea la necessità di simulazioni avanzate per ottenere informazioni sull'interdipendenza di numerose variabili di input e quindi sulle conseguenze delle alterazioni del progetto sul risultato complessivo. Rispetto ai processi di progettazione tradizionali, un processo di progettazione integrato richiede di abbinare gli obiettivi e le condizioni al contorno l'uno con l'altro nella fase iniziale di progettazione.

Da un punto di vista ingegneristico, come già detto, gli ingegneri cercano di determinare soluzioni progettuali che riducano al minimo la quantità di materiale richiesto utilizzando le loro conoscenze, esperienze e abilità alle risorse che vengono fornite; dal punto di vista dell'efficienza però, i metodi tradizionali sono a volte preferiti rispetto ai metodi di calcolo più avanzati (come le analisi FEM) per limitare i tempi stessi di analisi, il tutto compromettendo l'utilizzo ottimale del materiale. Per situazioni relativamente semplici, gli ingegneri sono molto capaci di prevedere la qualità di un'alternativa conoscendo e comprendendo le relazioni o le

equazioni sottostanti; per quanto concerne le situazioni più complesse, sebbene le conoscenza ingegneristica disponibile dovrebbe risultare ancora sufficiente per fornire una visuale completa del comportamento del sistema strutturale, è probabile che le interdipendenze quantitative delle variabili (o parametri) di progettazione siano difficili da comprendere appieno. Questa mancanza di comprensione può spesso portare a scelte di progettazione meno efficienti in cui le alternative potenzialmente più convenienti o più sostenibili potrebbero rimanere inesplorate. Un modo per compensare questa mancanza prevede di aumentare il numero di alternative valutate attraverso l'uso di metodi computazionali, senza richiedere grandi sforzi ingegneristici o tempi significativi per ciascuna alternativa valutata; le variazioni intenzionali dei parametri possono quindi aiutare a comprendere le influenze di ciascun parametro sulle prestazioni complessive di aspetti quali la geometria, comportamento strutturale, costi, costruibilità e molto altro. In altre parole, il concetto diventa: "imparare dalla simulazione".

### 2.5.3. L'impatto della progettazione computazionale sul ruolo degli ingegneri

In questa epoca, la tecnologia continuerà a svilupparsi ad un ritmo sempre più veloce e con essa, la disponibilità della tecnologia stessa da utilizzare nella pratica di progettazione quotidiana; tuttavia, ciò che realmente fa la differenza è la capacità degni ingeneri e dei progettisti di apprendere, adattare, regolare e implementare questi nuovi strumenti nei processi di design.

Le metodologie digitali sono spesso introdotte dagli ingegneri per migliorare l'efficienza del loro lavoro anche se spesso la scelta di applicare strategie di progettazione computazionale nei processi di design non viene fatta all'inizio di un progetto o nelle prime fasi del design. Questo deriva tipicamente dalla mancanza di familiarità con le nuove tecnologie digitali dovute al fatto che la maggior parte dei project leader e dei senior designer non sono una generazione che possiede una buona familiarità con questi nuovi strumenti digitali. D'altra parte, rispetto a un approccio tradizionale "lineare", un approccio di progettazione digitale o parametrico in genere non fornisce lo stesso risultato atteso nelle fasi iniziali del processo il ché può essere anche percepito come un rischio sia in termini di investimento sia di tempo di progettazione (perso).

Bisogna però tenere presente che, effettuare un calcolo o un disegno di dettaglio con mezzi tradizionali, permette di creare solo un insieme limitato di istruzioni capaci di produrre gli stessi risultati a livello di qualità attesa. Con le tecnologie digitali, si aggiunge la capacità di variazione, una flessibilità al processo che consente una semplice gestione delle modifiche del

progetto, una valutazione automatizzata di soluzioni comparabili capace di rendere possibile delle variazioni per scopi di simulazione. Il tutto si traduce in un aumento dell'efficienza che può determinare un aumento del valore aggiunto per il progetto (Figura 20).

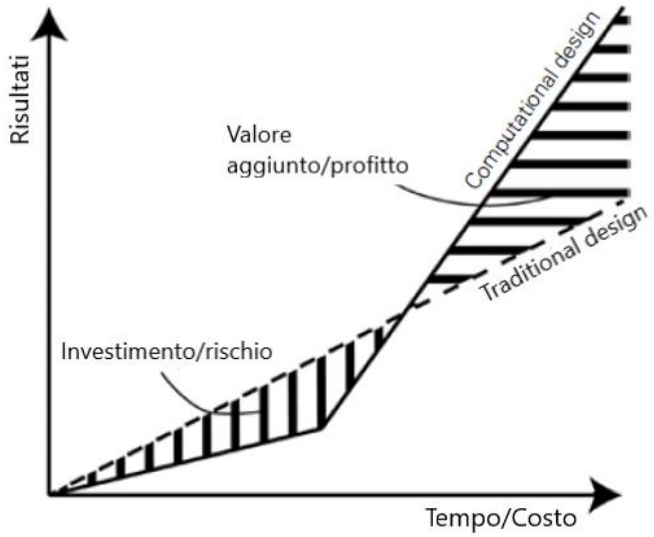



*Figura 22 - Comparazione tra l'approccio di progettazione computazionale e l'approccio tradizionale in termini di progressi e risultati relativi al tempo/costo*

Lo sviluppo delle tecnologie computazionali può essere in ogni caso considerato il prossimo passo logico nella cosiddetta rivoluzione digitale del settore architettonico; il modo in cui le informazioni vengono scambiate tra le parti si è trasformato infatti dallo scambio di disegni bidimensionali allo scambio di informazioni (ad esempio attraverso modelli digitali tridimensionali) e allo scambio di logica (ad esempio attraverso algoritmi digitali).

La tecnologia che ci circonda sta diventando sempre più "intelligente" con lo scopo di aumentare la sicurezza riducendo al minimo le interferenze, o gli errori, umani. E se i computer diventassero sempre più intelligenti e come tali iniziassero a progettare edifici completi? Cosa significherà questo per il futuro di architetti, ingegneri, imprenditori? Chi può essere definito il responsabile del design? Le domande che l'avvento del *Computational design* porta con sé sono tante e la sua diffusione in architettura ha riportato lo stesso interrogativo che si presentò con l'avvento del CAD (*Computer Aided Design*): l'evoluzione, minerà il ruolo dell'architetto? Il progettista verrà "sostituito" dal programma? La risposta, oggi come allora, è: assolutamente no! Questo perché il programma rappresenta uno strumento a supporto del progettista che resta sempre la "mente", colui che decide e fissa il risultato da ottenere. Il buon utilizzo del

programma, infatti, si basa sul presupposto che l'utilizzatore sappia inserire correttamente dati e parametri in modo da gestire il software e non sia invece "gestito" o vincolato da esso.

Aldilà di tutte le preoccupazioni, il design computazionale può essere visto come potente mezzo di cambiamento, un'opportunità che consente di dedicare del tempo a ciò che veramente è l'ingegneria: soddisfare le ambizioni dei clienti in modo economico e sostenibile utilizzando la tecnologia intelligente.

Il primo contatto con il mondo della modellazione parametrica può risultare sicuramente destabilizzante; il soggetto che tenta l'approccio con questo tipo di disciplina si trova infatti di fronte alla necessità di dover ampliare la propria concezione del processo creativo. E' necessario quindi non partire da geometrie reali bensì dalla logica che si cela dietro di esse; distaccarsi dalla forma immediata puntando piuttosto alla selezione di specifici parametri che, se correttamente utilizzati, rappresentano un enorme potenzialità di supposto al progettista permettendogli di spaziare diverse soluzioni fino ad arrivare a quella che meglio soddisfa le esigenze del designer.

Il design computazionale è quindi uno dei più potenti mezzi oggi a disposizione dei designers; rappresenta il presente e, allo stesso tempo, il futuro dell'architettura e dei processi compositivi che portano alla realizzazione di un progetto capace, grazie alle infinite possibilità generate dalle modifiche dei dati parametrici, di superare limiti geometrici e ottenere forme e soluzioni uniche.

# 3. Software e algoritmi parametrici

## 3.1. Panoramica letteraria

Analizzati alcuni degli aspetti principali della pratica progettuale digitale è possibile adesso discutere degli strumenti che negli ultimi anni hanno irrotto nella scena dei programmi informatici, i software parametrici. Si tratta di applicazioni che combinano le potenzialità di strumenti utili al disegno e alla rappresentazione ad altri finalizzati alla programmazione.

In particolare, i software parametrici sono dei casi particolari di modellatori costituiti da comandi in grado di consentire all'utente di modellare le geometrie volute, operazione già tipica dei software CAD dedicati alla modellazione 3D; oltre a questa funzione, questo tipo di programmi integra però una nuova possibilità tipica del mondo della programmazione: la scrittura di codici.

Le ultime frontiere delle esplorazioni compositive architettoniche, si sono spinte verso nuove famiglie di software dotati di un'interfaccia grafica tale per cui, lavorare sui codici non è più un lavoro per soli esperti di programmazione; grazie all'adattamento delle modalità di scrittura del linguaggio macchina in un'interfaccia semplificata, il programma permette ai progettisti di ibridare il proprio operato integrandolo con operazioni usualmente tipiche dei programmatori. Lavorando coì, direttamente sui codici che gestiscono le geometrie rappresentate, il progettista ha la possibilità di lavorare più che sull'oggetto finito, sulle relazioni che ne definiscono la forma, tramite la definizione di parametri, da cui il termine software di progettazione parametrica.

La cosa interessante di questo tipo di modelli è che se variano le condizioni di input, e cioè i parametri, le geometrie si "aggiustano" automaticamente, senza bisogno di nessun intervento da parte del disegnatore, se non quello di variare i parametri stessi.

Possiamo, in sintesi, definire questa tipologia di software dei programmi ibridi, in grado di gestire tanto la parte inerente alla rappresentazione, quanto quella relativa alla programmazione. La commistione di queste due modalità di lavoro, entrambe inerenti alla progettazione, pone il designer in un'ottica nella quale il risultato del proprio lavoro non è più univoco ma del tutto modificabile subordinatamente alla modifica di una serie di parametri; in questa maniera sono le relazioni geometriche stabilite tra i diversi parametri che diventano univoche.

L'attenzione si sposta dalla rappresentazione dell'oggetto, che diventa modalità di verifica intermedia o semplice output del processo, alla scrittura del codice che lo genera in cui il designer si deve soprattutto focalizzare sulla definizione delle relazioni geometriche che permettono la modificabilità dell'oggetto finito.

All'operazione di scelta delle azioni che il codice permetterà di eseguire, segue la fase in cui il progettista deve porle in relazione traducendole in linguaggio macchina tramite l'interfaccia grafica; durante il processo di scrittura l'operatore avrà inoltre la possibilità di confrontare, in tempo reale, i risultati volumetrici del codice che sta scrivendo, tramite la funzionalità rappresentativa del software. Sta poi alle abilità tecnico-matematiche del progettista riuscire a usare l'interfaccia per creare un codice che presenti un linguaggio di scrittura corretto e funzionante, nonché completo delle operazioni necessarie a raggiungere i range di parametri scegliendo l'input in base agi obiettivi peculiari di progetto (destinazione, funzioni, esigenze particolari della committenza) e alle situazioni esistenti (possibilità spaziali, condizioni ambientali, quadro legislativo, ecc.); è infatti il codice che permette, alla combinazione di input che lavorano nell'arco delle operazioni geometriche programmate e integrate tra loro, di andare a definire in maniera automatica il risultato volumetrico del processo.

In conclusione, il vero risultato del lavoro prodotto tramite software parametrici, il cui valore aggiunto è la grande versatilità e automaticità di applicazione, è il codice.

## 3.2. I principali software di design parametrico

Oggi, tramite rete, è possibile avere accesso a un'ampia panoramica di software utilizzati per la programmazione parametrica; infatti, sono molti i programmi e i plugin che è possibile facilmente ottenere per affiancare programmi di progettazione nella generazione e nella gestione di forme innovative e spesso anche molto complesse.

Tra i software più comuni utilizzati dai progettisti, interessati a sviluppare progetti di design parametrico, i principali vengono riportati di seguito:

- Grasshopper, associato a Rhinoceros;
- Dynamo, associato a Revit (Autodesk);
- Digital Project, di Gehry Technologies;
- Generative Components, distribuito da Bentley Systems

Senza entrare troppo nel dettaglio, di seguito vengono brevemente descritte le principali caratteristiche dei software sopra citati con particolare riguardo nei confronti di Grasshopper, uno dei più potenti strumenti di modellazione algoritmica per la generazione e il controllo di forme complesse a qualsiasi scala: dall'architettura al design.

- **Revit**

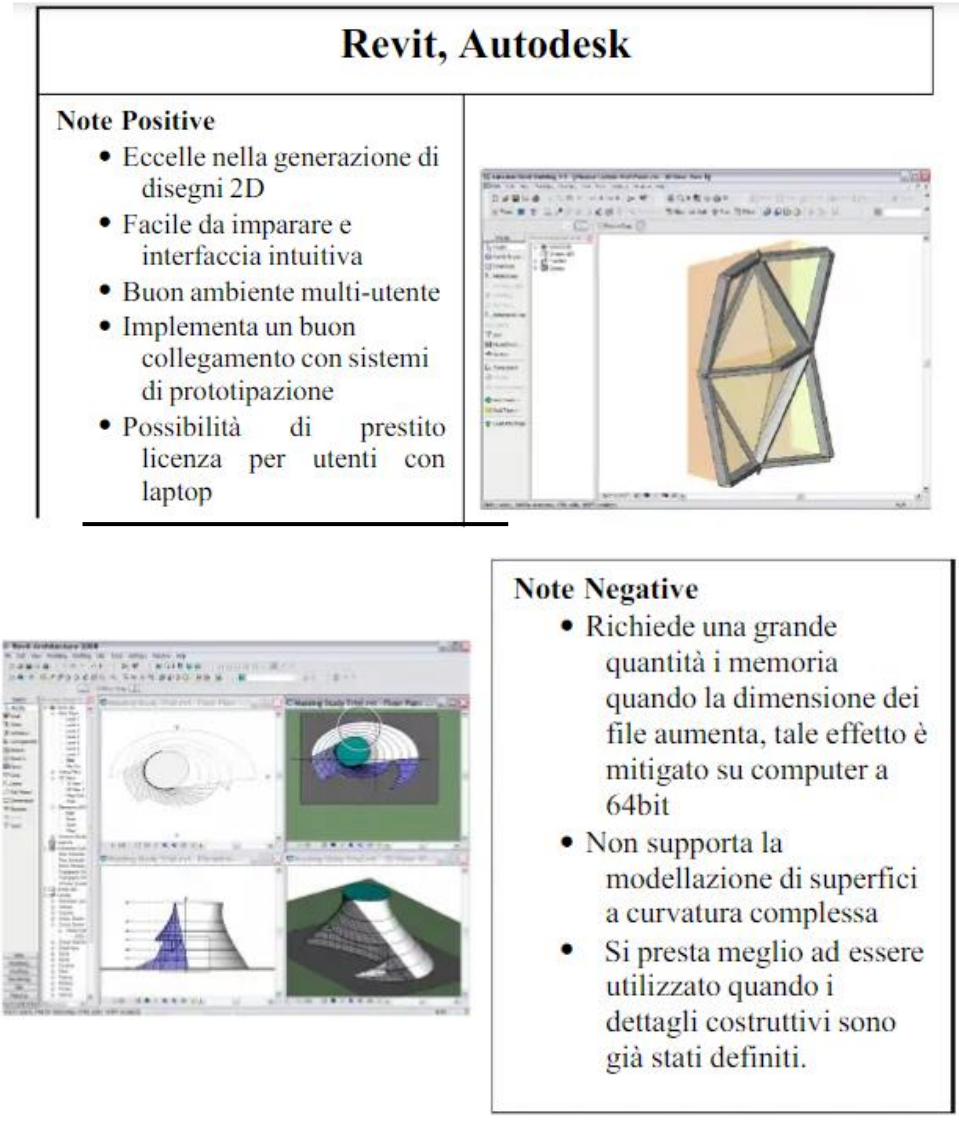



*Figura 23 – Alcune note positive e negative sul software Revit*

Tra i software più utilizzati nel campo, Revit (costruito da Autodesk), rappresenta il software parametrico più diffuso grazie alla sua semplicità d'uso, capace di affiancare, al settore dello

scambio e della gestione dei dati, un controllo rigoroso delle geometrie di progetto che si accompagna a quello della generazione di forme complesse.
Esso offre inoltre una serie di pacchetti (tra cui emerge sicuramente Dynamo) il cui utilizzo si sta espandendo a un gruppo sempre più ampio di persone che forniscono nuovi modi di semplificare la consegna progettuale in riferimento alla funzionalità e alla necessità del cliente e dell'impresa costruttrice. Infatti, come tutti gli strumenti geometrici, Dynamo permette di lavorare all'interno di un processo di programmazione visuale in cui possono essere collegati più elementi insieme, per definire i rapporti e le sequenze di azioni che compongono algoritmi personalizzati al fine di ottimizzare il flusso di lavoro; tali algoritmi possono poi essere utilizzati per una vasta gamma di applicazioni, in tempo reale e senza una riga di codice.

L'immagine 23 illustra, a titolo d'esempio, le principali caratteristiche offerte da Revit, distinguendone vantaggi e gli svantaggi.

- **Digital Project**

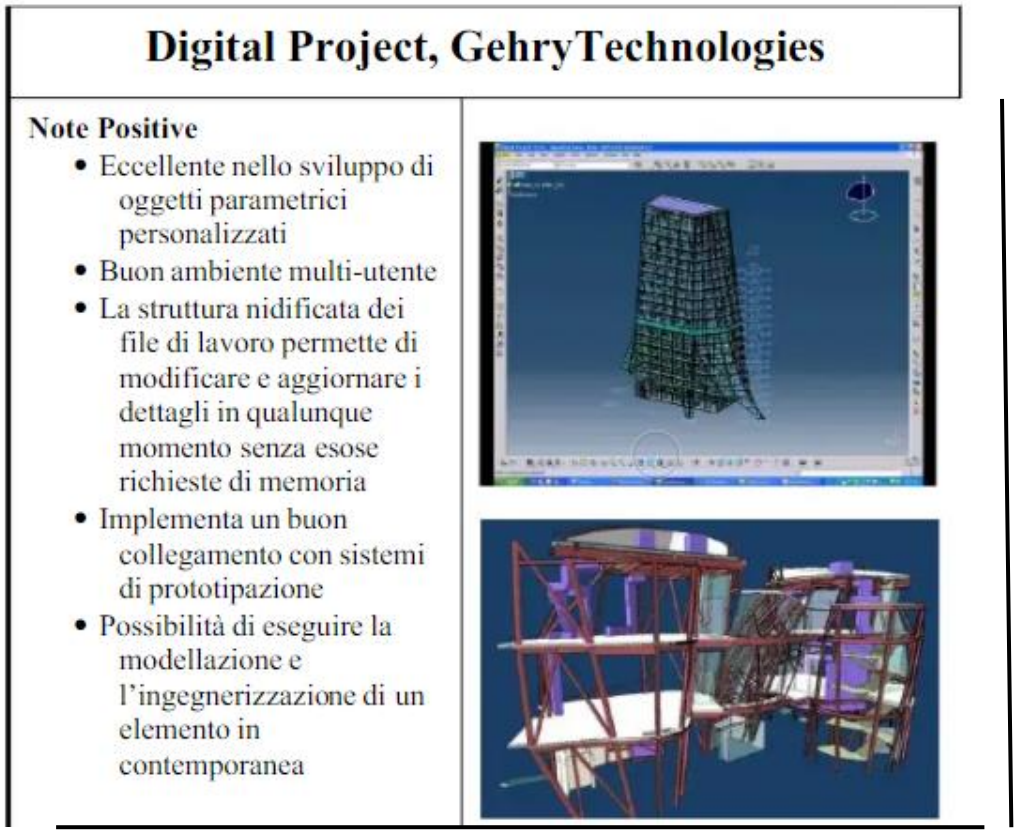



*Figura 24 - I vantaggi offerti dal software Digital Project*

Di seguito si riportano invece gli svantaggi offerti dal software di Gehry Technologies:

- È inferiore ad altri applicativi nella generazione di disegni 2D;
- Più lento di altri software, come il Generative Component, nella creazione di geometrie per punti;
- L'ingente quantità di temo richiesto ne suggerisce un uso più appropriato dopo aver già effettuato una progettazione di massima.

- **Generative Components**

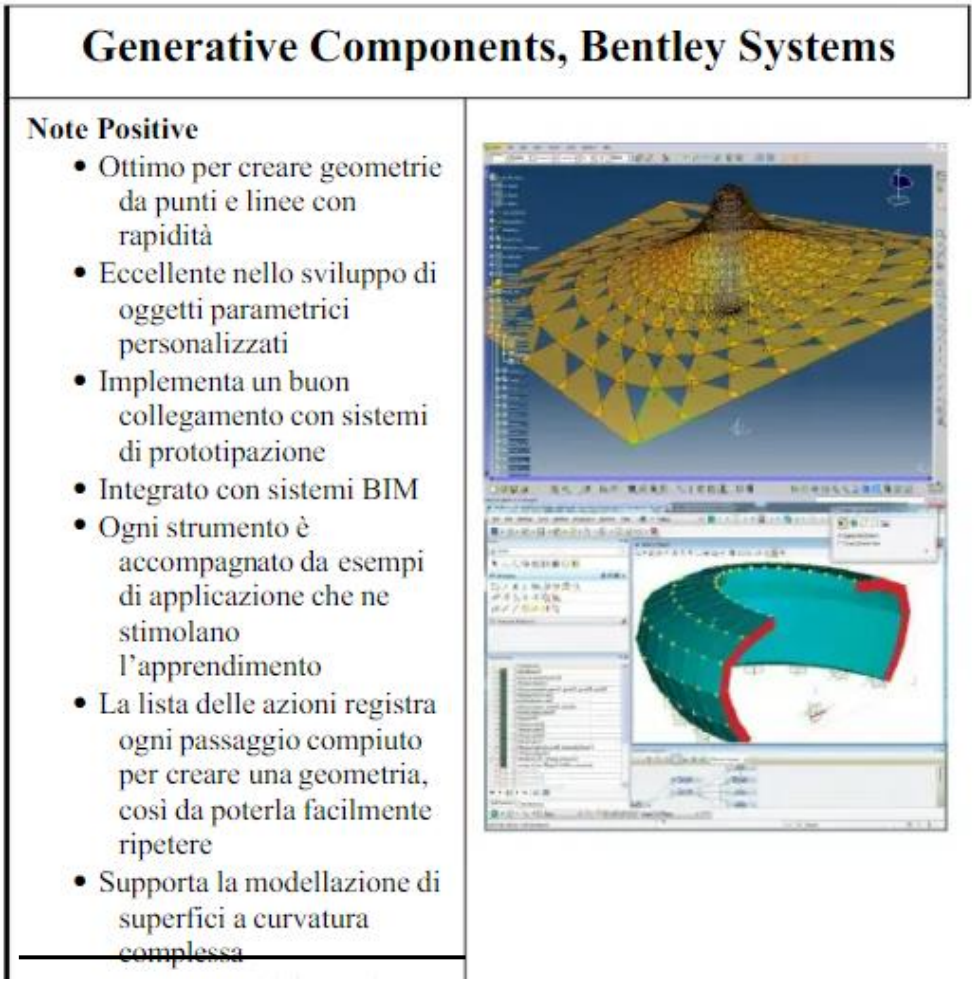



*Figura 25 – I vantaggi offerti del software Generative Components*

Gli svantaggi offerti da questo tipo di software vengono invece elencati di seguito:

- Complesso per i nuovi utenti;
- Non supporta un ambiente multi-utente;
- Non possiede una reale interfaccia visiva;

- Non può produrre modelli da cui ottenere disegni di piante/ prospetti / sezioni, per i quali diventa necessaria un'esportazione ad altri software CAD.

- **Grasshopper**

L'ultimo software trattato, e non per importanza, è il Grasshopper; componente aggiuntivo (*plug-in)* di Rhinoceros, distribuito da McNeel & Associates, nonché uno tra i primi strumenti a proporre la modellazione basata su superfici NURBS (*Non Uniform Rational B-Splines*) che ha permesso la generazione di superfici curve e complesse mediante un controllo matematico ma anche una alta manipolabilità grafica. Grazie a Grassoppher, Rhino ha infatti reso possibile una modellazione "procedurale" ovvero un tipo di modellazione che non costringe a disegnare uno per volta ogni singolo componente di un modello, ma permette di costruire procedure che ordinano al programma di generarle automaticamente.

La modellazione parametrica con Grasshopper richiede di un approccio un po' differente rispetto alla modellazione tradizionale e, più che uno strumento o un applicativo, presenta un modo di pensare la progettazione; appunto parametrico o associativo; grazie alle enormi potenzialità che è in grado di offrire, la razionalizzazione della forma, le scomposizioni e lo sviluppo di superfici complesse in elementi piani, cessano di essere operazioni "a posteriori" ma vengono integrate nel medesimo processo di definizione formale.
Esso è infatti in grado di generare forme tridimensionali complesse attraverso la definizione di un diagramma a nodi che descrive le relazioni tra le parti (logica associativa) di un qualsiasi progetto. I modelli 3D sviluppati con Grasshopper sono sistemi dinamici modificabili in tempo reale mediante la variazione dei parametri definiti durante la costruzione del diagramma, con vantaggi immediati in termini di esplorazione formale e di controllo/razionalizzazione della forma.
Come diretta conseguenza della logica associativa è possibile creare legami concettuali ed effettivi tra i diversi livelli di approfondimento progettuale. In altri termini, la modifica di un parametro a scala più ampia è in grado di generare una propagazione di modifiche tale da giungere alla congruente ridefinizione di dettagli a piccola scala: è possibile ipotizzare un link diretto tra i parametri relativi alla forma generale di una superficie complessa e le caratteristiche geometriche di un nodo strutturale, il tutto guidato da logiche di relazione definite dal designer.
Entrando più nel dettaglio, la modalità introdotta da Grasshopper è infatti definita *scripting visuale* poiché consente di lavorare all'interno dello spazio di modellazione non solo attraverso il tracciamento di punti, curve e polilinee, ma anche, e soprattutto, agendo e generando le loro

regole di interazione fino a permettere la costruzione di forme altamente complesse mantenendone però il controllo. Tali regoli, tra cui ripetizioni, cicli, serie numeriche ed equazioni matematiche di superfici, vengono poi rappresentate in elementi, che a loro volta diventano parte di un grafo nel quale vengono collegati tra loro (una curva, ad esempio, è collegata a un elemento che genera un gruppo di punti e, a sua volta, quest'ultimo è collegato a una serie numerica che ne genera le coordinate).

L'immagine sottostante riporta, così come fatto per gli altri software, i principali vantaggi e svantaggi dello strumento trattato.

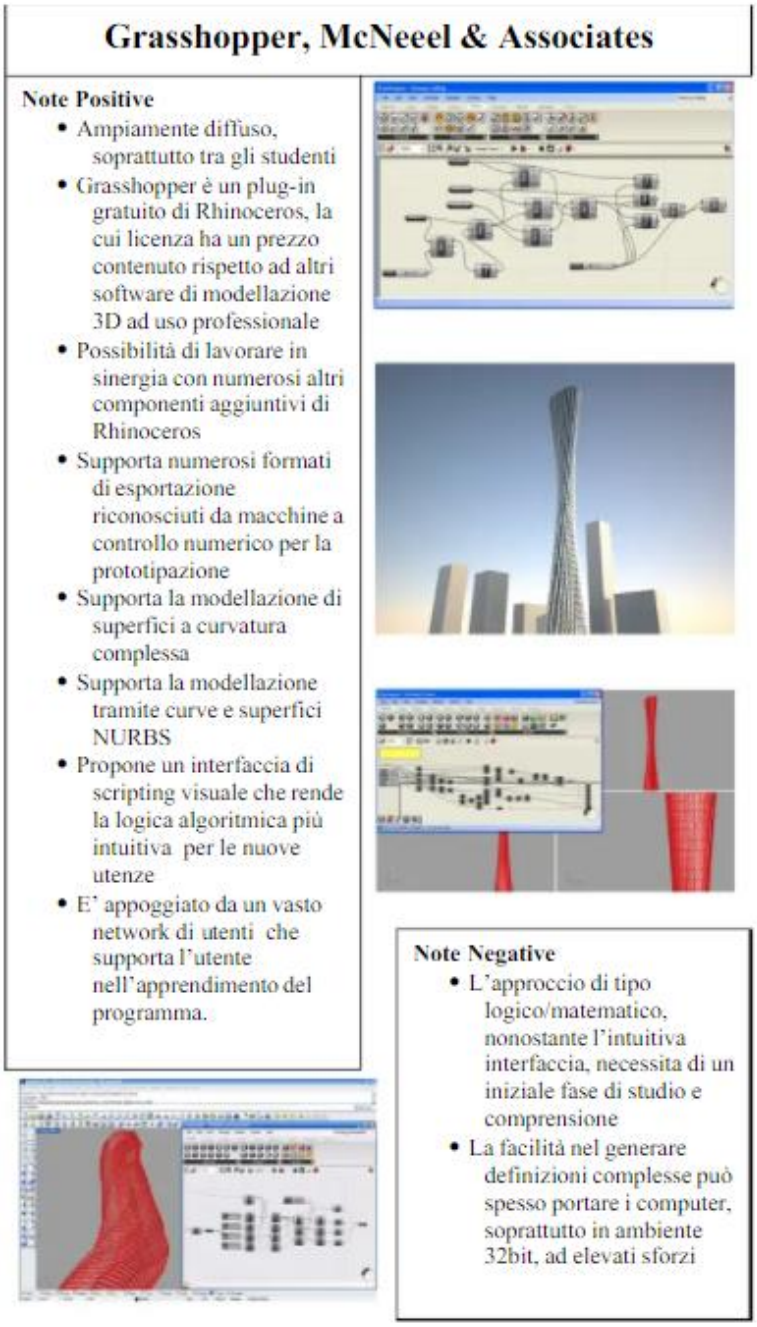



*Figura 26 – Aspetti positivi e negativi del software Grasshopper*

Ai fini dello sviluppo applicativo del lavoro di tesi, si è scelto di utilizzare Grasshopper come strumento per la realizzazione della copertura parametrica; nel paragrafo che segue verranno quindi discussi, in modo più dettagliato, alcune caratteristiche relative ad esso tra cui l'interfaccia e i principali oggetti che lo costituiscono.

### 3.2.1. Grasshopper

Grasshopper è interessante principalmente perché offre agli utenti la possibilità di creare una struttura algoritmica complessa che consiste nel generare una serie di componenti progettati per creare geometrie 3D, facendo in modo che il suo utilizzo risulti molto intuitivo e illustrativo. Il campo applicativo in cui Grasshopper viene prevalentemente utilizzato è quello della progettazione generativa, attualmente in via di sviluppo e basato su codici casuali e vari algoritmi; la progettazione di un componente avviene utilizzando gli strumenti di progettazione generativa di Rhinoceros 6 e Grasshopper, i quali lavorano insieme per creare il componente stesso.

- L'interfaccia del software

Essa è costituita da diversi elementi, come illustrato in figura 27, tra cui è possibile distinguere: la barra principale del menù (A), il Pannello dei componenti (B) e il Canvas (C).

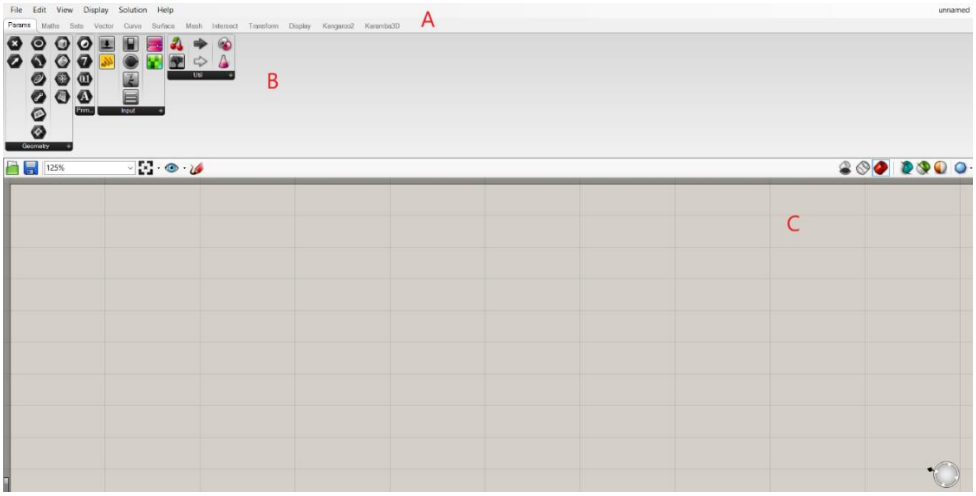



*Figura 27 - Componenti dell'interfaccia di Grasshopper*

Il pannello dei componenti è costituito da tutte le tipologie dei componenti, ognuno dei quali appartiene ad una certa categoria (ad esempio “Param” o “Curves” relativi rispettivamente a tutti i tipi di dati primitivi e a tutti gli strumenti relativi alla creazione di una curva), costituita a sua volta da uno specifico pannello. Per utilizzare uno specifico componente, all’interno del programma, è necessario selezionarlo e trascinarlo sul canvas, ovvero la zona dell’interfaccia dove si compone il progetto.

Poiché, i pannelli di categoria contengono tutti i componenti che vi appartengono, spesso, il loro numero è relativamente elevato e vengono mostrati sul pannello solo i più utilizzati; in ogni caso è possibile visualizzare e selezionare gli altri semplicemente dal menù a tendina nero in basso a sinistra della barra.


*Figura 28 - Componenti geometriche di Grasshopper*

Il Canvas, letteralmente “la tela su cui dipinge il pittore”, rappresenta invece la specifica area dell’interfaccia in cui il designer trascina i componenti e li collega tra loro per creare il suo progetto parametrico.

Esso costituisce la parte interattiva con l’utente rivelando anche una codifica degli oggetti che ne indicano lo stato (figura 29):

- Il colore arancione (A) indica che il parametro (o il componente) contiene degli avvisi; la maggior parte degli oggetti mostra inizialmente questo stato poiché non ancora connesso ad altri dati;
- Il colore rosso (B) indica invece che il parametro o il componente contengono un errore che può derivare o dal componente stesso o da uno dei parametri di input/output;

- Il colore grigio (C) indica invece che il parametro o il componente funzionano correttamente e non presentano né avvisi né errori; è possibile individuare quale parametro genera l'errore semplicemente cliccando sul corpo input/output dell'oggetto.


*Figura 29 - Stato degli elementi mostrato sul canvas*

- Gli oggetti

Gli oggetti costituente il GH sono di diversi tipi; i principali e i più utilizzati, come già accennato, sono:

- Parametri: contengono dei dati e immagazzinano informazioni
- Componenti: contengono delle azioni da svolgere

I parametri contengono dei dati e, dal momento che per essi non è richiesto nessuna connessione sul lato sinistro dell'oggetto, non ereditano dati da qualche altro elemento; essi non presentano né errori né avvisi e sono sempre caratterizzati da una dicitura orizzontale. Quando un parametro viene aggiunto al canvas, è di colore arancione, ad indicare che non contiene ancora dati e non ha alcun effetto sul risultato della soluzione dell'History (storia di costruzione del progetto); dal momento in cui ad esso viene attribuito un dato il suo colore diventa grigio e, da quel momento, è possibile utilizzarlo all'interno dell'History come input per ulteriori elementi. Un componente richiede invece, normalmente, dei dati di input per poter svolgere un'azione che permette di ottenere un risultato; per questo motivo la maggior parte dei componenti presenta un set di parametri, necessari per compiere quella specifica azione, di input (situati sul lato sinistro del componente) e di output (situati sul lato destro del componente). A titolo d'esempio, in figura 30, vengono differenziati gli input e gli output del componente Vettore

(abbreviato “Vec”) che richiede, in ingresso, le tre coordinate (X,Y,Z) e fornire come output il costrutto vettore (V) e la sua lunghezza (L).

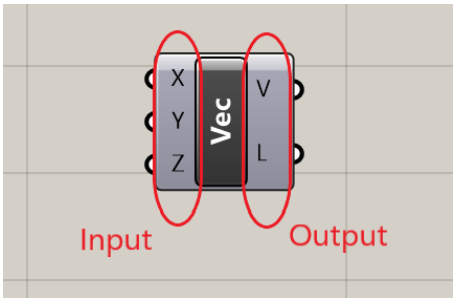
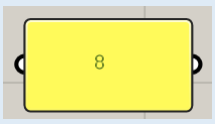



*Figura 30 - Inputs e outputs del componente "Vettore"*

A titolo d’esempio, si riportano inoltre alcuni degli oggetti (parametri e componenti) usati in Grasshopper per l’implementazione di un algoritmo, seguiti da una loro breve descrizione (Figura 31).

| NOME | IMMAGINE | DESCRIZIONE |
|---|---|---|
| **Point Parameter** |   | Rappresenta una raccolta di coordinate del punto 3D. I parametri dei punti sono in grado di memorizzare dati persistenti |
| **Number slider** |   | Si tratta di uno speciale oggetto dell’interfaccia che consente l’impostazione rapida di singoli valori numeri che possono essere modificati, con conseguenti effetti in tempo reale sull’algoritmo, semplicemente muovendo il cursore lungo lo slider. Trattandosi di parametri, essi non richiedono valori di input ma, grazie ai loro output, rappresentano valori di input per altri oggetti. |
| **Construct Point** |   | Costruisce un punto una volta passate le coordinate x,y,z. |
| **Text Panel** |   | Un pannello è come un adesivo Post-It; tipicamente è un oggetto inattivo che consente di aggiungere piccole osservazioni o spiegazioni a un documento e, a volte, anche di passare dei parametri fissi. Inoltre, se un parametro di output viene collegato ad un pannello, |

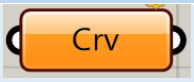
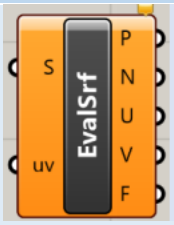
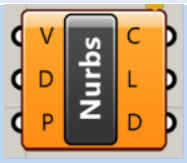
| | | |
|---|---|---|
| | | esso consente anche di vedere il contenuto di quel parametro in tempo reale. |
| **Series** |  | Crea una serie di valori numerici fornendo come output una lista (lista di punti, lista di linee, ecc). |
| **Curve** |  | Vincola una raccolta di curve generiche. |
| **Divide Curve** |  | Divide una curva in segmenti di uguale lunghezza |
| **Offset Curve** |  | Offset della curva, passata come input, di una distanza specificata. |
| **Move** |  | Trasla un oggetto lungo un vettore specificato. |
| **Evaluate Surface** |  | Valuta le proprietà della superficie locale in corrispondenza di una coordinata {uv} |
| **Nurbs Curve** |  | Costruisce una curva nurbs dai punti di controllo |

*Figura 31 - Tabella di alcuni componenti e parametri usati in Grasshopper*

Infine, a scopo prettamente illustrativo, si riportano di seguito due semplici algoritmi implementati in Grasshopper, usando alcuni degli oggetti sopra citati.

Il primo algoritmo (figura 32) ha la funzione di trovare uno o più punti su una curva o una superficie in base ad un parametro passato; il componente < evaluate curve > prende infatti una curva e un parametro numerico, passato tramite un <number slider>, e restituisce un punto sulla curva in base a questo parametro.

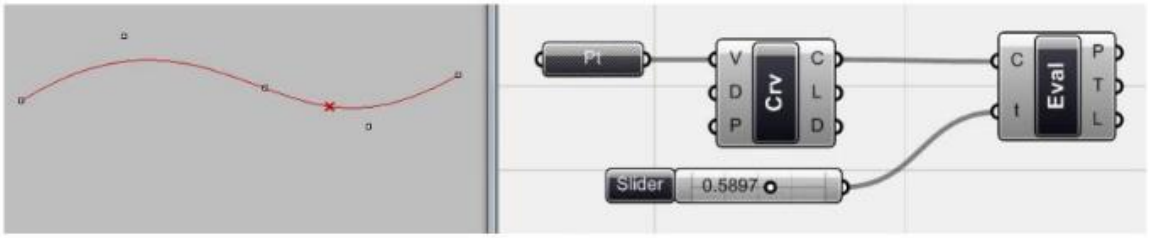



*Figura 32 –Valutazione, su una curva generata, di un punto in base ad un parametro specifico proveniente da un <number slider>*

Il secondo algoritmo (figura 33) genera dei punti a spirale utilizzando delle funzioni matematiche. Viene creato un <range> di numeri da 0 a 2 moltiplicato per 2Pi tramite il componente <Function> che a sua volta crea un range numerico da 0 a 4Pi; questo range viene inoltre suddiviso in 60 parti e il risultato alimenta il componente <Pt> attraverso la seguente funzione matematica:

$X = t * Sin(t), Y = t * Cos(t).$

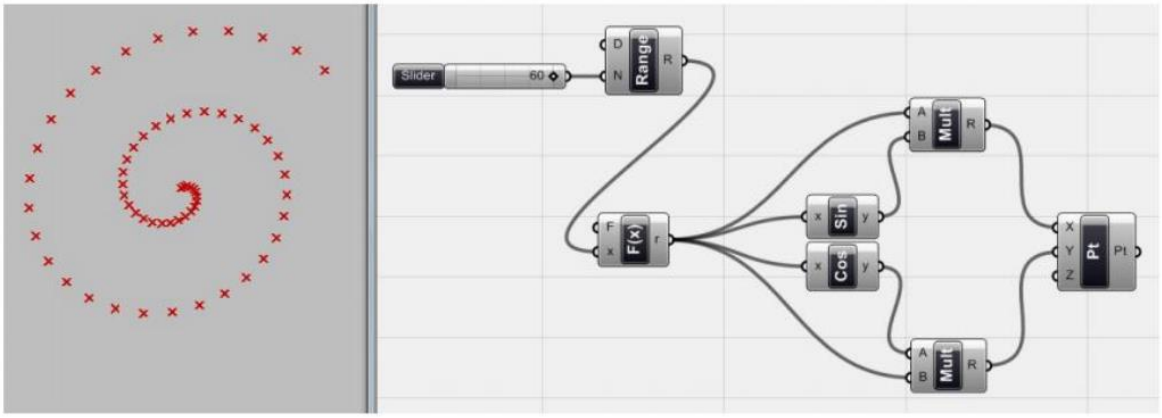



*Figura 33 – Generazione di una serie di punti a spirale mediante funzione matematica*

## 3.3. Gli algoritmi nei processi di design

L'unione tra il design e le scienze fisico-matematiche-computazionali si propone non solo di sistematizzare processi e pratiche ma anche di dividere i problemi di progettazioni in parti più piccole con lo scopo di definire strutture logiche che possano essere interconnesse tra loro, ovvero gli algoritmi.

Si dice generalmente che l'algoritmo supera i limiti degli strumenti CAD tradizionali e aiuta anche il progettista a lavorare oltre la geometria e lo spazio tridimensionale offrendogli la

possibilità di incorporare conoscenze derivanti da una varietà di discipline nell'esplorazione del design al fine di generare uno spettro di forme e possibilità tra cui scegliere. L'algoritmo può essere visto come una macchina astratta la cui applicazione, in CAD e software di modellazione 3D, consente il calcolo e l'incorporazione di tecniche computazionali all'interno progettazione architettonica.

### 3.3.1. Il concetto di algoritmo

Il concetto di algoritmo può essere visto come un metodo basato su obiettivi per eseguire un'attività e può quindi essere definito come un insieme di istruzioni o regole da seguire con criterio; un modo procedurale, passo dopo passo, per descrivere una serie di azioni da seguire al fine del raggiungimento di un obiettivo particolare come la risoluzione di un problema o la ricerca di qualcosa (una condizione di ottimo ad esempio).

Esso è costituito da una sequenza di operazioni in cui c'è un punto di partenza, che riceve zero o più input, e un corpo di elaborazione ovvero un insieme finito e definito di istruzioni che specifica, in modo inequivocabile, i passaggi per l'elaborazione del calcolo dell'input dato; una volta eseguiti gli steps, l'algoritmo è in grado di generare l'output prendendo l'input attraverso la sequenza di passaggi ben definiti. Un algoritmo, in termini sintetici, non è altro che un insieme di istruzioni di dimensioni finite create dall'uomo, il cui utilizzo implica logica e calcolo formali.

Solitamente gli algoritmi vengono utilizzati per elaborare delle informazioni relative alla risoluzione di problemi complessi, che vanno oltre la capacità di un essere umano. Ad esempio, è facile sommare due piccoli numeri usando le dita, ma quando il numero delle cifre aumenta e l'attività diventa più elaborata (sommare due numeri più grandi, quindi moltiplicarli con un altro numero, dividerli con un nuovo numero e così via) anche una persona che sa come sommare e sottrarre piccoli numeri impiegherebbe un algoritmo per definire la procedura aritmetica e risolvere il problema.

Il concetto di algoritmo non è nuovo tant'è che il primo algoritmo non banale fu proposto da Euclide per calcolare i massimi comuni divisori; le origini della parola *algoritmo*, così come l'idea di studiarli, deriva dal matematico persiano del IX secolo al-Khwarizmi, che "espose i metodi di base per sommare, moltiplicare e dividere i numeri (estraendo anche le radici quadrate e calcolando le cifre del $\pi$ ) tramite procedure per aiutare gli esseri umani a risolvere i problemi in modo preciso, inequivocabile, meccanico, efficiente e corretto.

Nonostante esso rimane un concetto difficile da definire, dal momento che non esiste una definizione formale univoca di algoritmo, è possibile definirne, in modo informale, diverse caratteristiche:

1) Un algoritmo è dato da un insieme di istruzioni di dimensione finita;
2) C'è un agente informatico (solitamente umano) che può reagire alle istruzioni ed eseguire i calcoli;
3) Sono presenti strutture per creare, memorizzare e recuperare passaggi in un calcolo;
4) Sia P un insieme di istruzioni (punto 1) ed L un agente informatico (punto 2); allora L reagisce a P in modo tale che, per ogni dato di input, il calcolo viene eseguito in modo discreto e graduale, senza utilizzare metodi continui o dispositivi analogici;
5) L reagisce a P in modo tale che il calcolo viene effettuato in modo deterministico, senza ricorrere a dispositivi casuali.

Secondo le caratteristiche sopra citate, la definizione informale descrive quindi l'algoritmo come un insieme di istruzioni ben definite che possono essere eseguite tramite un agente informatico, il quale "reagisce alle istruzioni ed esegue i calcoli" (Rogers 1967, 2). È infatti solo grazie all'agente di calcolo che un algoritmo è in grado di elaborare le informazioni e fornire i risultati al problema dato. L'interdipendenza tra l'algoritmo e l'agente informatico indica l'importanza dei collegamenti di comunicazione tra di loro. I passaggi e le regole di elaborazione nell'algoritmo devono essere espressi rispetto alla capacità dell'agente di calcolo e cioè, affinché l'algoritmo possa essere eseguito da un essere umano, le istruzioni devono essere espresse in un linguaggio che possa essere compreso ed eseguito da un essere umano; diventa quindi importante che il tipo di algoritmo e il linguaggio in cui esso è espresso siano adattati al tipo di agente informatico per elaborare le informazioni specifiche.

Guardando al campo di applicazione in CAD, come accennato precedentemente, l'aumento dell'algoritmo come mezzo di progettazione in architettura è stato reso possibile dai linguaggi di scripting e dagli ambienti disponibili di grafica 3D e dei software CAD, che hanno permesso ai progettisti di andare oltre i limiti della modellazione tradizionale.

Per creare modelli 3D, il progettista descrive ciò che il computer dovrebbe fare in passaggi precisi (tramite un diagramma di flusso ad esempio), esprimendo tali passaggi in termini di algoritmo all'interno dell'interfaccia algoritmica (ambiente di scripting o algoritmo grafico) chee, una volta eseguiti, vengono elaborati dal computer per generare gli output. Utilizzando questo tipo di interfaccia, il progettista può affrontare una maggiore complessità nella progettazione; il designer non sta infatti progettando il prodotto finale, ma definendo la

relazione associativa tra gli elementi geometrici e i fattori influenti coinvolti in un quadro di progettazione in termini di algoritmi generici. Pertanto, molti parametri potrebbero essere collegati al modello di esplorazione del progetto.

Il pensiero algoritmico è quindi usato in ambiente CAD per descrivere un modo di pensare e lavorare nel design e consente l'elaborazione di calcolo delle informazioni, come parte del processo di progettazione con l'ulteriore vantaggio di consentire al progettista di progettare liberamente e modificare l'algoritmo, la logica e la definizione del costrutto geometrico.

### 3.3.2. La natura degli algoritmi e l'esplorazione della forma geometrica

La sinergia tra algoritmo e computer (o l'applicazione dell'algoritmo nei programmi di modellazione 3D) apporta *"*generatività*"* e un certo grado di automazione al processo di design. La parola *generative* è la capacità di generare, produrre, creare e riprodurre output sotto forma di numeri, testo, immagini, geometrie 2D o 3D e altre forme ed è usata in matematica per descrivere la capacità di formare una linea, una superficie o un solido spostando teoricamente un punto, una linea o una superficie rispettivamente.

L'automazione denota invece la parte autonoma di un processo che funziona solo tramite un controllo umano diretto minimo del processo di progettazione ovvero una fase del processo che procede spontaneamente senza il controllo consapevole del progettista. Una volta che il progettista definisce infatti i passaggi e le regole dell'algoritmo, la sua esecuzione, effettuata mediante calcolo digitale, è in grado di interpretare le regole e tradurle automaticamente in output.

Il compito del designer non diventa quindi quello di progettare gli output, ma piuttosto le regole e i processi che il programma deve eseguire automaticamente affinché tali output possano essere prodotti; il progettista non si preoccupa di come disegnare una curva, ad esempio, ma piuttosto di capire come descrivere il processo di realizzazione della curva affinché essa possa essere generata automaticamente. L'utilizzo di un algoritmo come strumento di progettazione consente di generare tutte le potenziali curve tramite una funzione matematica in modo tale che i computer possano visualizzare una famiglia pressoché infinita di curve che condividono lo stesso processo, di cui i parametri possono essere inoltre modificati a piacimento.

L'algoritmo come mezzo per la concettualizzazione delle idee e la generazione di output si basa, come detto, sulla formalizzazione da parte del designer di una idea mediante un insieme di regole in modo chiaro e sequenziale, da calcolare; nella progettazione architettonica, le idee non sono sempre chiare nella fase iniziale della progettazione e vengono quindi sviluppate

attraverso un processo di progettazione iterativo che consente la valutazione degli output del processo di progettazione e il conseguente perfezionamento del progetto.

Seguendo questo metodo, una prima domanda che è lecito porsi riguarda il come formulare idee chiare in termini di algoritmo, un'altra concerne il risultato soddisfacente degli outputs in maniera automatizzata.

La risposta alla seconda domanda è semplice e immediata: gli algoritmi non producono automaticamente risultati soddisfacenti ma gli output devono essere migliorati attraverso l'osservazione cosciente e la valutazione critica del progettista; egli deve quindi modificare la logica, le regole e i parametri dell'algoritmo iniziale per produrre nuovi set di output. Infine, dopo una serie di iterazioni di riforma dell'algoritmo e l'osservazione e la valutazione dei suoi output da parte del progettista, l'algoritmo viene interpolato e perfezionato al fine di sviluppare il design (Figura 21). Ciò richiede un'interazione tra il progettista e gli output degli algoritmi in modo tale che il progettista faccia osservazioni significative, valuti gli output e di conseguenza sviluppi e interpoli gli algoritmi in base al contesto di progettazione.

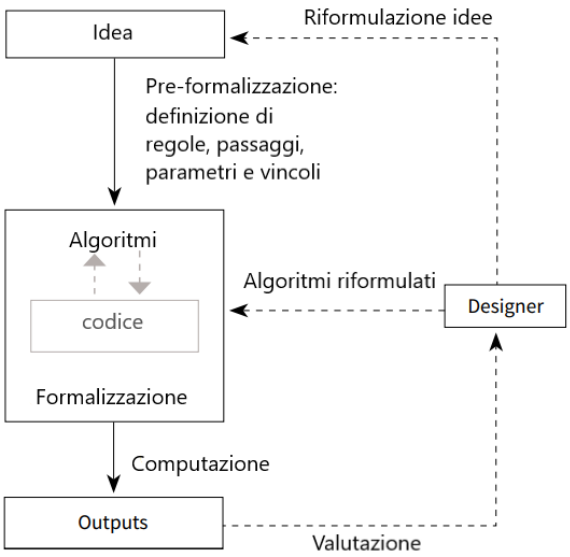



*Figura 34 – Diagramma illustrativo: come l'algoritmo e l'idea progettuale vengono riformulati e la valutazione degli output dopo ogni iterazione*

La natura degli output influenza il modo in cui il designer interagisce con essi e di conseguenza il modo in cui egli sviluppa gli algoritmi finalizzati al prodotto finale. A differenza degli strumenti fisici in cui l'imprevedibilità è di natura meccanica o chimica, gli strumenti algoritmici sono di natura astratta, razionale e intellettuale e quindi correlati alla mente umana;

in tale contesto quindi, l'output di un algoritmo deve essere associato al programmatore o al progettista.

Nel contesto dell'architettura come pratica materiale, è fondamentale inoltre esplorare, esaminare e valutare gli outputs algoritmici oltre il numero e la geometria. Procedere sulla base degli output puri e immediati degli algoritmi limiterebbe il progetto a una valutazione puramente intellettuale e, affinché il progettista possa entrare in un'esplorazione tangibile quando utilizza l'algoritmo, questi output devono essere poi materializzati.

La prima domanda, relativa a come il progettista possa astrarre idee vaghe in modo tale che vengano formalizzate chiaramente in termini di algoritmo, contiene invece qualcosa di contraddittorio. In generale, l'uso dell'algoritmo come strumento per concettualizzare le idee richiede un certo grado di formalizzazione prima dell'atto di progettazione dell'algoritmo: il progettista deve infatti pianificare l'idea progettuale in anticipo e trasformarla in un modello comprensibile, indipendentemente dal fatto che l'idea abbia origine dalla modellazione fisica o provenga direttamente dalla mente del progettista. Ad esempio, il progettista deve definire quali elementi principali dovrebbero dipendere da altri elementi e definire i parametri del modello e la gerarchia delle dipendenze tra i parametri geometrici e le funzioni parametriche; ovvero, il progettista deve utilizzare la tecnica associativa-parametrica.

Per prima cosa si deve concettualizzare in anticipo cosa si intende modellare e la sua logica, si esegue il debug e si testano tutte le possibili ramificazioni in cui il programma parametrico potrebbe fallire "(Smith 2007, 2). In tal modo i progettisti "possono forzare eccessivamente o scoprire di aver bisogno di aggiustare il programma o iniziare di nuovo a programmare perché hanno adottato l'approccio sbagliato". Pertanto, una volta che l'algoritmo viene introdotto in una fase di progettazione, quella fase viene sistematizzata.

Inoltre, il progettista può creare i modelli fisici in modo tale che possano lavorare all'unisono con l'algoritmo per consentire una esplorazione ibrida digitale e fisica del design. In Figura 22 vengono rappresentati tre diversi diagrammi; a sinistra un'idea viene prima modellata e poi viene prodotto l'algoritmo, al centro prima l'idea viene simultaneamente esplorata con la modellazione fisica e poi viene generato l'algoritmo, a destra invece l'idea viene esplorata simultaneamente con la modellazione fisica e con l'algoritmo.

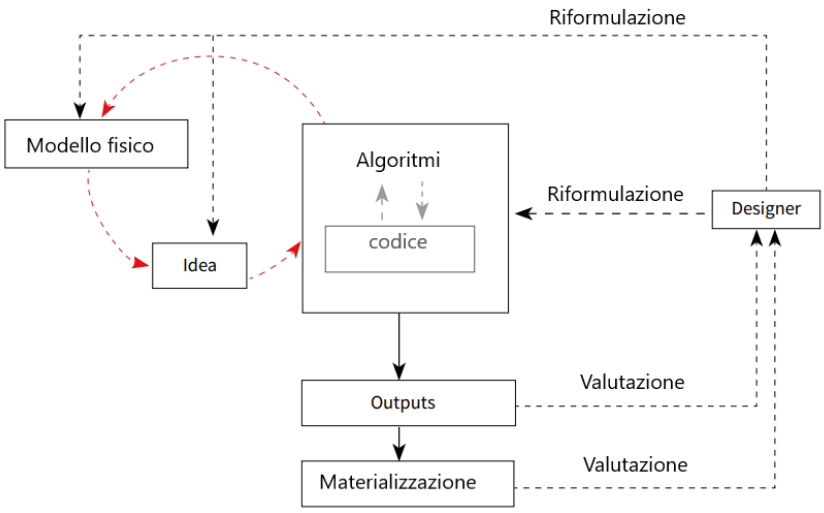



*Figura 35 – Tre differenti procedure algoritmiche seguite nel processo di design*

Guardando all'esplorazione della forma geometrica di un progetto, l'applicazione dell'algoritmo per definire e creare la forma geometrica di una struttura comporta una disposizione sistematica del codice; per mezzo di un codice o di un'interfaccia algoritmica (come Grasshopper ad esempio) un'idea formale viene infatti tradotta in un insieme di regole descritte in termini di algoritmo.

Queste regole o principi descrivono accordi formali che, una volta eseguiti, forniscono una serie di potenziali forme geometriche: "Invece di modellare una forma esterna, il progettista articola una logica generativa interna, che poi produce, in modo automatico, una serie di possibilità tra cui il designer può scegliere". In altre parole, il designer non disegna oggetti geometrici per creare una forma, ma piuttosto descrive l'obiettivo geometrico e le relazioni associate in termini di valori e parametri; ad esempio, per la creazione di un cerchio, il progettista può definire un punto e un raggio o tre punti sul suo perimetro; assegnando poi valori diversi ai parametri di raggio può esplorarne la forma geometrica e generare cerchi variabili. Usando un tale approccio diventa quindi possibile produrre non solo una forma, ma uno spettro completo di forme correlate ogni qualvolta vengono modificati determinati parametri.

L'attenzione si sposta quindi dal disegno di una forma allo "sviluppo di processi, sotto forma di algoritmi o regole generative, dai quali si ottiene un risultato specifico attraverso la definizione dei valori e dei parametri influenti" (Menges 2010, 3).

In figura 23 è riportata, a titolo d'esempio, la logica algoritmica alla base della generazione della forma variabile di un tipo di torre; in questo algoritmo il numero di piani, l'impronta e la

posizione di ogni piano su quell'impronta sono variabili. I parametri sono assegnati agli elementi geometrici e, ogni cambiamento di questi parametri influenza la forma complessiva della torre.

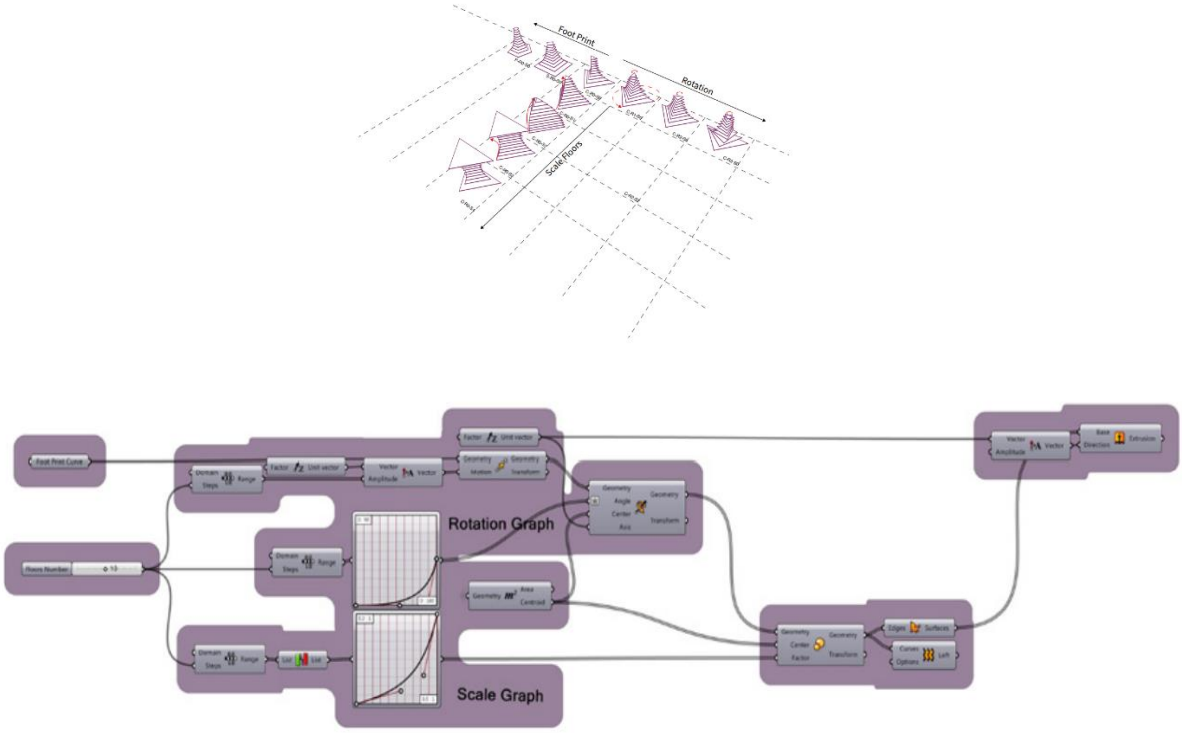



*Figura 36 – Logica algoritmica alla base della generazione della forma variabile di un tipo di torre*

Come già accennato precedentemente, l'applicazione dell'algoritmo in architettura è realizzata tramite CAD e software di modellazione 3D commerciali, tramite scripting, programmazione o interfacce algoritmiche dotate di sistema associativo e parametrico. Il sistema parametrico-associativo, utilizzato nei modelli CAD-geometrici, impiega una particolare composizione sequenziale e gerarchica delle geometrie, nonché un particolare modo mirato alla propagazione dei cambiamenti attraverso la struttura della dipendenza geometrica.

L'uso dell'algoritmo, per definire la logica della forma e calcolare le forme geometriche, implica la definizione e la descrizione del modo particolare in cui le parti e l'intero esistono e si relazionano tra loro per generare forme complesse; pertanto, il modo in cui il progettista descrive la geometria e le sue relazioni geometriche è cruciale, poiché tale descrizione avrà un impatto sullo spettro dei probabili risultati. Ciò che è importante capire è che la logica di un modulo deve essere descritta prima di calcolare ed esplorare i possibili output. In altre parole, in contrasto con l'esplorazione di forme tramite manipolazione manuale, dove l'esplorazione è intuitiva e parte integrante dell'atto di manipolazione, la creazione di forme mediante algoritmo

richiede che il progettista descriva la logica di calcolo prima dell'atto di esplorazione dei risultati; non è possibile quindi esplorare la forma geometrica senza prestare attenzione alle relazioni tra la parte e il tutto, o tra le componenti e la struttura complessiva.

Secondo Christopher Alexander in *Notes on the Synthesis of Form,* "ogni aspetto di una forma può essere inteso come una struttura di componenti; ogni oggetto è una gerarchia di componenti, con i più grandi che specificano il modello di distribuzione di quelli più piccoli e questi ultimi che, a loro volta, specificano la disposizione e la distribuzione di componenti ancora più piccoli ".La metodologia di progettazione offerta dall'algoritmo consente ai progettisti di accedere e modificare ogni livello dei componenti in qualsiasi momento durante l'intero processo di progettazione.

### 3.3.3. Introduzione agli algoritmi genetici

Negli ultimi decenni un grande interesse è stato posto nei confronti dei processi di evoluzione presenti in natura al punto da radicarsi l'idea di simularli per sviluppare metodi e algoritmi che, nel concreto, permettessero di risolvere i problemi di ricerca e ottimizzazione.

Gli algoritmi genetici (GA) sono procedure complesse, adattative che traggono la loro inspirazione dal principio della selezione naturale proposto, nel 1859, da Charles Darwin; l'idea alla base della teoria evoluzionistica è che gli organismi biologici si evolvono in base al principio della selezione naturale secondo cui solo quelli più adatti a vivere in specifiche circostanze sopravvivono. A causa di questa selezione, individui con scarso rendimento hanno meno probabilità di sopravvivere e gli individui più adatti o "in forma" si riproducono mescolando i propri patrimoni genetici, ovvero generando un numero relativamente elevato di prole a cui trasmettono specifici tratti (cromosomi). La riproduzione può, in tal caso, produrre una ricombinazione delle buone caratteristiche di ogni antenato, e la prole può presentare una forma fisica migliore di un genitore semplicemente ereditandone il "gene buono"; il risultato di tale processo è che, dopo poche generazioni, le caratteristiche vantaggiose diventano dominanti nella popolazione e le specie evolvono spontaneamente per adattarsi sempre più al loro habitat naturale.
A partire dalla seconda metà degli anni '50, molti ricercatori iniziarono a studiare dei sistemi "intelligenti" in grado di autoriprodursi o di avvicinarsi a sistemi artificiali che potessero implementare i processi biologici, generando così gli algoritmi definiti "evolutivi" (di cui i GA fanno parte).

L'aggettivo "genetico" per gli algoritmi deriva proprio dal fatto che il modello utilizzato trova spiegazioni nella branca della biologia detta genetica e, soprattutto, nel fatto che essi usano meccanismi concettualmente simili a quelli dei processi biologici. Un algoritmo genetico può quindi essere definito come una tecnica stocastica di ottimizzazione, che procede in modo iterativo, capace di valutare delle soluzioni di partenza per il problema da risolvere che, una volta ricombinate, in seguito agli incroci riproduttivi e introducendo anche degli elementi di disordine, evolve in modo automatico e per raffinamenti successivi, nonché converge verso una soluzione ottimale del problema oggetto.
I GA nascono quindi per gestire le possibili soluzioni, di un problema dato, che vengono fatte evolvere mediante l'applicazione di un certo numero di operatori stocastici; nel linguaggio tecnico degli algoritmi genetici si ha che:

- la funzione da minimizzare prende il nome di ***fitness*** (F);
- supponendo che tale funzione dipende da n variabili, ovvero $F = f(x_1, x_2, \ldots, x_n)$, il set di n valori $x_1, x_2, \ldots, x_n$ appartiene ad un certo intervallo e viene detta ***individuo*** (o ***soluzione***);
- l'insieme di tutte le soluzioni, ovvero di tutti gli individui presenti, costituisce la ***popolazione*** all'interno della quale va ricercato l'individuo migliore (quello con valore di fitness più basso, nel caso di un problema di minimizzazione);
- ogni individuo (soluzione) viene rappresentato, in forma del tutto astratta, ad esempio come codice binario (stringa) di lunghezza fissata, ovvero tramite una specifica sequenza biunivoca di numeri 0 e 1 chiamata ***cromosoma***. Così facendo, qualunque struttura dati, per quanto complessa e articolata; può essere sempre codificata nella memoria del calcolatore. La specifica sequenza di 0 e 1, da cui è possibile ricostruire una specifica soluzione, ricorda moltissimo un filamento di DNA; in altre parole, possiamo considerare una stringa binaria come il DNA di una soluzione del problema oggetto. Si osserva che, in ogni caso, tale tipo di codifica binaria è la più generale ma non è l'unica e non risulta sempre la più conveniente.

Un algoritmo genetico è principalmente composto da due parti; la prima parte comprende una procedura capace di generare la popolazione iniziale (random o usando qualche euristica), la seconda è invece costituita da un ciclo evolutivo che, ad ogni iterazione (o generazione), crea una nuova popolazione applicando degli operatori genetici alla popolazione precedente. L'iterazione permette l'evoluzione verso una soluzione ottimale del problema; come infatti

affermato dalla teoria di Holland, dopo una fase iniziale nella quale il GA esplora in modo quasi randomico lo spazio di ricerca o di campionamento, esso si concentra sulla regione più "promettente" ovvero quella caratterizzata da individui con fitness migliore.
La grande potenzialità dei GA risiede principalmente nel fatto che il processo evolutivo della popolazione di cromosomi corrisponde a una ricerca nello spazio delle soluzioni potenziali; quest'ultima richiede quindi l'equilibrio di due componenti apparentemente in contrasto: lo sfruttamento delle soluzioni migliori e l'esplorazione dello spazio delle soluzioni. Si tratta quindi di una vera e propria capacità di bilanciamento, ed è proprio tale capacità che differenzia gli algoritmi genetici dalle altre metodologie che invece sfruttano costantemente le soluzioni migliori trovate rinunciando all'esplorazione dello spazio delle soluzioni, o dalla ricerca casuale che, al contrario, esplora l'intero spazio senza dare alcuna importanza alle regioni mostratesi più performanti.

Fatta questa premessa, viene adesso posta l'attenzione sul funzionamento effettivo di un GA. In realtà il funzionamento è ben più complesso e prevede oltre alla selezione, l'applicazione di altri operatori genetici, che traggono quindi anch'essi inspirazione dai processi evolutivi, all'insieme delle soluzioni iniziali.

I GA operano, come detto, su una popolazione di cromosomi artificiali che vengono fatti riprodurre in modo selettivo; durante la riproduzione i cromosomi degli individui migliori vengono accoppiati in modo random e parte del materiale genetico viene scambiato, mentre alcune mutuazioni casuali alterano localmente la struttura del materiale genetico originale. Queste nuove strutture determinano la formazione di una nuova generazione e il processo continua fino a quando non viene generato un individuo corrispondente a una soluzione accettabile per il problema.

I tre operatori principali di un GA risultano quindi: la *selezione*, e la riproduzione effettuata mediante ricombinazione genetica (*crossover)* e mutazione genetica (*mutation*). Prima di analizzare in modo più dettagliato le caratteristiche degli operatori, e il come essi lavorano all'interno dell'algoritmo, si riporta di seguito il digramma di flusso del funzionamento di un GA:

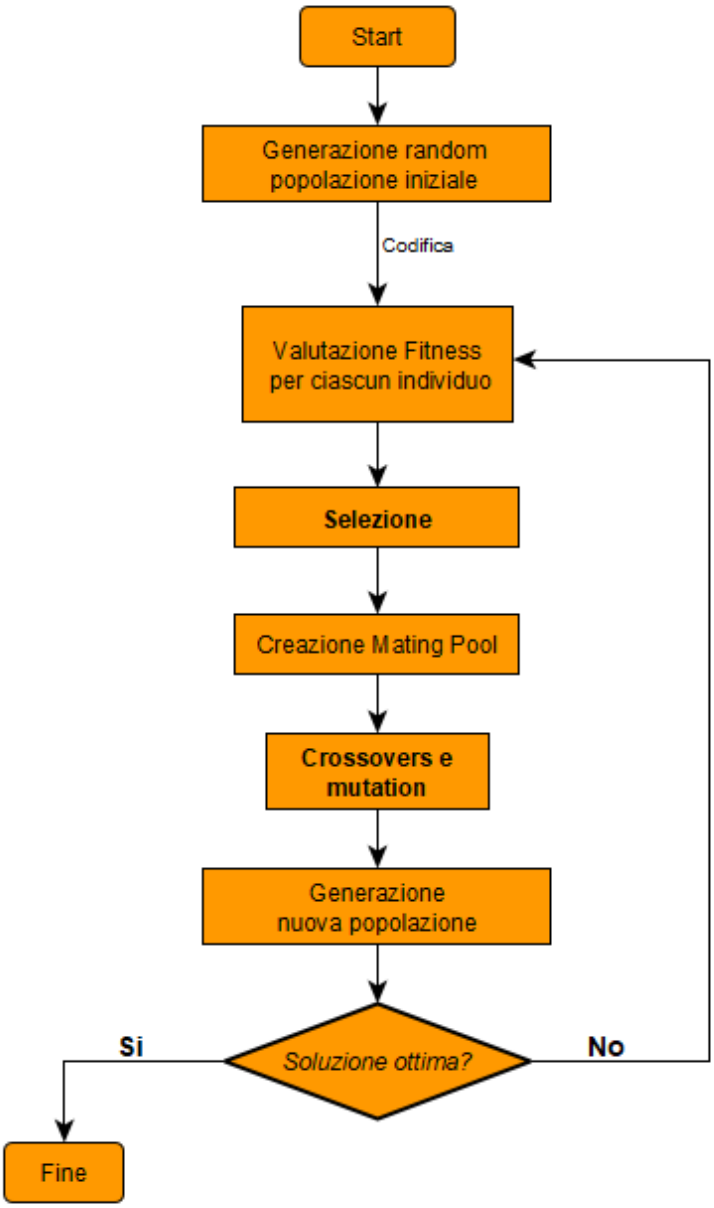



*Figura 37 – Diagramma di flusso di un GA*

Come è possibile notare, l'algoritmo è iterativo ed è costituito da più steps sequenziali, ognuno dei quali svolge una specifica funzione all'interno del processo risultando indispensabile per la convergenza alla soluzione ottima, o comunque ritenuta "buona" per il problema.

- **Generazione della popolazione iniziale**

Il GA inizia con la generazione di un gruppo di individui (cromosomi), noto come popolazione, che può essere effettuata mediante campionamento uniforme, casuale o utilizzando una

procedura, definita complementare, che prevede di generare casualmente la metà dei cromosomi e in modo complementare la restante parte.

Idealmente, l'obiettivo è avere la prima popolazione con un *pool genetico* il più ampio possibile in modo da poter esplorare l'intero spazio di ricerca e aiutare l'algoritmo a trovare più velocemente buone soluzioni; se la popolazione risultasse infatti carente di diversità, esso esplorerebbe soltanto una piccola parte dello spazio di ricerca e non troverebbe mai delle soluzioni ottimali globali. In questa fase, un'importante variabile da definire all'interno dell'algoritmo diventa quindi la dimensione della popolazione iniziale. Essa dipende dalla complessità del problema analizzato e, banalmente, maggiore è il numero degli individui costituente la popolazione più risulta facile esplorare lo spazio di ricerca; una scelta accurata di tale dimensione incide inoltre, in modo importante, sull'efficienza di un GA a convergere verso ottimi globali, piuttosto che ottimi locali. Una popolazione di 100 individui risulta, nella maggior parte dei casi, sufficiente, ma può comunque essere modificata in funzione del tempo e della memoria dei calcolatori o della qualità del risultato che si vuole ottenere.

A titolo d'esempio, si riporta di seguito una popolazione iniziale (espressa in forma binaria) costituita da soli quattro individui.

| | | |
|---|---|---|
| Popolazione | Individuo 1 | 0 1 1 0 0 |
| | Individuo 2 | 1 1 0 0 1 |
| | Individuo 3 | 0 0 1 0 1 |
| | Individuo 4 | 1 0 0 1 1 |

*Figura 38 – Esempio di individui costituenti una popolazione*

- **Valutazione della funzione di fitness**

Scelta la funzione che si vuole ottimizzare nel processo di ottimizzazione, ovvero la funzione obiettivo (o funzione di *fitness*), essa viene applicata alle soluzioni appartenenti alla popolazione attuale, insieme di variabili di input, generando un output. La funzione di fitness serve infatti per giudicare le prestazioni e l'idoneità di una soluzione, o set di parametri, relativo al problema che vogliamo risolvere; essa fornisce un valore numerico per ciascun individuo della popolazione proporzionale alla bontà della soluzione offerta e consente quindi di confrontare tra loro gli individui e stabilire quale tra questi è "migliore". Da un punto di vista pratico, si può pensare alla funzione obiettivo come una forma di costo, o utilità, o a una qualunque caratteristica che si è deciso di minimizzare (o massimizzare); essa svolge un ruolo

analogo a quello dell'ambiente fisico per gli organismi biologici, in quanto misura le prestazioni dell'individuo stesso (per questo motivo viene infatti definita funzione di *fitness*).

Si consideri, come esempio, un problema di massimizzazione con la seguente funzione di fitness:

$$f(x) = x^2$$

Per fare ciò si procede andando a calcolare i valori di *x* decodificati corrispondenti a ciascun individuo della popolazione considerata; a titolo d'esempio, per l'individuo 1 (01100) il valore di x può essere calcolato come segue:

$$01100 = 0 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 0 \times 2^0$$

$$= 0 + 8 + 4 + 0 + 0 =$$

$$= 12$$

Il valore di fitness per l'individuo considerato è ottenuto semplicemente sostituendo il valore di $x$ nella funzione $f(x)$*:*

$$f(x) = x^2 = (12)^2 = 144$$

Procedendo in modo analogo per l'intera popolazione si ottengono i valori di fitness per tutti gli individui, come illustrato nella seguente tabella:

| N° Individuo | Popolazione iniziale (random) | Valore $x$ | **Valore di fitness $f(x) = x^2$** |
|---|---|---|---|
| *1* | 0 1 1 0 0 | 12 | **144** |
| *2* | 1 1 0 0 1 | 25 | **625** |
| *3* | 0 0 1 0 1 | 5 | **25** |
| *4* | 1 0 0 1 1 | 19 | **361** |
| *massimo* | | | **625** |

*Figura 39 - Calcolo dei valori di fitness della popolazione*

- **Selezione**

Una volta calcolata la *fitness* per ciascun individuo, lo step successivo prevede di selezionare una sottopopolazione per generare delle nuove soluzioni, ovvero quali individui devono essere privilegiati per riprodursi. Questo operatore trae la sua inspirazione dalla selezione naturale

Darwiniana secondo la quale gli individui più "adatti" hanno maggiore probabilità di sopravvivere e quindi di riprodursi, andando a costituire la base per la nuova generazione.

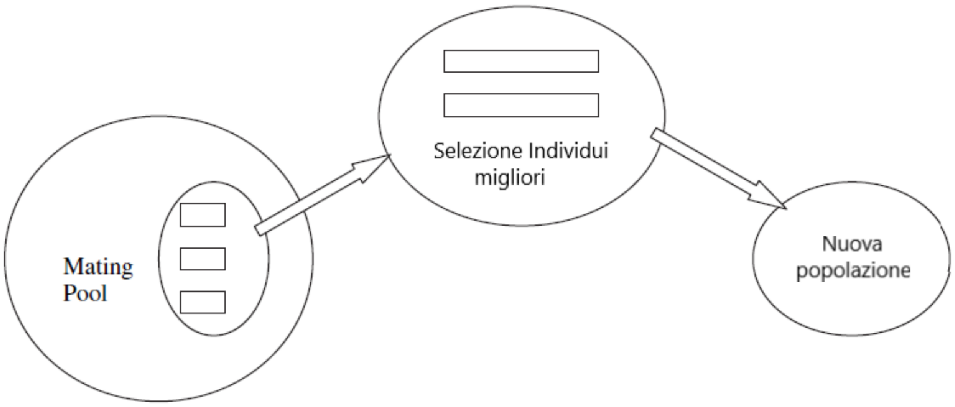



*Figura 40 - Operatore di selezione nei GA*

Il processo di selezione naturale deve avvenire ad ogni generazione (iterazione dell'algoritmo) consentendo alla popolazione di cromosomi di evolversi nel corso delle generazioni fino a membri più adatti alla funzione di costo.

La scelta degli individui più performanti per la riproduzione è basata, oltre che sui valori della funzione di *fitness*, su specifiche tecniche usate per selezionare tale insieme delle soluzioni (definito *mating pool)* tra cui la più semplice è la "selezione tramite troncamento" che prevede di ordinare gli $n$ individui della popolazione secondo il loro valore di fitness e di scegliere, all'interno della *mating pool*, solo le prime $k < n$ soluzioni andando a troncare la restante parte degli individui. Considerando una generica popolazione di $n$ individui ordinati in modo crescente, se l'algoritmo deve risolvere un problema di minimo, andrà a selezionare le prime $k$ soluzioni, viceversa nel caso di un problema di massimo verranno estratte le ultime $k$ soluzioni. Poiché $k < n$, la nuova popolazione avrà un numero di individui inferiore rispetto alla popolazione iniziale; per mantenere la dimensione della popolazione iniziale si procede infine a generare $n - k$ delle copie degli individui selezionati.

- **Crossover**

Il *crossover* rappresenta il primo operatore di riproduzione all'interno degli algoritmi genetici. Una volta selezionati gli individui che costituiscono il *mating pool*, lo step successivo prevede di scambiarne e ricombinarne il loro materiale genetico, come nel caso della riproduzione sessuata in natura, per creare una nuova popolazione i cui individui presentino un codice genetico mutato in parte dall'uno e in parte dall'altro genitore.

Esso fonde quindi le informazioni dei cromosomi di una coppia di individui, scelta in modo del tutto casuale, per esplorare lo spazio di ricerca.

Il *crossover* opera, nella sua forma più semplice e applicato al caso binario, come segue:

- Si scelgono a caso due individui genitori, o stringhe, all'interno del *mating pool* (*parents*);
- Si sceglie casualmente una posizione $k$ all'interno delle stringhe, detto punto di crossover, con $k$ che può variare tra 1 e la lunghezza della stringa $l$ meno 1 [$l$ ; $l$-$1$];
- Si vanno a creare due nuove stringhe figlie (*offsprings*) scambiando i caratteri a destra del punto di crossover, ovvero i caratteri dalla posizione $k+1$ a $l$ compresi.
- L'operatore di crossover viene applicato, in accodo a una prefissata probabilità $P_{cross}$), $N_{pop}/2$ volte in modo da ottenere $N_{pop}$ discendenti. $P_{cross}$ (probabilità di crossover per coppia di individui, detta *crossover rate*) è il parametro di base del processo poiché descrive la frequenza con cui verrà eseguito il crossover. Se $P_{cross} = 0\%$ la nuova generazione sarà composta da copie esatte dei cromosomi della precedente popolazione (ciò non implica però che la generazione sia la stessa), se $P_{cross} = 100\%$ tutta la prole verrà creata con il crossover. La probabilità di crossover è quindi applicata nella speranza che i nuovi cromosomi contengano parti buone dei vecchi cromosomi, risultando migliori, ma facendo in modo che una parte della vecchia popolazione sopravviva in ogni caso alla generazione successiva. La $P_{cross}$ potrebbe essere fissata o calcolata usando la seguente formula:

$$P_{cross} = \begin{cases} P_{c1} = \dfrac{(P_{c1} - P_{c2})(f' - f_{avg})}{f_{max} - f_{avg}} & f' \geq f_{avg} \\ P_{c1} & f' < f_{avg} \end{cases}$$

  dove $f_{max}$ è il valore di fitness più alto nella popolazione, $f_{avg}$ è il valore medio della fitness della popolazione e $f'$ rappresenta il valore di fitness più elevato tra due individui.

Tale procedura, definita *single point crossover*, è illustrata nella seguente figura:

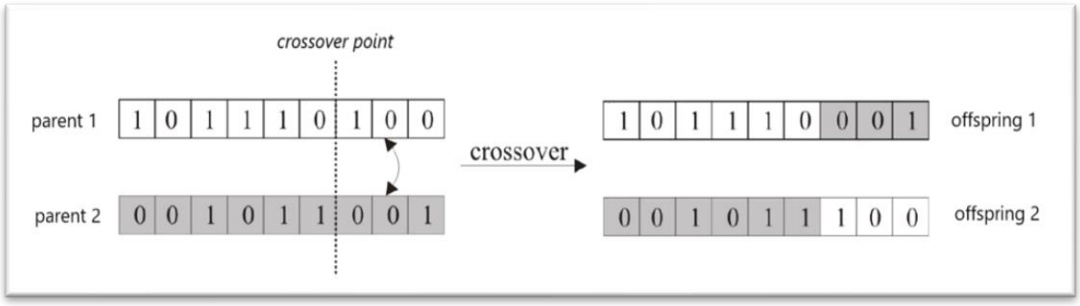



*Figura 41 - Esempio dell'operatore crossover nei GA*

Un'alternativa applicazione del crossover al GA, che in realtà risulta più performante nel caso di una codifica reale delle variabili o dei GA continui, prevede di utilizzare una strategia differente che consente di combinare insieme, tramite un coefficiente $\beta$ , i singoli valori delle variabili dei genitori per ottenere i singoli valori delle variabili dei figli. Considerando due generici individui *parents*:

$$\textit{parent 1} = x^1 = [\ x_1^1, x_2^1\ , x_3^1\ , x_4^1, x_5^1 \ldots\ , x_n^1\ ]$$

$$\textit{parent 2} = x^2 = [\ x_1^2, x_2^2\ , x_3^2\ , x_4^2, x_5^2 \ldots\ , x_n^2\ ]$$

Secondo quanto visto, l'applicazione del crossover single-point produrrebbe ad esempio i seguenti figli:

$$\textit{offspring 1} = y^1 = [\ x_1^1, x_2^1\ , x_3^1, x_4^2, x_5^2 \ldots\ , x_n^2\ ]$$

$$\textit{offspring 2} = y^2 = [\ x_1^2, x_2^2\ , x_3^2\ , x_4^1, x_5^1 \ldots\ , x_n^1\ ]$$

Il problema è che, utilizzando un metodo del genere, nessuna nuova informazione verrebbe introdotta e ogni valore avviato casualmente nella popolazione inziale viene propagato nella generazione successiva solo in differenti combinazioni; per risolvere questo problema tale metodo consente di calcolare i nuovi valori delle variabili della prole usando la formula:

$$y_i^1 = x_i^1 + \beta_i \left| x_i^1 - x_i^2 \right|$$

$$y_i^2 = x_i^2 + \beta_i \left| x_i^1 - x_i^2 \right|$$

In cui $\beta$ è un numero random, che soddisfa la distribuzione di Laplace, calcolato come segue:

$$\beta_i = \begin{cases} a - b\ \log(u_i) & r_i \leq 1/2 \\ a + b\ \log(u_i) & r_i \leq 1/2 \end{cases}$$

Dove a è il parametro di locazione e b>0 è il parametro di scala. Se le variabili decisionali hanno la restrizione di essere interi allora $b = b_{int}$ altrimenti allora $b = b_{reale}$ ; il parametro di scala risulta quindi differente per variabili di progetto reali o intere. Se il valore di b è piccolo è probabile che la prole venga prodotta più vicino ai genitori, viceversa per valori di b sufficientemente elevati i figli vengono generati più lontani dai genitori.

A prescindere dalla modalità scelta, il crossover risulta essere sorprendentemente facile da attuare, richiedendo niente di più che una generazione di numeri casuali, copie delle stringhe (nel caso binario) e qualche scambio parziale. Seppur randomizzato, esso genera quindi uno scambio di informazioni importanti donando all'algoritmo genetico gran parte della sua importanza. Nonostante la riproduzione.

- **Mutazione**

L'applicazione dell'operatore crossover può comportare il fatto che, dopo un certo numero di iterazioni, alcuni individui figli presentano lo stesso valore per il gene i-esimo dei genitori; ovvero che, a causa dell'assortimento genetico, vi sia la convergenza di uno, o anche più geni, allo stesso valore. Per risolvere questo problema, il GA prevede l'applicazione, successiva al crossover, di un ulteriore operatore: la mutazione.

L'operatore di mutazione (con il quale di completa una generazione) modella il fenomeno genetico della rara variazione di elementi del genotipo negli esseri viventi durante la riproduzione; una volta che due individui figli sono stati generati tramite il crossover, si estrae un numero casuale $q$ e si calcola la probabilità di mutazione $P_m$:

$$P_m = \begin{cases} P_{m1} = \dfrac{(P_{m1} - P_{m2})(f - f_{avg})}{f_{max} - f_{avg}} & f \geq f_{avg} \\ P_{m1} & f < f_{avg} \end{cases}$$

in cui $f_{max}$ è il valore di fitness più alto nella popolazione, $f_{avg}$ è il valore medio della fitness della popolazione e $f$ rappresenta il valore di idoneità alla mutazione più elevato tra due individui; $P_{m1}$ e $P_{m2}$ vengono impostati rispettivamente a 0,1 e 0,001.

Se $q \leq P_m$, il valore di alcuni bit, scelti anch'essi in modo casuale, dei nuovi individui vengono cambiati (nel caso binario quindi 1 diventa 0 e 0 diventa 1).

Come per l'operatore crossover, anche per la mutazione esistono diverse tipologie; nel metodo *Bit flip mutation* (usato nel caso di codifica binaria) si inverte il valore di un gene scelto in modo random da 0 a 1 o viceversa (figura 42).

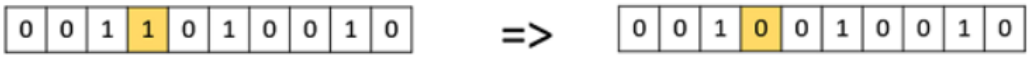



*Figura 42 - Esempio di mutazione ad un individuo della popolazione*

Inoltre, così come visto per l'operatore crossover nel caso di variabili reali o intere, anche per la mutazione la procedura utilizzata può essere differente e prevede di creare una soluzione $x$ in prossimità di una soluzione genitore $\bar{x}$; per fare questo si procede generando, prima di tutto, un numero random $s$:

$$s = (s_1)^p$$

dove $s_1$ (valore che governa la perturbazione della mutazione) è un numero random uniforme compreso tra 0 e 1, chiamato indice di mutazione. Il valore è assunto $p = p_{int}$ o $p = p_{real}$ in funzione dell'intero o della restrizione reale sulla variabile decisionale; in altre parole, per variabili decisionali intere il valore di $p$ è $p_{int}$ mentre per variabili decisionali reali $p$ è $p_{real}$. Una volta determinato s, la soluzione mutata è creata come segue:

$$x = \begin{cases} \bar{x} - s(\bar{x} - x^l) & t < r \\ \bar{x} + s(x^u - \bar{x}) & t \geq r \end{cases}$$

In cui $t = \frac{\bar{x} - x^l}{x^u - \bar{x}}$, $x^l$ e $x^u$ rispettivamente il limite inferiore e superiore della variabile decisionale e $r$ un numero casuale uniformemente distribuito tra 0 e 1.

- **Criteri di convergenza dell'algoritmo**

L'algoritmo genetico, come ampiamente discusso, si occupa di riprodurre le soluzioni fino ad ottenere la convergenza verso una soluzione ottima di tipo globale o locale. Applicando gli operatori di selezione, crossover e mutazione alla popolazione iniziale si ottiene la prima generazione; arrivati a questo punto il processo diventa iterativo e, ad ogni interazione, la nuova

popolazione è ottenuta considerando, come popolazione iniziale quella dell'iterazione precedente e la prole generata.

La domanda che allora diventa lecito porsi è: quando l'algoritmo deve stoppare il ciclo iterativo per restituire all'utente la soluzione finale "ottima" del problema? Purtroppo, non esiste una risposta univoca a questa domanda; esistono diversi criteri di arresto alcuni dei quali possono risultare, per il problema considerato, più performanti di altri. Di seguito solo elencati i principali criteri:

- *No change in fitness*: è il criterio più utilizzato e prevede di considerare l'individuo migliore (ad esempio quello caratterizzato da una OF più bassa) per ogni iterazione e stoppare l'algoritmo quando si giunge a convergenza (figura 43), ovvero quando, per un numero specifico di iterazioni, il valore di OF si stabilizza e l'individuo migliore risulta lo stesso, con uguale fitness o comunque molto simile.

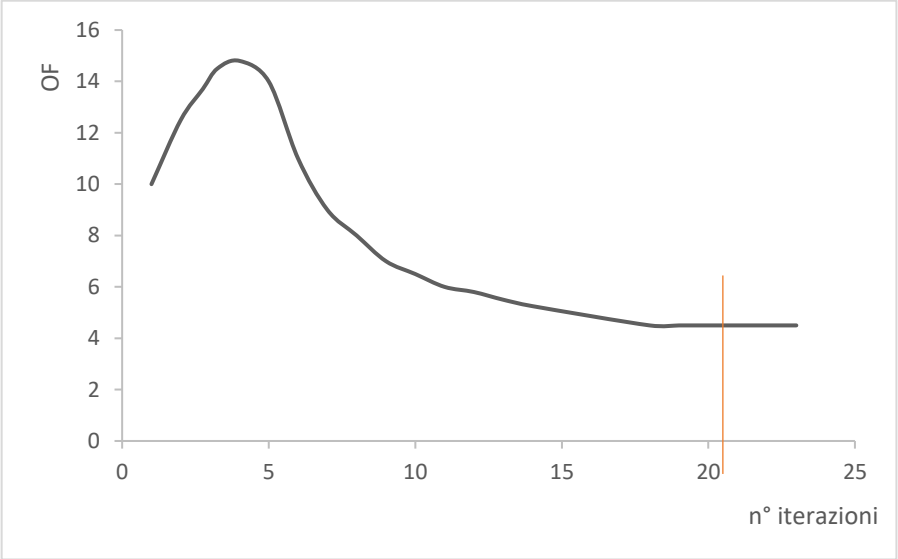



*Figura 43 - Criterio di convergenza della funzione obiettivo (OF)*

- *Maximum generations*: l'algoritmo si arresta quando viene raggiunto un fissato numero di iterazioni $K_{max}$ (generazioni);
- *Elapsed time*: l'algoritmo viene interrotto quando viene superato un tempo di esecuzione limite $T_{lim}$ fissato;
- *Stall generations*: l'algoritmo si ferma se non ci sono miglioramenti nella funzione obiettivo per una sequenza, fissata, di generazioni successive.

# 4. Ottimizzazione topologica, dimensionale e di forma: il caso studio

L'obiettivo che il lavoro di tesi si pone riguarda la realizzazione di una copertura reticolare che risulti ottimizzata, nel complesso e in ogni suo elemento asta costituente, in termini di massa e spostamento massimo (ottimizzazione topologica) oltre che in termini di sezione trasversale degli elementi (ottimizzazione dimensionale e di forma).

Un passaggio fondamentale per una progettazione ottimale è quello di eseguire una soluzione "iniziale" affidabile che riassuma tutti i criteri di progettazione da eseguire fino alla fase finale. Questa è una delle fasi più importanti della progettazione costruttiva, comunemente chiamata "*Conceptual Design*", che rappresenta proprio la fase preliminare del progetto architettonico in cui vengono inserite tutte le basi per ottenere il prodotto finale. In questa fase preliminare, esiste la decisione intrinseca del futuro, e la risposta performativa, della struttura progettata che, il più delle volte, dipende strettamente dalla forma (unita alle proprietà del materiale) che attribuiamo agli elementi strutturali.

Per il processo di ottimizzazione, nel lavoro di tesi è stato implementato uno specifico algoritmo (Figura 44), che il capitolo andrà ad analizzare in ogni suo step per discuterne infine i risultati ottenuti, utilizzando i seguenti strumenti:

1) *Grasshopper*: per la parametrizzazione delle geometrie tramite programmazione visuale;
2) *Karamba 3D*: plug-in utilizzato per effettuare l'analisi FEM;
3) *Octupus*: algoritmo di stima per l'ottimizzazione multi-obiettivo.

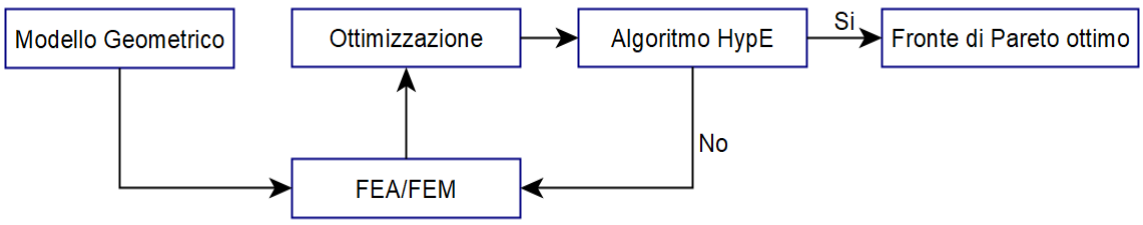



*Figura 44 - Diagramma di flusso dell'algoritmo implementato*

## 4.1. Sviluppo di una copertura parametrica (*frame structure*) assistita da algoritmo

Nella prima parte del lavoro svolto è stato implementato un algoritmo che permettesse di realizzare la geometria della copertura in esame in modo tale che essa risulti variabile in ogni sua parte ovvero modificabile sia in termini di lunghezza delle campate, in direzione x e in direzione y, sia in termini di numero degli elementi ($n_x$ e $n_y$) rispettivamente presenti lungo lo sviluppo delle campate $L_x$ e $L_y$.

L'idea di sviluppare una struttura che presenti tali caratteristiche, coerentemente con il concetto di design parametrico, è basata sul fatto che, scelti i parametri di design da modificare opportunamente, è possibile ottenere una struttura "elastica" capace di aggiornarsi in tempo reale al variare dei parametri stessi, rimanendo comunque coerente con quanto stabilito in partenza; in tal senso, una volta ottenuto il numero ottimale di elementi $n_x$ e $n_y$, è possibile usare la stessa struttura milioni di volte semplicemente variandone i parametri di luce o viceversa.

Per fare ciò, prima di tutto l'attenzione è stata posta sul singolo elemento trave (*beam*) e, tra le varie tipologie reticolari, o *truss*, possibili (alcune delle quali illustrate in figura 45), si è scelto di modellare una *"Howe truss"* caratterizzata de aste diagonali che si mantengono parallele ma specchiate rispetto all'asse di simmetria, lo stesso asse in corrispondenza del quale vi è la convergenza di due aste nel nodo superiore dell'elemento.

Inoltre, si è deciso di utilizzare la stessa tipologia di trave sia lungo la lunghezza $L_x$ sia lungo $L_y$, facendo in modo che l'elemento trave risulti sempre simmetrico e quindi vincolato a un numero di elementi, $n_x$ o $n_y$, che, seppur variabile, assuma sempre un valore pari.

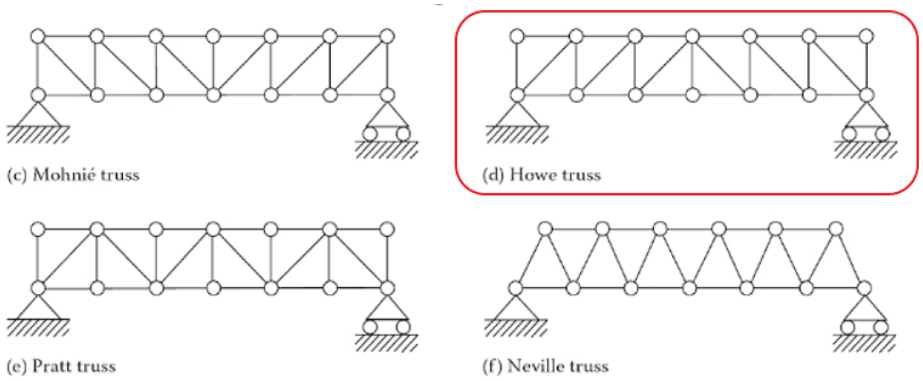



*Figura 45 – Tipologie di travi reticolari (truss beams)*

La costruzione del modello geometrico è stata realizzata utilizzando il software parametrico Grasshopper, ampiamente introdotto nel paragrafo 3.2.1, che nasce, principalmente, come strumento di programmazione visiva offrendo la possibilità di utilizzare una serie di oggetti (componenti e parametri) che permettono di svolgere specifiche azioni e che, opportunamente collegati tra loro, costituiscono un algoritmo.

Nonostante ciò, la prima parte relativa alla realizzazione del modello geometrico della copertura, nonché la costruzione dei punti (o nodi) e delle linee di collegamento tra essi, è stata implementata tramite codice; Grasshopper offre infatti anche la possibilità di poter utilizzare degli opportuni oggetti di *script* all'interno dei quali, una volta caricata la libreria di Rhino, possono essere implementate delle righe di codice in un linguaggio coerente con l'oggetto scelto (*Python, C++*, *Visual Basic*) grazie alle quali possono essere richiamati tutti i parametri e le componenti del software, svolgendo esattamente le stesse funzioni, tramite codice e non tramite oggetto visivo.

Nel nostro caso è stato scelto di utilizzare l'oggetto <Python>, che è stato trascinato sul canvas dell'interfaccia del software, all'interno del quale è stato poi implementato il codice geometrico (interamente riportato in Appendice 1); in tal modo nel flusso dell'algoritmo è visibile solo il componente, mentre il codice è accessibile soltanto tramite un doppio click sull'oggetto stesso.

Il componente <Python> (figura 46), così come tutti i componenti disponibili in Grasshopper, una volta trascinato sul canvas, per poter funzionare richiede la definizione di parametri di input (*x, y*) che, una volta collegati, vengono acquisiti come parametri dal codice implementato al suo interno; il codice viene poi elaborato ed eseguito all'interno dell'oggetto permettendo di ottenere così i valori di outputs (*out, a*).


*Figura 46 – Componente per lo script del codice in Python*

A differenza di tutti gli oggetti presenti sul software, in cui gli input da passare devono essere necessariamente quelli richiesti dall' oggetto specifico, e devono inoltre essere passati nel modo richiesto (lista, vettore, linea, curva, ecc) al fine di ottenere degli outpust specifici, il componente <$Phython$> è molto "flessibile" e permette di passare come input un numero e dei

tipi di parametri scelti dal progettista in funzione della logica con cui egli ha implementato il codice di script; di conseguenza anche gli outputs ottenuti possono essere scelti opportunamente in funzione di ciò che si vuole ottenere.

Al fine di ottenere il modello geometrico della copertura, sono stati scegli i seguenti parametri di design:

- $L_x$: lunghezza della campata in direzione $x$;
- $L_y$: lunghezza della campata in direzione $y$;
- $nX\_segments$: numero di elementi lungo lo sviluppo $L_x$ ($n_x$);
- $nY\_segments$: numero di elementi lungo lo sviluppo $L_y$ ($n_y$);
- $H\_beam$: altezza della copertura (H).

Gli outputs scelti in Python, in modo tale da essere utilizzati come parametri di input per le successive parti dell'algoritmo, sono invece le linee costituenti tutta la geometria della copertura:

- $o\_VLines$: contenente tutte le linee che rappresentano gli elementi verticali;
- $o\_OlowLines$: contenente tutte le linee costituenti le aste orizzontali inferiori della copertura;
- $o\_OuppLines$: contenente tutte le linee costituenti le aste superiori della copertura;
- $o\_Diagonals$: contenente tutte le linee che rappresentano le aste diagonali.

In figura 47, si riporta quanto detto sul componente <Python>, sui parametri di input passati come <Number slider >, ovvero variabili, e sui parametri di output, ricavati una volta eseguito il codice all'intero dell'oggetto.

Si nota che, nel componente <A/B> è stato inserito il rapporto tra due numeri passati come input, rispettivamente A e B, per ottenere come output il risultato R; in questo caso sono stati inseriti due componenti di divisione che eseguono i rapporti ($L_x$ / $nX\_segments$) e ($L_y$ / $nY\_segments$) fornendo rispettivamente come risultati (mostrati dagli oggetti <Panel>) l'interasse tra gli elementi in direzione x ($Delta\_iX$) e l'interasse tra gli elementi in direzione y ($Delta\_iY$); essi sono stati forniti come input al codice insieme ai parametri variabili sopra citati.

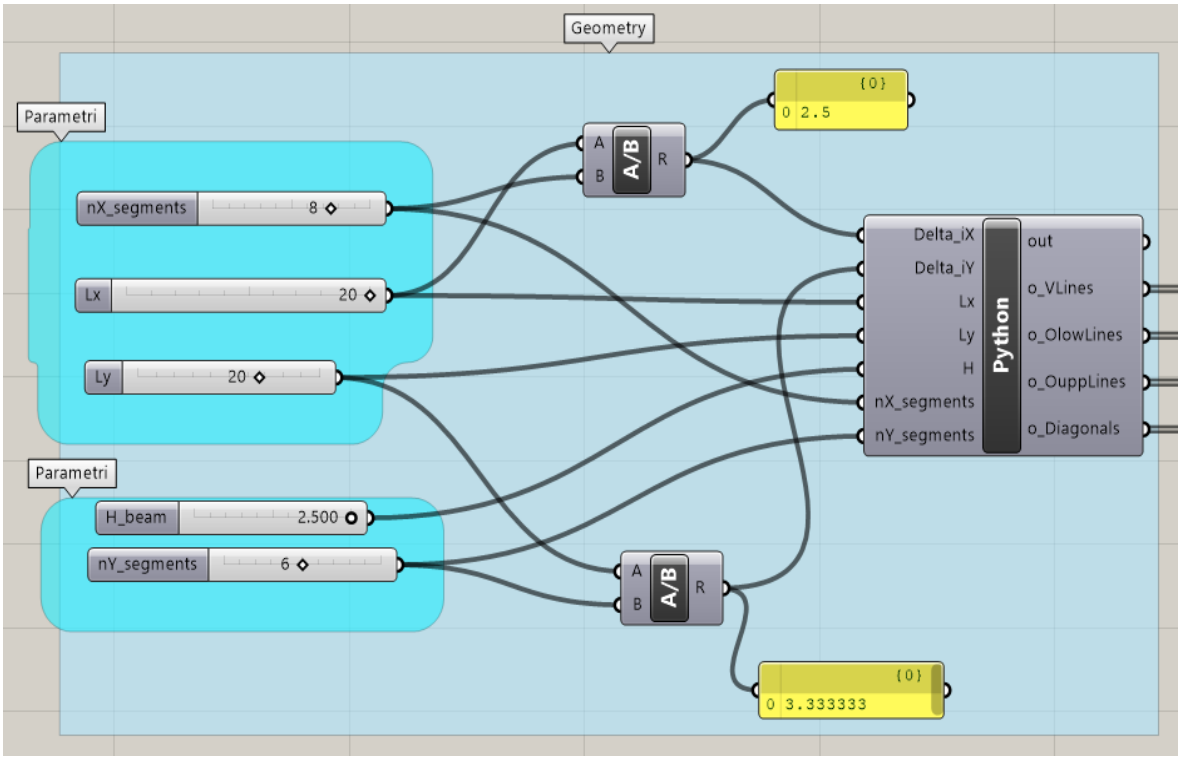



*Figura 47 – Algoritmo per la modellazione geometrica della copertura*

Infine, si osserva che tutti i parametri di input e output hanno come unità di misura il metro [m].

Senza entrare troppo nel dettaglio del codice implementato in Python (vedi Appendice 1), la logica seguita è stata quella di realizzare, prima di tutto, il singolo elemento trave sia in direzione x che in direzione y e, successivamente, traslare tali elementi di una quantità fissa, pari a $(2 * Delta\_iY)$ per generare le travi parallele a quella creata in x lungo lo sviluppo $L_y$ e pari a $(2 * Delta\_iX)$ per generare, invece, le travi parallele a quella in y lungo lo sviluppo $L_x$, ottenendo il modello geometrico finale in funzione dei valori dei parametri scelti.

Per la modellazione delle singole travi sono stati richiamate, da codice, le componenti:

- < Pt >: componente < Construct Point > che richiede in ingresso le coordinate di un punto;
- < $Ln$ >: componente < Line > che richiede in ingresso i due punti da collegare tramite l'oggetto linea;

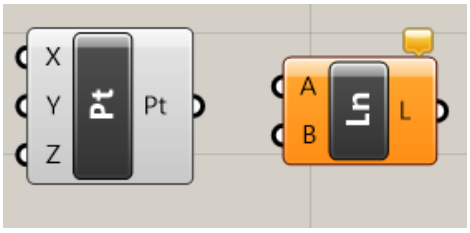



*Figura 48 – Componenti per la generazione di punti e linee*

Generati i punti e collegati opportunamente, sono state modellate la trave in direzione x e la trave in direzione y, figura 49 e figura 50 rispettivamente.

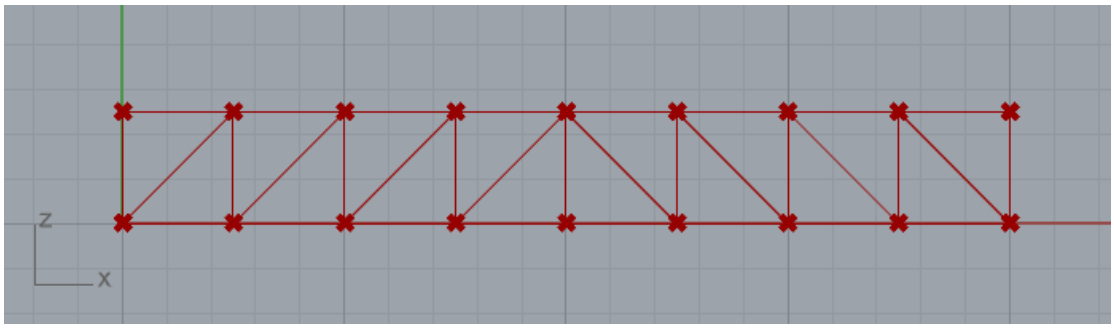



*Figura 49 - modello geometrico della trave in direzione x con:* $Lx = 20\ m, nX\_segments = 8, Delta\_iX = 2{,}5\ m, H = 2{,}5\ m$

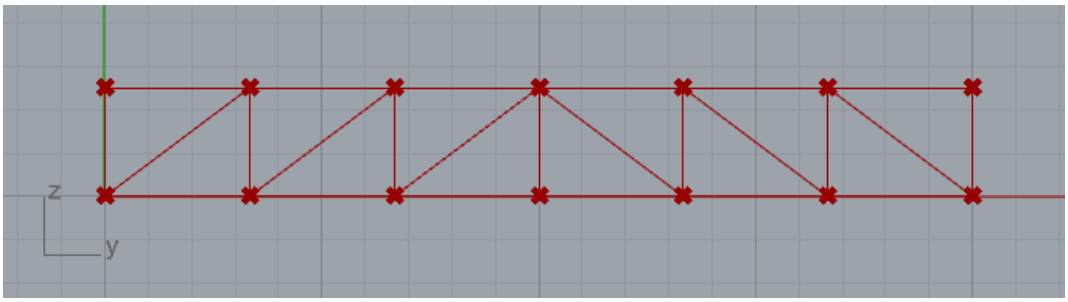



*Figura 50 - modello geometrico della trave in direzione y con:* $Ly = 20\ m, nY\_segments = 6, Delta\_iX = 3{,}333\ m, H = 2{,}5\ m$

A partire dalle singole travi, tramite il componente <Move>, che richiede in ingresso la geometria da traslare (G) e il vettore di traslazione (T), sono state realizzate le altre travi in direzione x e y ed è stato costruito il modello geometrico della copertura in funzione dei parametri di design.

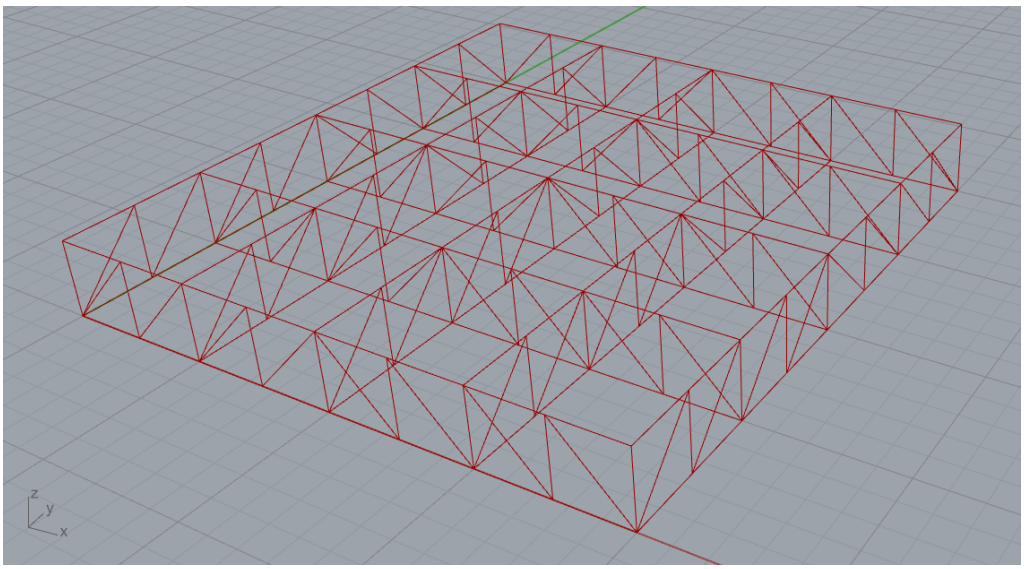


*Figura 51 – Modello geometrico della copertura con:* $Lx = 20m, Ly = 20m,\ nX_segments = 8,\ nY_segments = 6,$ $Delta\_iX = 2{,}5\ m, Delta\_iY = 3{,}333, H = 2{,}5\ m$

Una volta costruiti gli oggetti geometrici del modello, al fine di poter fare delle analisi FEM e studiare il comportamento della struttura, lo step successivo è stato quello di trasformare le linee in elementi travi (o *beam*); riconoscendole come tali, il software è infatti in grado di associare ad esse tutte le caratteristiche e le proprietà beam utili per effettuarne le analisi strutturali.

## 4.2. Analisi FEM utilizzando Karamba3D

Per fare questo, Grasshopper ha a disposizione uno specifico plug-in chiamato KARAMBA; si tratta infatti di un software parametrico di analisi strutturale agli elementi finiti che, rispetto ai classici modelli FEM, risulta di facile utilizzo, soprattutto per i non esperti di programmazione parametrica, poiché molto intuitivo ed estremamente adattabile alle esigenze di architetti e ingegneri nelle fasi iniziali dell'iter progettuale. Inoltre, la sua capacità di essere gestito in maniera interattiva, lavorando nell'ambiente parametrico di Grasshopper, lo rende pratico ed efficiente grazie al fatto che consente di combinare facilmente l'analisi strutturale ai modelli geometrici parametrizzati tramite algoritmi di ottimizzazione; in tal senso è sufficiente trasformare il modello geometrico parametrizzato in un modello discreto mesh, definirne le proprietà strutturali (sezioni, materiali, vincoli e carichi), assemblare il tutto (procedimento tipico di un analisi agli elementi finiti, FEM) e, per completare l'analisi strutturale, analizzare il modello tramite un *solver*.

Questo tipo di simulazione strutturale statica rappresenta quindi oggetti fisici come insieme di componenti discreti o "elementi" (che possono, a loro volta, essere rappresentati come elementi monodimensionali, tipo linee, bidimensionali o tridimensionali), i quali vengono utilizzati dalla simulazione FEM per calcolare il modo in cui le forze, provenienti da un carico applicato all'oggetto, si ridistribuiscono sull'oggetto stesso; sulla base del calcolo effettuato e delle proprietà del materiale di ogni elemento, è possibile prevedere lo spostamento che subirà la struttura sotto il carico e, di conseguenza, la sollecitazione a cui sarà soggetto ogni elemento che la costituisce.

Di seguito, viene descritto ogni singolo step dell'algoritmo implementato per poter eseguire l'analisi strutturale utilizzando il plug-in Karamba3D.

- Assemblaggio del modello

L'assemblaggio di un modello è un passaggio indispensabile per poter eseguire le analisi strutturali e prevede di convertire il modello geometrico, realizzato in Grasshopper, in un vero e proprio modello di simulazione. Tale operazione è possibile grazie alla presenza in Karamba3D di uno specifico componente <*Assemble* >, situato sotto la voce "Model" della barra degli strumenti, che prende in ingresso tutte le informazioni indispensabili e produce un modello pronto per eseguire l'analisi strutturale.

In figura 52, vengono indicati, tramite frecce rosse, gli input che bisogna necessariamente fornire all'oggetto in modo tale da renderlo funzionante; per i restanti valori di input (che rappresentano funzionalità e opzioni aggiuntive per l'analisi) non è invece richiesta una specifica, al fine di eseguire la simulazione, poiché presentano già dei valori predefiniti; in ogni caso possono essere facilmente personalizzati dal designer, a secondo dell'accuratezza dei risultati finali che egli desidera ottenere.

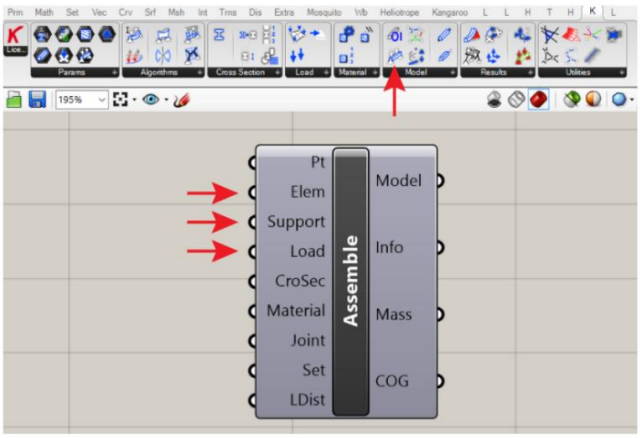



*Figura 52 – Valori di input del componente <Assemble > per la realizzazione di un modello di simulazione*

- Definizione degli elementi trave (*beam*)

Come già accennato, prima di tutto, ogni elemento linea (asta della struttura reticolare) del modello geometrico è stato trasformato in elemento beam tramite un opportuno componente <LinetoBeam >, sempre disponibile in Karamba3D. Tale oggetto prende infatti in ingresso la geometria delle linee da Grasshopper e la converte in elementi del fascio Karamba che possono, a questo punto, essere inseriti come input per il componente <*Assemble* >.

Prima della conversione delle linee in elementi trave è stato però effettuato un controllo sull'eventuale presenza di geometrie duplicate; Karamba fornisce infatti una serie di oggetti per il pre-processing della geometria della linea prima che essa venga convertita in trave, tra cui il nodo <RDLines > che opera rimuovendo gli elementi duplicati. Le linee duplicate sono infatti difficili da individuare nel modello ma incidono fortemente sulla qualità delle analisi effettuate.

Inoltre, la geometria delle linee (nessuna delle quali presenta valori duplicati) è stata esplosa in segmenti più piccoli tramite il nodo <Explode> prima di essere convertita in beam e quindi passata al <LinetoBeam>.

Così definiti, gli elementi travi sono infine stati collegati al nodo per l'assemblaggio.

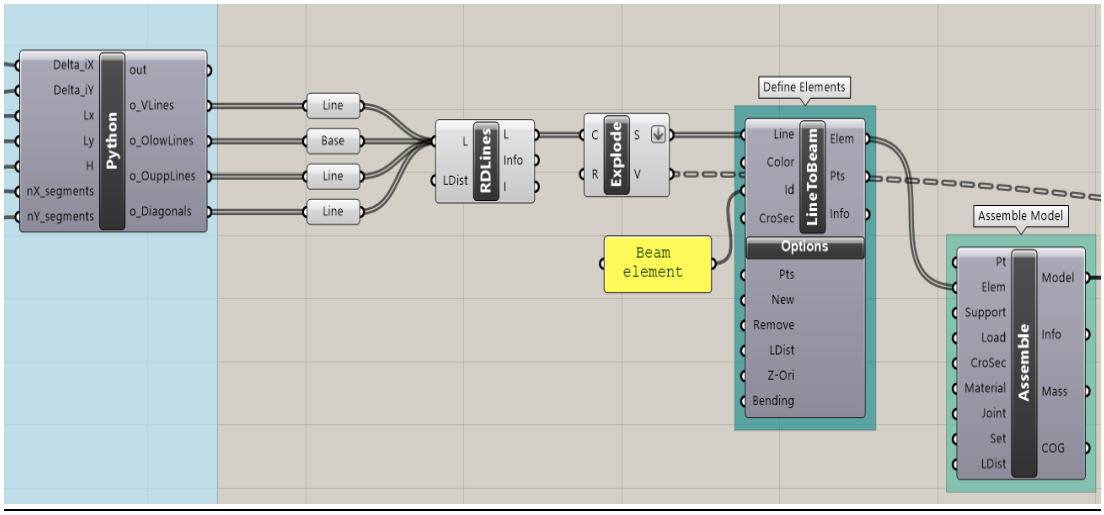



*Figura 53 – Algoritmo per la conversione degli elementi linea (del modello geometrico) in elementi trave*

- Definizione del materiale e della sezione trasversale (da ottimizzare)

La specifica delle proprietà strutturali come il materiale e la tipologia della sezione trasversale, come già detto, non è necessaria al fine dell'esecuzione delle analisi dato che presenta già dei

valori preimpostati in Karamba; in ogni caso andarli a definire opportunamente può avere un effetto importante sui risultati finali.

La definizione del materiale viene effettuata tramite un opportuno componente < MatSelect > (disponibile nell'area "Material" della barra degli strumenti), grazie al quale è possibile sia selezionare un materiale predefinito dalla libreria di Karamba (utilizzando il menù a tendina nella parte inferiore del componente) sia assegnare delle diverse proprietà del materiale, anche a elementi diversi del modello, tramite le opzioni di Input tra cui "Elems|Id" che richiede l'ID degli elementi a cui andare ad applicare lo specifico materiale.

Per quanto riguarda la definizione della sezione trasversale di un elemento trave, anche in questo caso esiste uno specifico componente chiamato < CrosSecRSelect>, situato nell'area "Cross Section" della barra degli strumenti, che consente di specificare le altre due forme dimensionali degli elementi trave.

Così come per il materiale, è possibile selezionare un tipo di sezione della libreria di Karamba utilizzando il menù nella parte bassa del componente e passando opportuni valori di input specifici per quella data sezione, oppure personalizzare la sezione da assegnare.

Come mostrato in figura 54, nel caso in esame sono state definite tre differenti sezioni della normativa EU (da sottoporre ad un processo di ottimizzazione): sezione circolare cava, sezione rettangolare cava e sezione ad I (IPE); in tutti questi casi, il componente ha richiesto in ingresso il massimo spessore della sezione da definire (maxW), il suo massimo valore in altezza (maxH) e il materiale da attribuire alla sezione trasversale da creare (Material).

Come parametri di massimo spessore e massima altezza, sono stati assegnati dei range di valori compresi tra [10 cm – 80 cm] passati a tutte le tre tipologie di sezione trasversale; inoltre, come è possibile vedere, alle sezioni è stata anche assegnata la stessa tipologia di materiale. Quest'ultimo è stato selezionato dalla libreria, dal momento in cui si è deciso di utilizzare un acciaio "*Steel*" S235.

A questo punto, gli outputs generati dalla definizione delle sezioni trasversali ("CroSec"), che comprendono già l'assegnazione del materiale, sono stati collegati come parametri di input al <LinetoBeam>; oltre ai parametri di tipo "linea" esso richiede infatti anche altri parametri in ingresso, tra cui la sezione dell'elemento trave che, una volta definita, viene automaticamente assegnata dall'algoritmo come proprietà degli elementi passati.

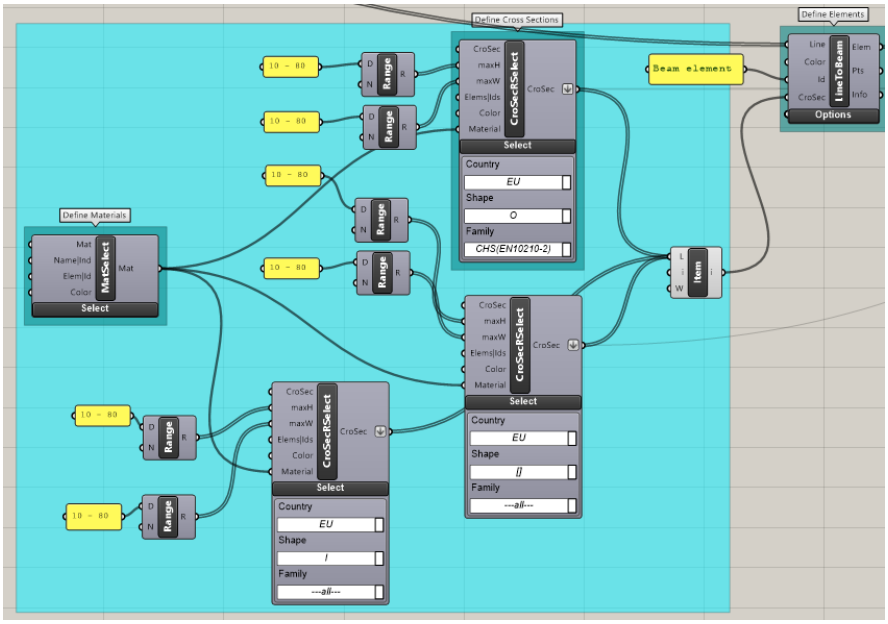



*Figura 54 – Algoritmo per la definizione e l'assegnazione di materiali e sezioni trasversali agli elementi beam*

➢ <u>Definizione dei carichi</u>

Tra le cose da definire per ottenere un valido strumento di simulazione, di estrema importanza risulta la definizione e l'assegnazione di uno o più carichi al modello al fine di poter studiare il comportamento strutturale.

Il componente utilizzo, situato nell'area "Load" della barra degli strumenti, è un nodo multi-uso e permette di specificare diverse tipologie di carico da assegnare al modello. Il tipo di carico da applicare può infatti essere selezionato mediante lo specifico menù a tendina, che si trova nella parte inferiore del componente stesso, e, in funzione della tipologia di carico scelto, è necessario fornire al nodo degli input diversi in base alle informazioni richieste.

Nel caso di studio, al modello sono stati applicati due diversi tipi di carico:

- *Peso proprio*: è il tipo da carico più semplice da applicare dal momento che non richiede nessun valore aggiuntivo di input. Per definirlo è stato utilizzato il componente < Loads > selezionando dal menù 'Type of Load' la voce " Gravity" che calcola la forza dovuta al peso proprio di ogni elemento del modello;

- *Carico variabile*: per definirlo è stato utilizzato sempre il componente < Loads >, ma stavolta è stata selezionata la voce " Uniform Line" rappresentate un carico uniformemente distribuito ($[F/L]$) sulla struttura che, a sua volta richiede in ingresso un parametro vettoriale. Dal momento che si è preso in considerazione un carico

variabile neve, trattandosi di una copertura, di $100\ K_g/m^2 (1\ KN/m^2)$ è stato considerato un vettore unitario in Z con un fattore moltiplicativo pari alla lunghezza della campata (ovvero 20 m), passato tramite <number slider >, in modo da ottenere un carico lineare di $20\ KN/m$.

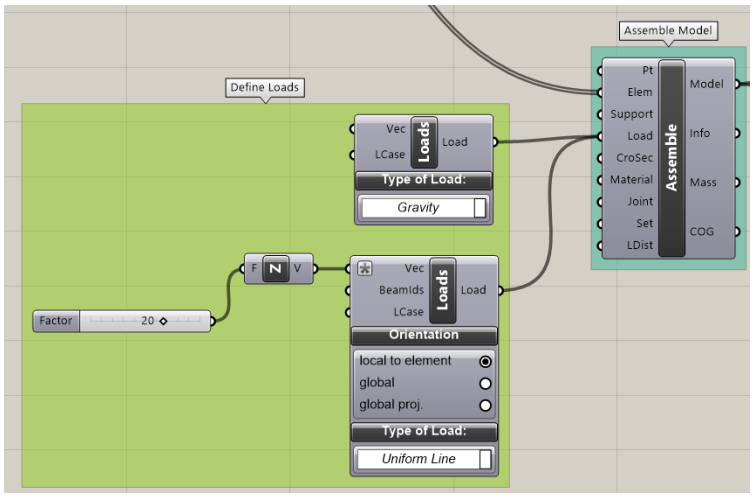



*Figura 55 – Algoritmo per la definizione e l'applicazione dei carichi al modello di simulazione*

Definiti i carichi da applicare, gli outputs dei componenti "Loads" sono stati passati come ulteriori inputs al nodo <$Assemble$ >.

- Definizione di support e joints

Una volta creati gli elementi del modello da analizzare, è necessario andare a specificare il modo in cui la copertura è vincolata. La definizione dei supporti viene effettuata tramite un ulteriore componente chiamato <Support >, nell'area "Model" presente tra gli strumenti di Karamba, che accetta una serie di punti da Grasshopper e li converte in punti fissi nel modello.

Al componente devono essere passate le coordinate dei punti da vincolare ("Pos|Ind") ed è possibile scegliere la tipologia di supporto da applicare andando a selezionare quali dei 6 gradi di libertà fissare (3 traslazioni:$T_x, T_y, T_z$, e 3 rotazioni: $R_x, R_y, R_z$).

Nel caso in esame sono state create quattro cerniere bidirezionali, che non consentono alcun tipo di traslazione lungo gli assi ($T_x = T_y = T_z = 0$), la cui posizione è stata ottimizzata utilizzando uno specifico componente, chiamato <Gene Pool> (letteralmente piscina genetica) contenente una collezione di valori (o geni), nel nostro caso 4, delle coordinate dei punti di base e le fa variare in modo randomico trovandone la posizione migliore affinché sia la massa che lo spostamento massimo risultino minimizzati.

L'output del <Gene Pool > è stato collegato come input al componente dei supporti fornendone, una volta eseguita l'ottimizzazione, la loro posizione finale (discussa nel paragrafo 4.4).

Per quanto riguarda la definizione dei *joints*, ovvero gli elementi di connessione tra le travi, è stato utilizzato il componente <Joint – Agent> che, andando ad individuare i nodi tra gli elementi beam, consente di vincolarli opportunamente; nel nostro caso si è scelto di fissare tutti i gradi di libertà in modo da irrigidire la struttura. In entrambi i casi, gli outputs dei componenti sono stati infine passati come input ad <*Assemble*>.

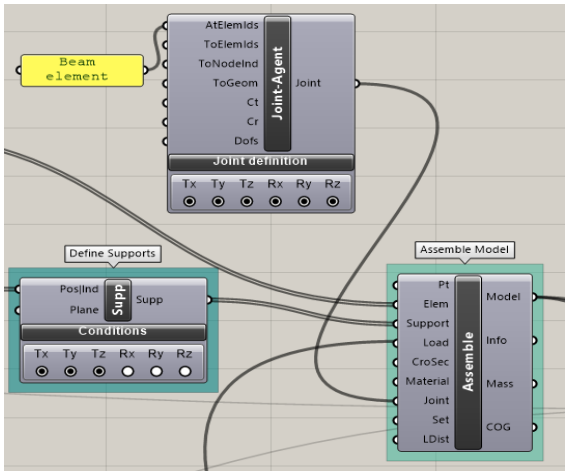



*Figura 56 -Algoritmo per la definizione e l'assegnazione di supporti e joints*

- Calcolo del modello

Definite tutte le proprietà strutturali e assegnati i vincoli e i carichi alla struttura, il tutto, come già visto ad ogni step, è stato assemblato con il componente <*Assemble* >, che ha generato come output ("Model") il modello finale da poter sottoporre all' analisi FEM e ai processi di ottimizzazione.

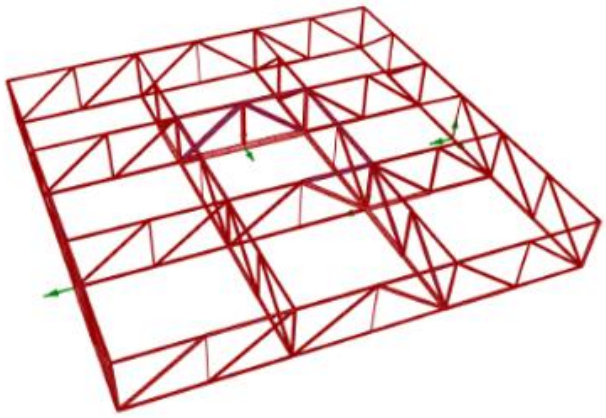


*Figura 57 – Algoritmo per la realizzazione del modello di simulazione*

Una volta assemblato il modello, è stato possibile eseguire il calcolo strutturale selezionando il componente di analisi desiderato dall'area "Algorithm" della barra degli strumenti; è infatti possibile scegliere tra una serie di componenti di analisi in funzione del tipo di calcolo strutturale che si vuole eseguire.

Nel caso esaminato, tramite il componente <Analyze> è stato possibile calcolare le deflessioni del modello, una volta lanciata l'ottimizzazione, usando la teoria del primo ordine per piccole deflessioni (discusso nel paragrafo 4.4 - Risultati).

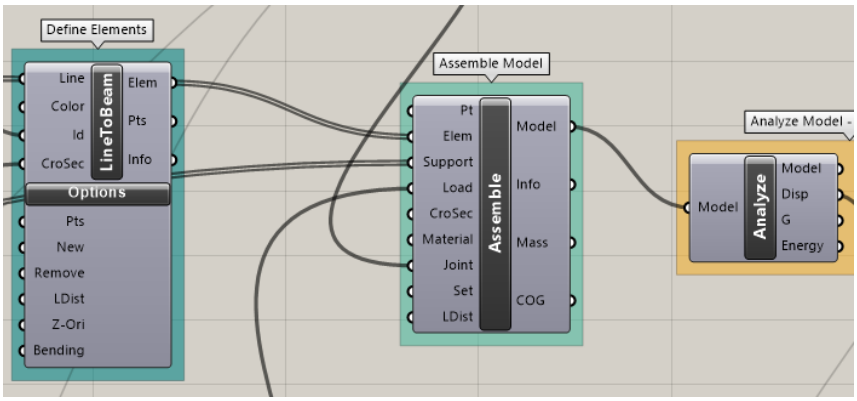



*Figura 58 – Algoritmo per l'analisi del modello*

## 4.3. Problema di ottimizzazione: il modello numerico

Come già discusso, l'obiettivo del lavoro di tesi è quello di sviluppare una copertura reticolare, su cui effettuare un'analisi agli elementi finiti, che risulti ottimizzata, in termini di peso e di massimo spostamento subito da ogni elemento della struttura per effetto dei carichi applicati, andando a variare non solo il numero degli elementi, costituenti le travi lungo lo sviluppo delle campate $L_x$ e $L_y$ ($n_x$, $n_y$), ma anche la posizione dei supporti della struttura.

Affiancato ad un processo di questo tipo (ottimizzazione topologica) sono state anche implementate un'ottimizzazione di tipo dimensionale, o "*size*" e un'ottimizzazione di forma, o "*shape*", grazie alle quali è stato possibile, in contemporaneo, andare a ottimizzare le sezioni trasversali degli elementi trave in termini di spessore scegliendo tra tre diverse categorie di sezioni possibili.

Si sottolinea che in tutti e tre i casi si tratta di un'ottimizzazione vincolata dal momento in cui è stato inserito un vincolo in termini di spostamento massimo:

$$\delta_{max} \leq \frac{1}{150} \cdot L_{Tot} = 13\ cm$$

dove $L_{Tot}$ rappresenta la luce massima della campata pari a 20 m.

Le ottimizzazioni di tipo *size e shape* sono state realizzate utilizzando uno specifico componente di Karamba3D chiamato <OptiCrosSec > che tra i principali parametri di input richiede:

- il modello per il quale si vuole ottimizzare la sezione trasversale "Model";
- l'elenco delle sezioni trasversali, fornite già delle informazioni relative ai materiali, per l'ottimizzazione "CroSecs";

Nel caso studio, al componente è stato passato il modello ottenuto dall'assemblaggio e, come già visto, sono state inserite le 3 categorie di sezioni trasversali definite precedentemente tramite il componente < CrosSecRSelect >.

Prendendo questi parametri in ingresso, il componente di ottimizzazione sceglie la sezione migliore e la assegna a tutti gli elementi trave del modello variandola in spessore.

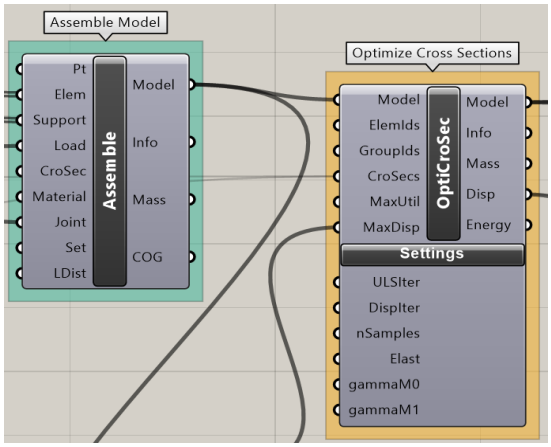



*Figura 59 - Componente per l'ottimizzazione della sezione trasversale*

Per quanto riguarda invece l'ottimizzazione topologica, all'interno dell'algoritmo è stato impostato un problema di ottimizzazione multi-obiettivo (ovvero un problema caratterizzato da più funzioni che devono essere ottimizzate simultaneamente) esprimibile come segue:

$$min \begin{cases} f(x) \\ g(x) \end{cases}$$

dove la funzione $f(x)$ rappresenta il peso, o la massa, della struttura e $g(x)$ rappresenta invece lo spostamento massimo, rispettivamente da andare a minimizzare.

L'opportuna valutazione di questi due aspetti risulta estremamente importante nell'ambito delle analisi strutturali; la minimizzazione della massa, o del peso, del modello rende infatti possibile la progettazione di una struttura che risulta più piccola e leggera possibile, capace di garantire in ogni caso il raggiungimento delle prestazioni minimi richieste, la minimizzazione dello spostamento massimo del modello, è invece volta alla limitazione dei valori di spostamento subiti dai singoli elementi, e quindi complessivamente dall'intera struttura, sottoposti all'azione dei carichi esterni.

I parametri scelti da variare e modificare opportunamente, all'interno del processo di ottimizzazione topologica, sono:

- Posizione dei vincoli di supporto;
- $n_x$;
- $n_y$
- interassi degli elementi lungo x e y, come diretta conseguenza della variazione di $n_x$ e $n_y$.

### 4.3.1. SPEA-2 e HypE Reduction Algorithm: risoluzione del problema di ottimizzazione tramite l'uso di algoritmi genetici evolutivi

Per la risoluzione dei problemi di ottimizzazione multi-obiettivo, all'interno di Grasshopper è possibile utilizzare un ulteriore plug-in, chiamato "*Octupus*", specializzato appunto nell''ottimizzazione evolutiva multi-obiettivo, che consente la ricerca di molti obiettivi contemporaneamente, producendo una gamma di soluzioni ottimizzate tra gli estremi di ogni obiettivo basandosi sull'uso di due tipologie di algoritmi:

- SPEA2 ("*Strength Pareto Evolutionary Algorithm 2*")
- $HypE\ Reduction$ ("$Hypervolume\ Reduction\ Algorithm$")

I due algoritmi svolgono la medesima funziona e, in generale, la scelta del prediligere l'utilizzo dell'uno rispetto all'altro si basa solo sul numero degli obiettivi da andare ad ottimizzare; se questi risultano inferiori a cinque viene utilizzato l'algoritmo Hype, viceversa l'SPEA2.

In entrambi i casi, si tratta di una tipologia di algoritmi genetici (discussi nel paragrafo 3.3.3) che ricercano le condizioni di ottimo simulando il processo di evoluzione degli organismi nell'ambiente naturale; operando sull'intera popolazione contemporaneamente, essi sono infatti in grado di cercare, in una singola esecuzione dell'algoritmo, più soluzioni in parallelo

rivelandosi fortemente versatili per affrontare problemi complessi di ottimizzazione multi-obiettivo che risulterebbero difficili da risolvere con metodi di ottimizzazione tradizionali.

Di seguito verranno brevemente introdotti i due algoritmi, descrivendone il principio base di funzionamento.

- Teoria di base degli algoritmi SPEA2

SPEA2 è un efficiente algoritmo genetico di ottimizzazione multi-obiettivo, proposto da Zitzler et al, come versione migliorata di "$Strength\ Pareto\ Evolutionary\ Algorithm$" (SPEA), che si basa sul concetto di dominio Pareto, per le operazioni di selezione e assegnazione dei valori di fitness, incorporando anche informazioni sulla densità.

Per descrivere il concetto di dominio di Pareto è necessario prima introdurre il concetto di dominanza che è alla base di molti algoritmi per risolvere i problemi multi-obiettivo ed è fondamentale per riconoscere l'insieme delle soluzioni ottime.

Dato un insieme P di n soluzioni, è possibile considerare una soluzione $x_i$ e confrontarla con le altre rimanenti $x_j$ , con j = 1, 2, …,n e j ≠ i ottenendo, come risultato di tale confronto, n-1 risultati che possono valere *dominato* o *non-dominato*. A questo punto la soluzione $x_i$ diventa *non-dominata* se tutti gli n-1 confronti hanno dato *non-dominato*, viceversa, è dominata se almeno un confronto è stato *dominato*.

Procedendo con il confronto di tutte le soluzioni $x_j$ (con j = 1, 2, …,n) si può dividere *P* in due sottoinsiemi chiamati insieme delle soluzioni *non-dominate* e insieme delle soluzioni *dominate* seguendo questa definizione: "*Dato un insieme di soluzioni P, l'insieme delle soluzioni non-dominate P' è formato da tutte quelle soluzioni che non sono dominate da nessun altro membro dell'insieme P*". L'insieme appena costruito è però formato dalle soluzioni che sono le migliori all'interno di P, mentre lo scopo degli algoritmi, che risolvono problemi multi-obiettivo, è quello di trovare le soluzioni che sono migliori in tutto lo spazio di ricerca S.

Per questo motivo viene definito un altro insieme, chiamato "insieme delle soluzioni ottime di Pareto", che contiene le soluzioni ottime del problema multi-obiettivo: "*L'insieme non-dominato su tutto l'intero spazio di ricerca ammissibile S viene chiamato l'insieme ottimo di Pareto*".

In altre parole, l'insieme ottimo di Pareto non è altro che un particolare insieme di non dominanza ottenuto però considerando tutto lo spazio di ricerca S.

Un algoritmo genetico basato su questo concetto non ha quindi il solo scopo di cercare soluzioni ottime ma anche quello di fare in modo che queste rappresentino al meglio l'insieme ottimo di Pareto (in genere di dimensione infinita).

In figura 61 è illustrato il generico flusso di un algoritmo genetico SPEA-2:

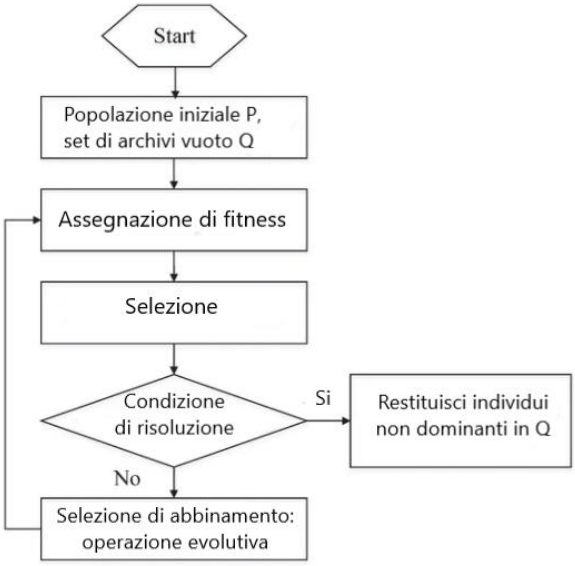



*Figura 60 – Descrizione tramite diagramma di flusso di un algoritmo SPEA-2*

- Teoria di base degli algoritmi HypE

L'HypE è uno dei più comuni algoritmi di stima dell'*ipervolume* (Zitzler e Thiele, 1998a, 1999), uno specifico indicatore che si basa sul concetto di dominanza impiegato come misura della qualità di un insieme di soluzioni dell'ottimizzazione multi-obiettivo; si tratta infatti dell'unico indicatore di qualità noto per essere completamente sensibile alla dominanza di Pareto, una proprietà, come già detto, particolarmente desiderabile quando sono coinvolte funzioni con più obiettivi.

Un problema di ottimizzazione multi-obiettivo può essere, in genere, formulato come segue:

$$min\, f(x) = [f_1(x), f_2(x), f_3(x), \dots f_d(x)]$$

dove $x = [x_1, x_2, x_3, \dots x_m]$ è il vettore delle variabili decisionali, *f(x)* è il vettore delle funzioni obiettivo (relativo a *x*) e *d* è il numero delle funzioni obiettivo. La soluzione ideale del problema di ottimizzazione sarebbe quella che presenta il minor valore di tutte le funzioni obiettivo ma, poiché nella maggior parte dei casi questa soluzione non esiste, la definizione di ottimo deve essere estesa introducendo il concetto di soluzione dominata.

Come già detto, una soluzione si dice non dominata se non esiste nessuna soluzione che la domini e l'insieme delle soluzioni non dominate rappresenta l'insieme delle ottimali secondo Pareto mentre il corrispondente insieme di punti nello spazio (che corrispondono a soluzioni di compromesso, tra le quali il designer può scegliere) delle funzioni obiettivo definisce il fronte di Pareto.

Il fronte Pareto - Ottimale nei problemi di ottimizzazione multi-obiettivo (MOOP) rappresenta quindi un insieme di soluzioni che non sono dominate l'una dall'altra, essendo le migliori considerando il resto delle possibili soluzioni. Significa che un'unica soluzione non può essere superiore a tutte le altre soluzioni rispetto a tutti gli obiettivi. Man mano che un singolo obiettivo nei MOOP migliora, un altro peggiorerà; pertanto, ogni soluzione dell'insieme di Pareto include almeno un obiettivo inferiore a un'altra soluzione in quell'insieme di Pareto, sebbene entrambi siano superiori agli altri nel resto dello spazio di ricerca (Akbari et al., 2014).

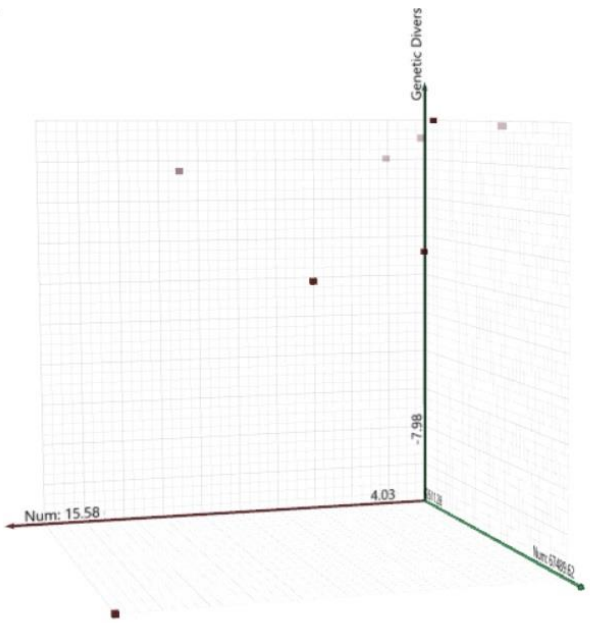



*Figura 61 - Fronte di Pareto ottimale con evidenziate le soluzioni possibili*

Preso un insieme di punti *P* e un punto di riferimento *r* nello spazio degli obiettivi, l'ipervolume misura la grandezza (misura di Lebesgue) della regione di spazio dominata dai punti in P e limitata superiormente dal punto r. Se P è una approssimazione del fronte di Pareto, l'ipervolume può essere impiegato come indice di qualità dell'approssimazione, e precisamente, maggiore è l'indicatore migliore è l'approssimazione del fronte.

Sebbene spesso limitati alla risoluzione di problemi ad alta dimensione, con sei o più obiettivi, le applicazioni dell'ipervolume per la risoluzione di problemi multi-obiettivo risultano

molteplici e, ad esempio, negli algoritmi evoluzionistici, la selezione degli individui migliori può essere fatta in base al contributo di questi ultimi all'ipervolume che permette di estrarre k punti da un insieme $n > k$ soluzioni non dominate.

Poiché l'indicatore in quanto tale opera su (multi) insiemi di soluzioni, mentre la selezione considera soluzioni singole, è necessaria una strategia per assegnare valori di fitness alle soluzioni. La maggior parte degli algoritmi basati sull' ipervolume esegue prima uno smistamento e poi classifica le soluzioni all'interno di un particolare fronte in base alla perdita di ipervolume derivante dalla rimozione di una soluzione specifica (Knowles e Corne, 2003; Emmerich et al., 2005, Igel et al., 2007; Bader et al., 2010).

Il ciclo principale di HypE rappresenta un algoritmo standard evolutivo che consiste nella successiva applicazione di selezione dell'accoppiamento (*mating selection*), variazione e selezione ambientale (*environmental selection*).

La selezione dell'accoppiamento consiste nella selezione delle soluzioni da variare e può, in genere, essere effettuata tramite un procedimento detto a "torneo" in cui l'algoritmo opera andando a creare dei tornei abbinando a caso *k* individui della popolazione iniziale e andandone a confrontare i corrispondenti valori di *fitness; l*a soluzione migliore viene scelta e va avanti al turno successivo, entrando a fare parte così del *mating pool*, mentre le altre vengono squalificate. Se eseguito sistematicamente, ogni individuo può essere fatto per partecipare esattamente a *k* tornei e la soluzione migliore della popolazione sarà quella che vincerà tutti e *k* gli scontri.

La procedura di variazione comprende l'applicazione di operatori di mutazione e ricombinazione per generare N prole; infine, la selezione ambientale mira a selezionare le N soluzioni più promettenti dall'unione degli insiemi della popolazione dei genitori e della prole; più precisamente, crea una nuova popolazione eseguendo i seguenti due passaggi:

1. In primo luogo, l'unione di genitori e figli è divisa in partizioni disgiunte usando il principio dello smistamento non dominato (Goldberg, 1989; Deb et al., 2000), noto anche come profondità di dominanza. A partire dal livello di profondità di dominanza più basso, le partizioni vengono spostate una ad una nella nuova popolazione fintanto che viene raggiunta la prima partizione che non può essere trasferita completamente. Ciò corrisponde allo schema utilizzato nella maggior parte degli ottimizzatori multi-obiettivo basati su ipervolumi (Emmerich et al., 2005; Igel et al., 2007; Brockhoff e Zitzler, 2007).

2. La partizione che si adatta solo parzialmente alla nuova popolazione viene quindi elaborata. In ogni fase, vengono calcolati i valori di fitness per la partizione in esame e l'individuo con la forma fisica peggiore viene rimosso; se più individui condividono la stessa forma fisica minima, uno di loro viene selezionato in modo uniforme a caso. Questa procedura viene ripetuta fino a quando la partizione non è stata ridotta alla dimensione desiderata, cioè fino a quando non si adatta agli slots rimasti nella nuova popolazione.

Per quanto riguarda l'assegnazione del fitness, il numero di obiettivi determina se vengono considerati i valori di ipervolume $(I_h^k)$ esatti o stimati; se sono coinvolti meno di quattro obiettivi, si utilizza l'algoritmo 1, altrimenti l'algoritmo 3 in figura 62. Quest'ultimo lavora con un numero fisso di punti di campionamento per stimare i valori dell'ipervolume $I_h^k$, indipendentemente dalla decisione da prendere; non è quindi necessario calcolare la varianza delle stime ed è sufficiente aggiornare per ogni campione un array che memorizza i valori di fitness dei membri della popolazione.

**Algorithm 1** Hypervolume-Based Fitness Value Computation

**Require:** population $P \in \Psi$, reference set $R \subseteq Z$, fitness parameter $k \in \mathbb{N}$

**procedure** *computeHypervolume*$(P, R, k)$
  $\mathcal{F} \leftarrow \bigcup_{a \in P}\{(a, 0)\}$
  **return** *doSlicing*$(\mathcal{F}, R, k, n, 1, (\infty, \infty, \ldots, \infty))$;
**end procedure**

**Algorithm 3** Hypervolume-based Fitness Value Estimation

**Require:** population $P \in \Psi$, reference set $R \subseteq Z$, fitness parameter $k \in \mathbb{N}$, number of sampling points $M \in \mathbb{N}$

**procedure** *estimateHypervolume*$(P, R, k, M)$
  **for** $i \leftarrow 1, n$ **do** /*determine sampling box $S$*/
    $l_i = \min_{a \in P} f_i(a)$
    $u_i = \max_{(r_1,\ldots,r_n) \in R} r_i$
  **end for**
  $S \leftarrow [l_1, u_1] \times \cdots \times [l_n, u_n]$
  $V \leftarrow \prod_{i=1}^{n} \max\{0, (u_i - l_i)\}$
  $\mathcal{F} \leftarrow \bigcup_{a \in P}\{(a, 0)\}$ /*reset fitness assignment*/
  **for** $j \leftarrow 1, M$ **do** /*perform sampling*/
    choose $s \in S$ uniformly at random
    **if** $\exists r \in R : s \preceq r$ **then**
      $UP \leftarrow \bigcup_{a \in P,\ f(a) \preceq s}\{f(a)\}$
      **if** $|UP| \leq k$ **then** /*hit in a relevant partition*/
        $\alpha \leftarrow \prod_{l=1}^{|UP|-1} \frac{k-l}{|P|-l}$
        $\mathcal{F}' \leftarrow \emptyset$
        **for all** $(a, v) \in \mathcal{F}$ **do** /*update hypervolume estimates*/
          **if** $f(a) \preceq s$ **then**
            $\mathcal{F}' \leftarrow \mathcal{F}' \cup \{(a, v + \frac{\alpha}{|UP|} \cdot \frac{V}{M})\}$
          **else**
            $\mathcal{F}' \leftarrow \mathcal{F}' \cup \{(a, v)\}$
          **end if**
        **end for**
        $\mathcal{F} \leftarrow \mathcal{F}'$
      **end if**
    **end if**
  **end for**
  **return** $\mathcal{F}$
**end procedure**

*Figura 62 – Algoritmi Hype per l'assegnazione dei valori di fitness*

### 4.3.2. Risultati

I risultati dell'analisi strutturale possono essere visualizzati utilizzando tre principali componenti "View" disponibili nel plug-in Karamba (nell'area 'Result' della barra degli strumenti), ognuno dei quali ha una differente funzionalità di visualizzazione:

- < Model View >: consente di etichettare varie componenti e visualizzare la forma del modello sotto deformazione;
- < Beam View >: consente di visualizzare i dati sulle prestazioni del modello tramite dei colori direttamente applicati sugli elementi del fascio modello stesso;
- < Shell View >: consente di visualizzare i dati sulle prestazioni del modello tramite dei colori direttamente applicati sugli elementi guscio (*shell*) del modello stesso.

Tutti e tre i componenti richiedono un valore di input "Model" che prende in ingresso il modello strutturale da visualizzare e forniscono un valore di output "Model" che restituisce il modello strutturale analizzato con le informazioni di visualizzazione richieste.

Le opzioni dei componenti di visualizzazione possono essere viste cliccando sulle barre nere di intestazione; esse contengono infatti un set di opzioni tra cui è possibile scegliere per visualizzare, nel caso del < Model View > ad esempio, la forma deformata della struttura e i vari elementi del modello.

Nel caso del < Beam View >, il componente consente invece di vedere diversi tipi di informazioni strutturali come colori applicati direttamente sugli elementi; l'opzione "Cross Section" colora diversamente le travi in funzione della sezione trasversale assegnata, quella "Displacement" le colora in base alla distanza di cui si muovono per effetto del carico, le opzioni "Axial Stresses" e "Utilization" colorano le travi in base alle sollecitazioni sotto carico e forniscono rispettivamente informazioni sulle sollecitazioni e su queste ultime in termini di percentuale della capacità massima di sollecitazione del materiale, ovvero in funzione delle proprietà del materiale della trave considerata.

Nel caso in esame (Figura 63), è stata creata una visualizzazione della struttura con informazioni sulle prestazioni strutturali visualizzate sugli elementi; il modello analizzato (ottimizzato in termini di sezione trasversale) è stato prima passato in ingresso al componente < Model View > per creare la forma e, l'output "Model" di quest'ultimo, è stato a sua volta passato come input al componente < Beam View > per visualizzare le prestazioni di ogni trave.

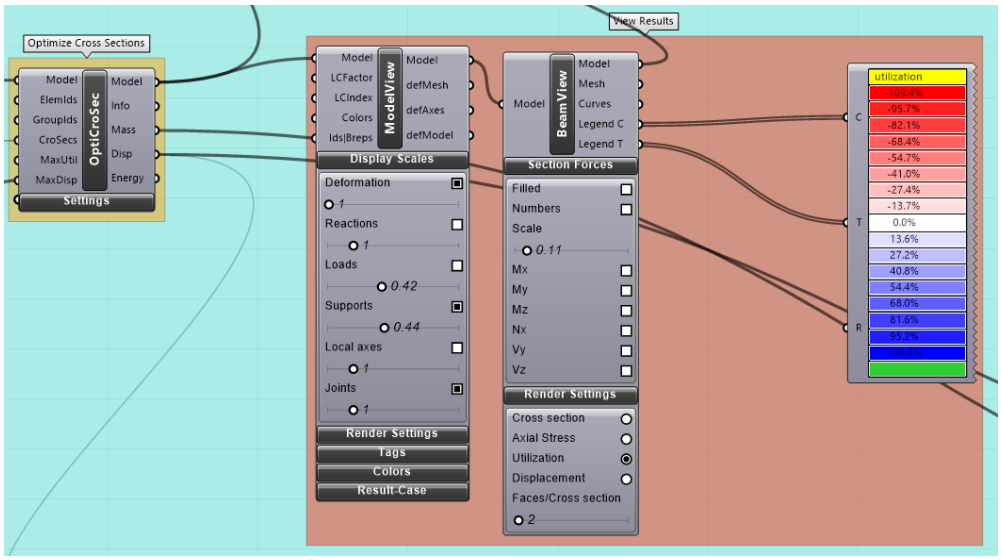



*Figura 63 – Componenti di visualizzazione dei risultati*

A questo punto è stata avviata l'ottimizzazione (topologica, dimensionale e di forma contemporaneamente) i cui risultati sono discussi di seguito.

- Risultati dell'ottimizzazione topologica

I risultati dell'ottimizzazione topologica sono stati ottenuti considerando dei valori iniziali (valori prima del processo di ottimizzazione), che servono solo per effettuare delle comparazioni in termini di riduzione di massa e di spostamento massimo.

I valori iniziali della struttura non ottimizzata in termini di massa e spostamento massimo sono:

- Massa: $m = 53546{,}2518\ kg$
- Spostamento massimo: $\delta_{max} = 23{,}1193\ cm$

I valori iniziali delle variabili di design sono invece:

- Numero di elementi in x: $n_x = 10$
- Numero di elementi in y: $n_y = 8$
- Posizione (x, y, z ) dei quattro punti di appoggio, scelte random:

$$(10, 6.667, 0)$$
$$(6.667, 13.333, 0)$$
$$(0, 10, 0)$$
$$(20, 13.333, 0)$$

Una volta impostati tutti i parametri iniziali, è stata eseguita l'ottimizzazione multi-obiettivo effettuata mediante l'algoritmo $HypE\ Reduction$ in cui, come già detto, le funzioni da minimizzare simultaneamente sono:

- $f(x)$: Massa (m)
- $g(x)$: Spostamento massimo $\delta_{max}$

Imponendo:

- Disequazione di vincolo: $\delta_{max} \leq \frac{1}{150} \cdot L_{Tot}$
- Max numero di generazioni: $N_{max,gen} = 300$
- Dimensione della popolazione: $N_{pop} = 20$

Alla fine del processo di ottimizzazione, effettuato con il solver *Octupus*, l'algoritmo fornisce un set di soluzioni incluse nel Fronte di Pareto ottimale rappresentanti le migliori soluzioni (che non sono dominate da nessun'altra) tra tutte quelle possibili nel problema di ottimizzazione.

Nel caso studio, si è ottenuto che il numero delle soluzioni non dominate sul fronte di Pareto è pari a 4; le soluzioni sono evidenziate in giallo sul dominio in figura 64 mentre i restanti punti rappresentano i "genitori", chiamati *elite*, delle soluzioni effettive.

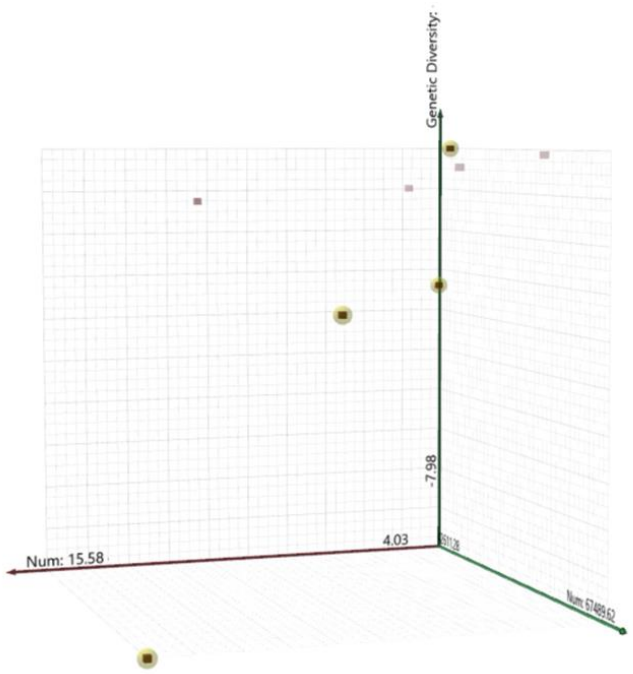
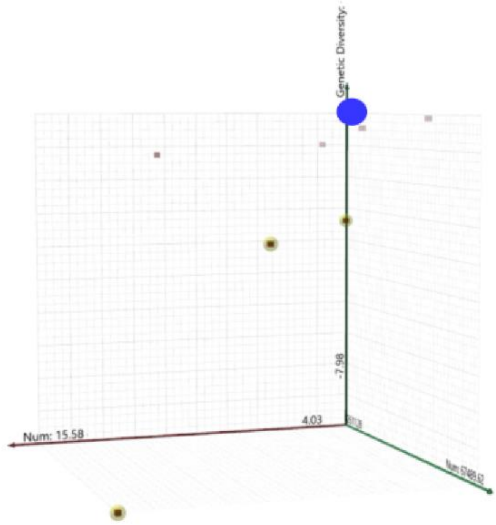



*Figura 64 – Raffigurazione delle soluzioni dell'ottimizzazione sul fronte di Pareto*

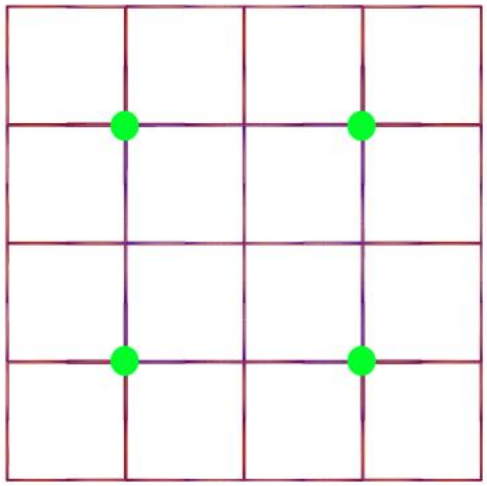
Di seguito, si riportano risultati ottimali in termini numero degli elementi in direzione x e y $(n_x, n_y)$, massa, spostamento massimo e posizione degli appoggi.

- **Soluzione 1**




Massa (Kg) = 43776.947341
$\delta_{max}$(cm) = 4.10869

$\delta_{max} \leq 1/150 * L_{tot}$



Posizione dei supporti:
{5, 5, 0}
{15, 5, 0}
{5, 15, 0}
{15, 15, 0}

nx= 8
ny= 8

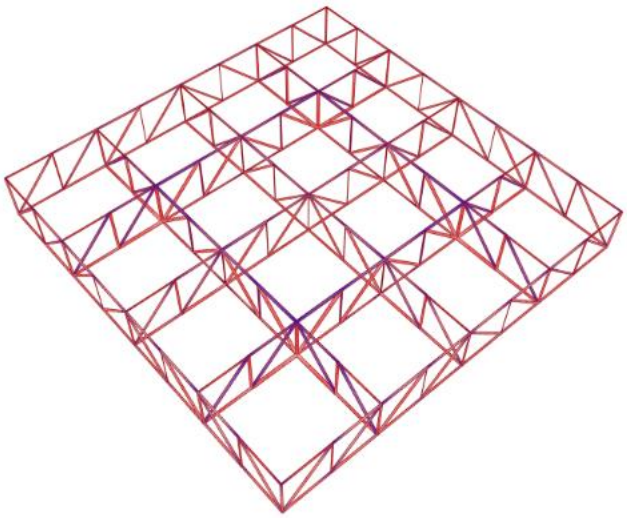
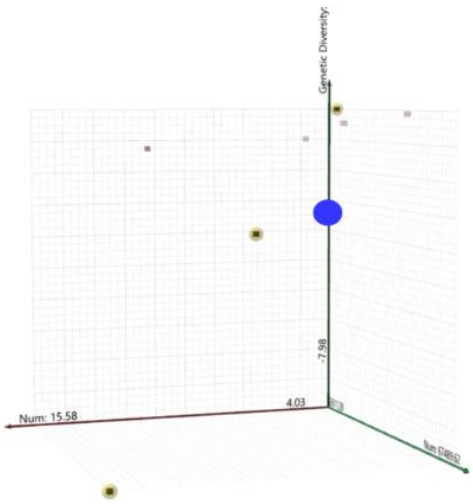


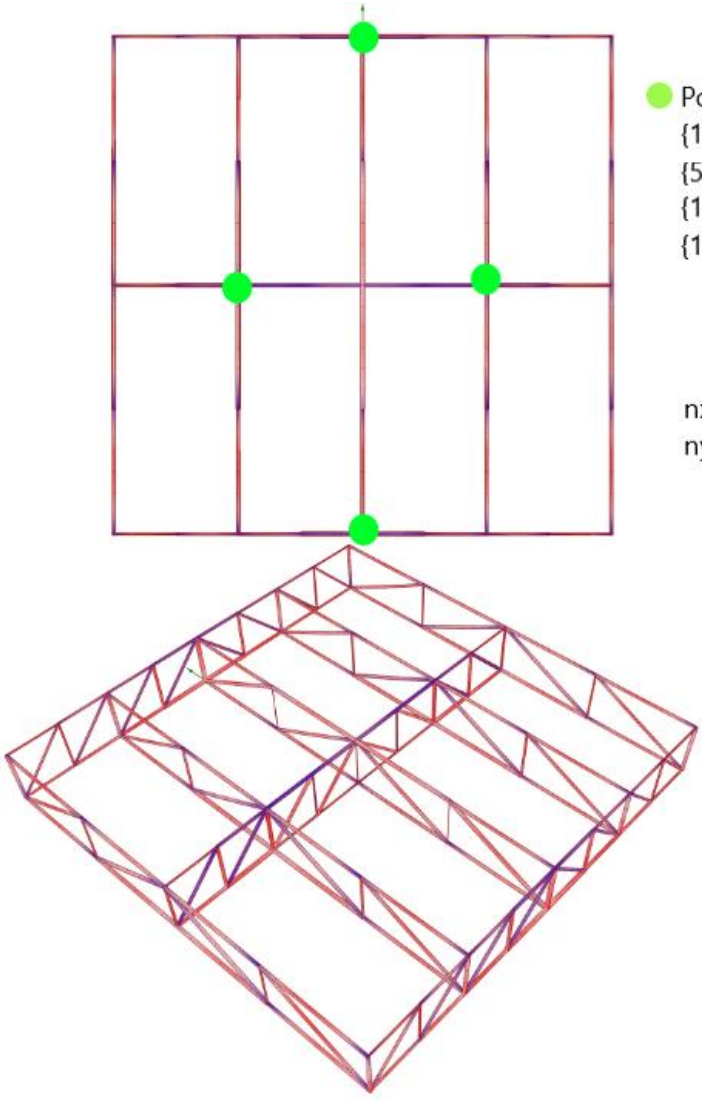
*Figura 65 - Risultati ottimizzati in corrispondenza della Soluzione 1*

- **Soluzione 2**




Massa (Kg) = 51737.971102
$\delta_{max}$(cm) = 6.264787

$\delta_{max} \leq 1/150 * L_{tot}$

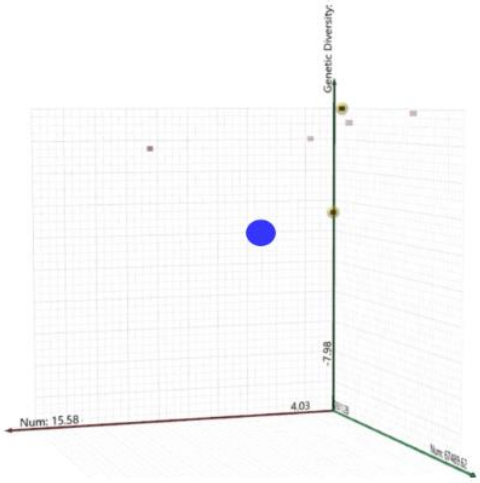


Posizione dei supporti:
{10, 0, 0}
{5, 10, 0}
{10, 20, 0}
{15, 15, 0}

nx= 8
ny= 4

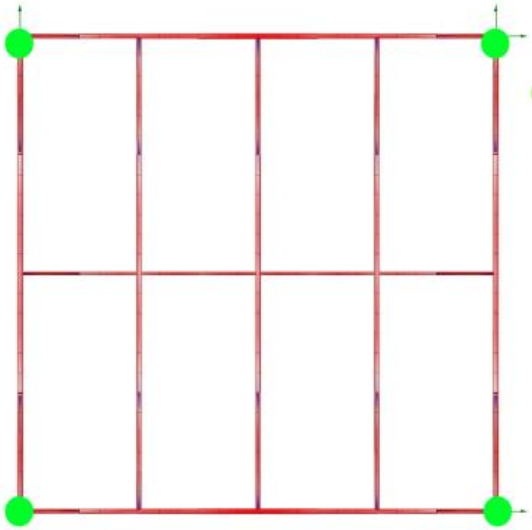
*Figura 66 – Risultati ottimizzati in corrispondenza della Soluzione 2*

- **Soluzione 3**




Massa (Kg) = 60603.16635
$\delta_{max}$(cm) = 6.770903

$\delta_{max} \leq 1/150 * L_{tot}$



Posizione dei supporti:
{0, 0, 0}
{20, 0,0}
{20, 20, 0}
{0, 20,0}

nx= 8
ny= 4

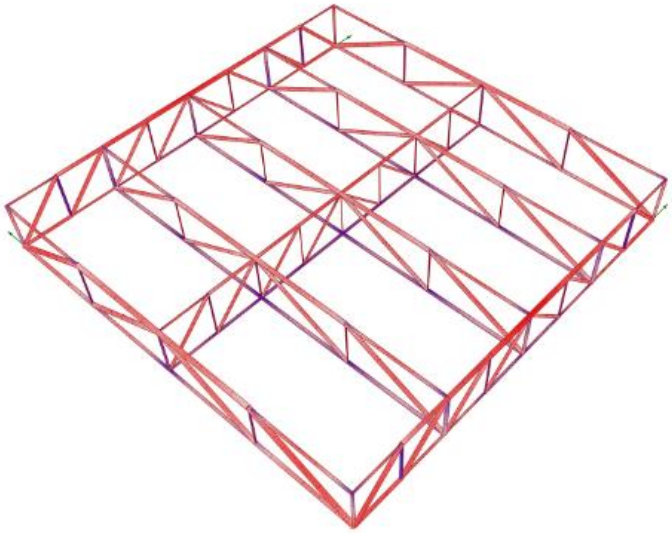
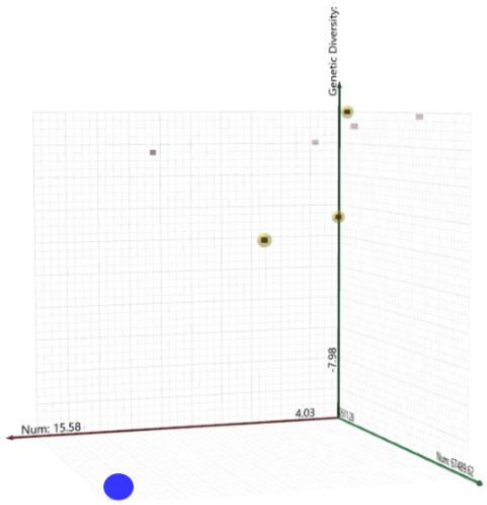


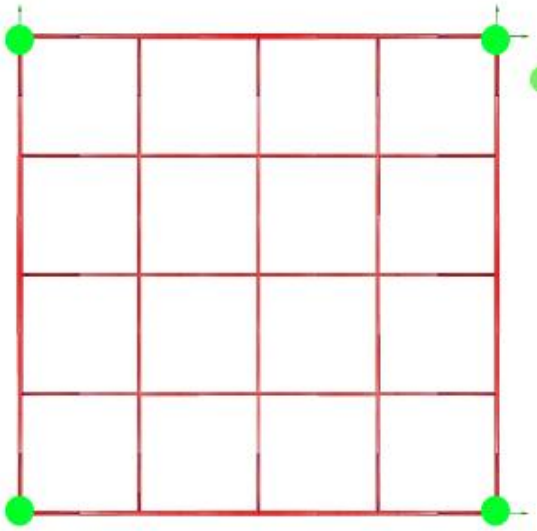
*Figura 67 – Risultati ottimizzati in corrispondenza della Soluzione 3*

- **Soluzione 4**




Massa (Kg) = 64918.161412
$\delta_{max}$(cm) = 8.163198

$\delta_{max} \leq 1/150 * L_{tot}$



Posizione dei supporti:
{0, 0, 0}
{20, 0, 0}
{20, 20, 0}
{0, 20, 0}

nx= 8
ny= 8

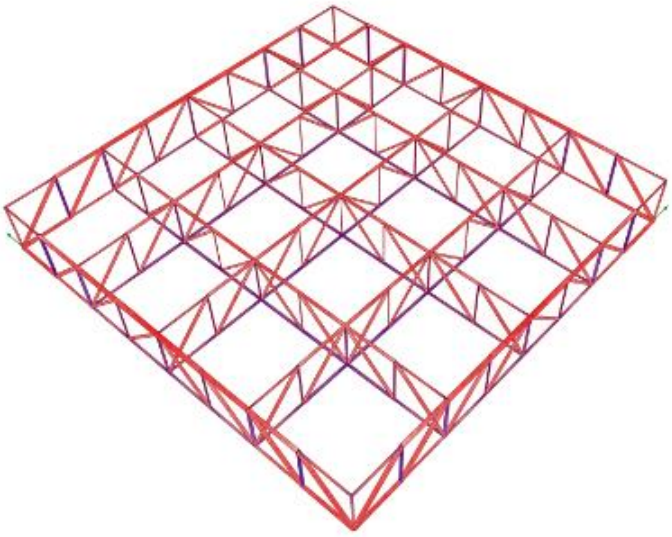


*Figura 68 - Risultati ottimizzati in corrispondenza della Soluzione 4*

La tabella 1 seguente riassume i valori ottenuti dal processo di ottimizzazione:

| ***Soluzioni*** | $\boldsymbol{m}$ **[Kg]** | $\boldsymbol{\delta\ max}$ **[cm]** |
|---|---|---|
| **S1** | 43776,9473 | 4,1087 |
| **S2** | 51737,9711 | 6,2648 |
| **S3** | 60603,1663 | 6,7709 |
| **S4** | 64918,1614 | 8,1632 |

*Tabella 1- Risultati dell'ottimizzazione*

In tabella 2 vengono, invece, riportate le variazioni in percentuale di massa e spostamento massimo dei valori ottenuti rispetto a quelli iniziali:

| ***Soluzioni*** | $\boldsymbol{Var.m\ \%}$ **[Kg]** | $\mathbf{Var.}\boldsymbol{\delta\ max\ \%}$ **[cm]** |
|---|---|---|
| **S1** | -18,24 | -82,8 |
| **S2** | -3,38 | -72,9 |
| **S3** | +13,18 | -70,7 |
| **S4** | +21,24 | 8,1632 |

*Tabella 2 – Risultati dell'ottimizzazione in termini di variazione percentuale*

Come è possibile osservare, solo le prime due soluzioni mostrano una variazione percentuale negativa rispetto ai valori iniziali, ovvero una riduzione, sia in termini di massa sia di spostamento massimo; la terza soluzione determina invece un incremento della massa con conseguente riduzione dell'abbassamento e risulta, pertanto, un'alternativa plausibile dal momento in cui le due variabili in gioco presentano un legame di inversa proporzionalità. La quarta soluzione determina un incremento di entrambe le variabili considerate risultando, pertanto, la meno performante.

- Risultati dell'ottimizzazione dimensionale e di forma

Al fine di ottenere una sezione trasversale ottimizzata per la struttura in esame, al componente <OptiCrosSec > è stato passato il modello ottenuto dall'assemblaggio e sono state inserite tre diverse famiglie delle sezioni tra cui l'algoritmo può scegliere al fine di eseguire l'ottimizzazione:

- sezione circolare cava (RO);
- sezione rettangolare cava;

- sezione a I (IPE)

Prendendo questi parametri in ingresso, il componente sceglie la sezione ottimizzata, che minimizza al contempo massa e spostamento massimo, e la assegna, variandola però in spessore (range di 10 – 80 cm), a tutti gli elementi beam del modello.

Alla fine del processo di ottimizzazione, i risultati mostrano che la copertura risulta ottimizzata utilizzando la sezione circolare cava (RO); essa si è infatti rivelata la più performante per tutte e quattro le migliori soluzioni, alla fine del processo di ottimizzazione topologica (soluzioni non dominate sul fronte di Pareto), in corrispondenza di ognuno delle quali sono stati solo variati gli spessori degli elementi, delle sezioni trasversali RO ottimizzate.

I risultati numerici ottenuti dall'ottimizzazione per le quattro soluzioni sono interamente riportati in Appendice 2.

Di seguito si riportano invece le sezioni trasversali ottimizzate del modello nei quattro casi:

- **Soluzione 1**

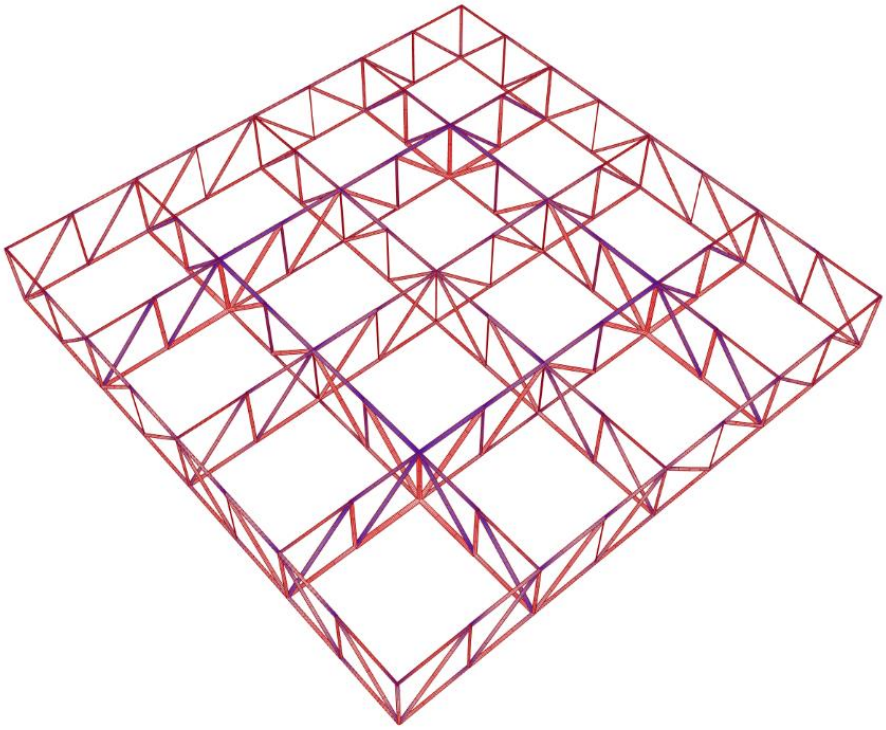


*Figura 69 – Ottimizzazione shape e size per la soluzione 1*

- **Soluzione 2**

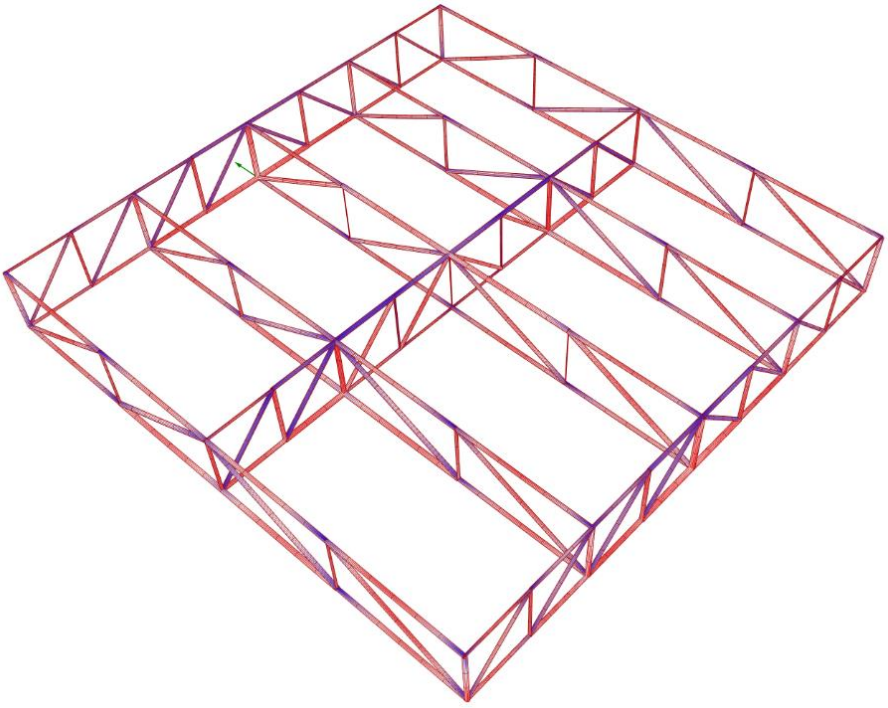


*Figura 70 - Ottimizzazione shape e size per la soluzione 2*

- **Soluzione 3**

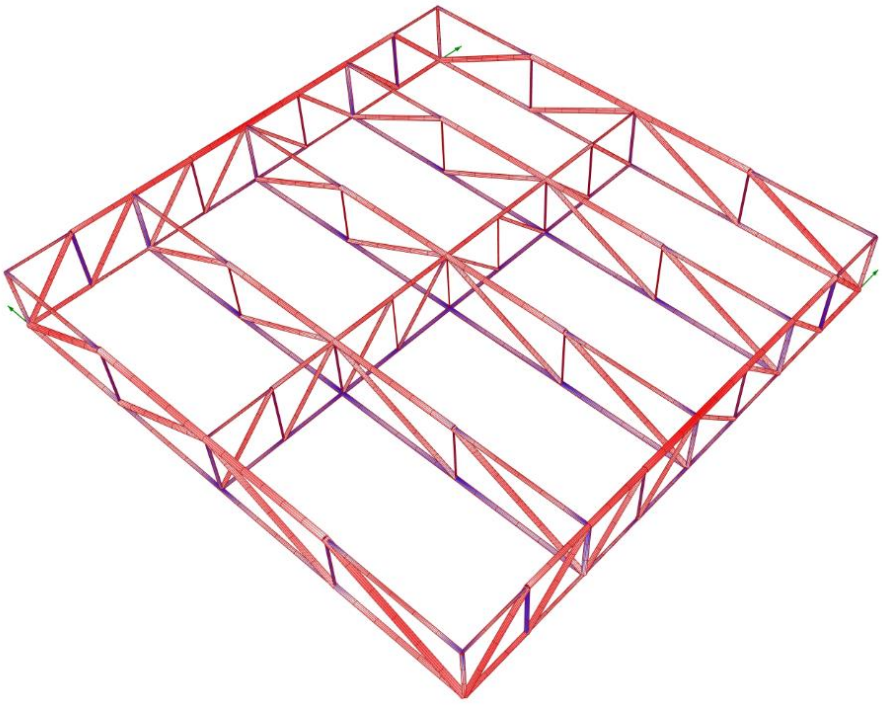


*Figura 71 - Ottimizzazione shape e size per la soluzione 3*

- **Soluzione 4**

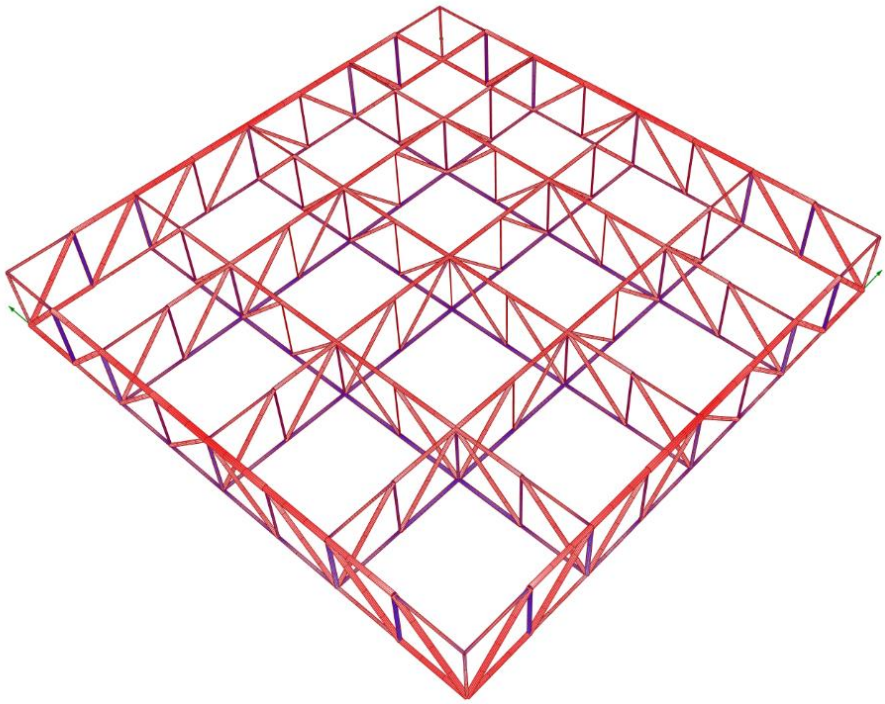


*Figura 72 - Ottimizzazione shape e size per la soluzione 4*

# 5. Conclusioni

La presente tesi affronta una metodologia di ottimizzazione strutturale per la progettazione preliminare di una copertura, fase in cui vengono sviluppate tutte le basi per ottenere il prodotto finale, utilizzando strumenti innovativi che consentono di fondere le grandi potenzialità offerte dal design computazionale e parametrico e dagli algoritmi evolutivi.

Attraverso il processo di ottimizzazione parametrica è infatti possibile innovare la forma delle strutture finali, fornendo al progettista spunti e suggerimenti spesso fuori dagli schemi, oltre che ridurre notevolmente i tempi e la complessità di calcolo rispetto ai metodi tradizionali, fino ad ora utilizzati, che spesso necessitano dello sviluppo di funzioni derivate.

Il metodo proposto ha permesso di creare un modello con una "storia di costruzione" in grado di rimane accessibile e modificabile in ogni momento della fase progettazione (andandone semplicemente a variare i parametri di design definiti dal progettista) fornendo la possibilità di ottenere diverse soluzioni fattibili da un unico studio e, soprattutto, la soluzione migliore per quanto riguarda il comportamento strutturale della copertura, aspetto oggigiorno richiesto nel mondo della progettazione strutturale in cui c'è la necessità di controllare la forma degli elementi strutturali e, allo stesso tempo, le prestazioni offerte.

Partendo dalla parametrizzazione di ogni elemento costituente la copertura in acciaio, la forma maggiormente performante è stata ricavata utilizzando degli algoritmi evolutivi che hanno permesso, al fine di minimizzare simultaneamente la massa e lo spostamento massimo della struttura, di mettere insieme i tre differenti tipi di ottimizzazione strutturale (topologica, dimensionale e di forma) fornendo la possibilità di ottenere, allo stesso tempo, una struttura ottimizzata in termini di forma e dimensione delle sezioni trasversali e di distribuzione della densità del materiale.

Inoltre, l'utilizzo di Grasshopper, come strumento di programmazione visual, ha reso possibile implementare un'accurata analisi strutturale FEM, per il pre-processing e il post-processing dell'elemento da ottimizzare, senza aver dovuto fare uso di tecniche di programmazione complesse; questo tipo di strumento presenta infatti l'enorme vantaggio di non richiedere agli utenti un alto livello di esperienza di scripting o di programmazione (C ++, Phyton, Visual Basic) per generare i modelli, anche i più complicati. Nonostante ciò, il modello geometrico è stato realizzato facendo uso di un codice di programmazione base implementato in Python.

Il vantaggio del metodo sperimentale proposto è dato anche dalla comunicazione diretta, permessa da Grasshopper, tra il modello geometrico, l'analisi agli elementi finiti e l'implementazione del problema di ottimizzazione che permette di dimezzare il tempo di lavoro; inoltre, attraverso l'implementazione dell'ottimizzazione multi-obiettivo (tramite l'algoritmo HypE Reduction), è stato possibile ricavare delle soluzioni capaci di determinare la riduzione dello spostamento massimo della copertura e, al contempo, la riduzione della quantità di materiale di circa l'80% rispetto alla soluzione iniziale non ottimizzata che si traduce, a sua volta, in un notevole abbattimento dei costi.

Infine, grazie alla potenzialità offerta dal design computazionale, che permette di aggiornare il modello in tempo reale alle modifiche effettuate sui valori di input, è possibile utilizzare lo stesso framework per risultati o elementi diversi variando semplicemente la geometria da analizzare/ottimizzare; tuttavia, l'utilizzo di questo metodo, richiede una determinata conoscenza dei concetti base di geometria computazionale e delle logiche derivanti dall'uso della programmazione visuale.

# 6. Appendici

## Appedice 1

```
#Import Libraries
import ghpythonlib.components as gh
import rhinoscriptsyntax as rhlib
import Rhino as rc

#variable
_Lx=Lx
_Ly=Ly
_H=H
delta_iX=Delta_iX
delta_iY=Delta_iY
_nX=nX_segments
_nY=nY_segments
#------------------X_Direction--------------------------------------
#POINTS
my_listPX_Lower=[]
my_listPX_Upper=[]
copy_my_listPX_Lower=[]

a=0
for i in range(0,_nX+1): #nPunti volte
    my_point_low=gh.ConstructPoint(delta_iX*a,-(2*delta_iY),0)
    my_listPX_Lower.insert(a,my_point_low)
    copy_my_listPX_Lower.insert(a,my_point_low)
    my_point_upp=gh.ConstructPoint(delta_iX*a,-(2*delta_iY),_H)
    my_listPX_Upper.insert(a,my_point_upp)
    a=a+1

#LINES
my_LineV=gh.Line(my_listPX_Lower,my_listPX_Upper)

my_LineO_low=[]
d=0
for i in range (0,nX):
    my_LinesO_low=gh.Line(my_listPX_Lower[d],my_listPX_Lower[d+1])
    my_LineO_low.insert(d,my_LinesO_low)
    d=d+1

my_LineO_upp=[]
d1=0
for i in range (0,nX):
    my_LinesO_upp=gh.Line(my_listPX_Upper[d1],my_listPX_Upper[d1+1])
    my_LineO_upp.insert(d1,my_LinesO_upp)
    d1=d1+1

#DIAGONALS
newlstPX_Upper=list(my_listPX_Upper)
newlstPX_Upper.pop(0)

slstA_upp=newlstPX_Upper[:int(_nX/2)]
slstA_low=my_listPX_Lower[:int(_nX/2+1)]
slstB_upp=newlstPX_Upper[int(_nX/2-1):]
slstB_low=my_listPX_Lower[int(_nX/2+1):]
```

```
my_LineDsx=gh.Line(slstA_low,slstA_upp)
my_LineDdx=gh.Line(slstB_upp,slstB_low)
my_LineDsx.pop(-1)
my_LineDdx.pop(-1)

#PARALLEL X-ELEMENTS
lstL1=[]
lstL2=[]
lstL3=[]
lstLD1=[]
lstLD2=[]


for k in range(0,int(_nY/2+1)):
    Vect=[0,delta_iY*2,0]
    for cLow in range(0,len(my_LineO_low)):
        my_moveLineO_low=rhlib.MoveObject(my_LineO_low[cLow],Vect)
        lstL1.append(my_moveLineO_low)
    for cUpp in range(0,len(my_LineO_upp)):
        my_moveLineO_upp=rhlib.MoveObject(my_LineO_upp[cUpp],Vect)
        lstL2.append(my_moveLineO_upp)
    for ddx in range(0,len(my_LineDdx)):
        my_moveLineDdx=rhlib.MoveObject(my_LineDdx[ddx],Vect)
        lstLD2.append(my_moveLineDdx)
    for dsx in range(0,len(my_LineDsx)):
        my_moveLineDsx=rhlib.MoveObject(my_LineDsx[dsx],Vect)
        lstLD1.append(my_moveLineDsx)
    for j in range(0,len(my_LineV)):
        my_moveLineV=rhlib.MoveObject(my_LineV[j],Vect)
        lstL3.append(my_moveLineV)

#-----------------------------Y_Direction-------------------------
#POINTS
my_listPY_Lower2=[]
my_listPY_Upper2=[]
copy_my_listPY_Lower2=[]

a=0
for i in range(0,_nY+1):
    my_point_low2=gh.ConstructPoint(-(2*delta_iX),delta_iY*a,0)
    my_listPY_Lower2.insert(a,my_point_low2)
    copy_my_listPY_Lower2.insert(a,my_point_low2)
    my_point_upp2=gh.ConstructPoint(-(2*delta_iX),delta_iY*a,_H)
    my_listPY_Upper2.insert(a,my_point_upp2)
    a=a+1

#LINES
my_LineV2=gh.Line(my_listPY_Lower2,my_listPY_Upper2)
my_LineO_low2=[]
d=0
for i in range (0,nY):
    my_LinesO_low2=gh.Line(my_listPY_Lower2[d],my_listPY_Lower2[d+1])
    my_LineO_low2.insert(d,my_LinesO_low2)
    d=d+1

my_LineO_upp2=[]
d1=0
for i in range (0,_nY):
    my_LinesO_upp2=gh.Line(my_listPY_Upper2[d1],my_listPY_Upper2[d1+1])
    my_LineO_upp2.insert(d1,my_LinesO_upp2)
```

```
    d1=d1+1

#DIAGONALS
newlstPY_Upper2=list(my_listPY_Upper2)
newlstPY_Upper2.pop(0)

slstA_upp2=newlstPY_Upper2[:int(_nY/2)]
slstA_low2=my_listPY_Lower2[:int(_nY/2+1)]
slstB_upp2=newlstPY_Upper2[int(_nY/2-1):]
slstB_low2=my_listPY_Lower2[int(_nY/2+1):]
my_LineDsx2=gh.Line(slstA_low2,slstA_upp2)
my_LineDdx2=gh.Line(slstB_upp2,slstB_low2)
my_LineDsx2.pop(-1)
my_LineDdx2.pop(-1)

#PARALLEL Y-ELEMENTS
lstL1Y=[]
lstL2Y=[]
lstL3Y=[]
lstLD1Y=[]
lstLD2Y=[]

for k in range(0,int(_nX/2+1)):
    Vect2=[delta_iX*2,0,0]
    for cLow in range(0,len(my_LineO_low2)):
        my_moveLineO_low2=rhlib.MoveObject(my_LineO_low2[cLow],Vect2)
        lstL1Y.append(my_moveLineO_low2)
    for cUpp in range(0,len(my_LineO_upp2)):
        my_moveLineO_upp2=rhlib.MoveObject(my_LineO_upp2[cUpp],Vect2)
        lstL2Y.append(my_moveLineO_upp2)
    for ddx in range(0,len(my_LineDdx2)):
        my_moveLineDdx2=rhlib.MoveObject(my_LineDdx2[ddx],Vect2)
        lstLD2Y.append(my_moveLineDdx2)
    for dsx in range(0,len(my_LineDsx2)):
        my_moveLineDsx2=rhlib.MoveObject(my_LineDsx2[dsx],Vect2)
        lstLD1Y.append(my_moveLineDsx2)
    for j in range(0,len(my_LineV2)):
        my_moveLineV2=rhlib.MoveObject(my_LineV2[j],Vect2)
        lstL3Y.append(my_moveLineV2)

#Base POINTS
lstBasePoint=[]
for k in range (0,int(_nY/2+1)):
    for i in range(0,int((_nX/2+1))):
        my_P=gh.ConstructPoint(delta_iX*2*i,delta_iY*2*k,0)
        lstBasePoint.append(my_P)

#OUTPUT FINALI
lstVtot=[] #lista totale linee verticali
lstVtot=lstL3+lstL3Y
lstOlowTot=[] #lista totale linee orizzontali inferiori
lstOlowTot=lstL1+lstL1Y
lstOuppTot=[] #lista totale linee orizzontali superiori
lstOuppTot=lstL2+lstL2Y
lstDleftTot=[] #lista totale linee diagonali sx
lstDleftTot=lstLD1+lstLD1Y
lstDrightTot=[] #lista totale linee diagonali dx
lstDrightTot=lstLD2+lstLD2Y
lstDTot=[] #lista totale diagonali
lstDTot=lstDleftTot+lstDrightTot
```

```
o_VLines=lstVtot
o_OlowLines=lstOlowTot
o_OuppLines=lstOuppTot
o_Diagonals=lstDTot
o_BasePoints=lstBasePoint
```

## Appedice 2

| Solizione 1 | Soluzione 2 | Soluzione 3 | Soluzione 4 |
|---|---|---|---|
| RO101.6/28 | RO152.4/36 | RO127/30 | RO88.9/17.5 |
| RO73/16 | RO88.9/17.5 | RO127/28 | RO127/30 |
| RO101.6/17.5 | RO152.4/40 | RO159/45 | RO133/36 |
| RO31.8/8 | RO127/25 | RO76.1/20 | RO76.1/20 |
| RO76.1/16 | RO193.7/45 | RO127/36 | RO101.6/22.2 |
| RO127/25 | RO108/30 | RO101.6/22.2 | RO76.1/17.5 |
| RO152.4/45 | RO133/36 | RO70/16 | RO88.9/22.2 |
| RO88.9/20 | RO159/45 | RO70/14.2 | RO44.5/11 |
| RO60.3/16 | RO70/16 | RO40/10 | RO70/14.2 |
| RO101.6/16 | RO40/10 | RO82.5/20 | RO70/17.5 |
| RO44.5/10 | RO76.1/17.5 | RO57/14.2 | RO44.5/10 |
| RO33.7/8.8 | RO101.6/14.2 | RO152.4/45 | RO33.7/8.8 |
| RO114.3/32 | RO30/7.1 | RO101.6/28 | RO159/40 |
| RO101.6/20 | RO108/20 | RO88.9/25 | RO101.6/25 |
| RO101.6/22.2 | RO127/32 | RO108/25 | RO101.6/16 |
| RO127/36 | RO152.4/32 | RO114.3/28 | RO101.6/28 |
| RO168.3/45 | RO139.7/40 | RO127/32 | RO127/25 |
| RO108/20 | RO177.8/55 | RO168.3/45 | RO133/32 |
| RO88.9/16 | RO101.6/16 | RO133/30 | RO127/32 |
| RO114.3/25 | RO82.5/17.5 | RO159/40 | RO108/25 |
| RO127/30 | RO139.7/28 | RO152.4/40 | RO114.3/32 |
| RO114.3/28 | RO168.3/45 | RO168.3/36 | RO127/36 |
| RO101.6/25 | RO141.3/40 | RO88.9/14.2 | RO88.9/14.2 |
| RO82.5/12.5 | RO139.7/32 | RO168.3/40 | RO168.3/45 |
| RO108/25 | RO101.6/17.5 | RO193.7/50 | RO193.7/55 |
| RO82.5/16 | RO133/40 | RO219.1/45 | RO219.1/60 |
| RO108/28 | RO82.5/16 | RO82.5/17.5 | RO82.5/22.2 |
| RO139.7/40 | RO127/28 | RO133/32 | RO139.7/32 |
| RO127/28 | RO152.4/45 | RO139.7/36 | RO139.7/40 |
| RO101.6/14.2 | RO133/25 | RO219.1/40 | RO82.5/16 |
| RO152.4/32 | RO139.7/36 | RO133/25 | RO177.8/55 |
| RO133/36 | RO127/36 | RO193.7/36 | RO177.8/50 |
| RO108/30 | RO133/28 | RO193.7/40 | RO141.3/40 |
| | RO168.3/40 | RO177.8/50 | RO133/40 |
| | RO101.6/28 | RO177.8/45 | RO127/20 |

| | | | |
|---|---|---|---|
| | RO114.3/25 | RO152.4/30 | RO101.6/20 |
| | RO133/32 | RO133/28 | |
| | RO127/22.2 | RO114.3/25 | |
| | | RO101.6/14.2 | |
| | | RO219.1/32 | |
| | | RO159/32 | |
| | | RO159/36 | |

# 7. Bibliografia


- A. Tedeschi, “Architettura parametrica. Introduzione a Grasshopper: il plug-in per la modellazione generativa in Rhino*”*, IIa edizione, Brienza (PT): Le Penseur, 2010.
- A. Tedeschi, “Il processo è più importante del risultato, AAD Algorithmic Aided Design*”*.
- Altair Engineering Inc., “*Practical Aspects of Structural Optimization.”*, 2015.
- Andrea G.B.Tettamanzi, “Algoritmi evolutivi: concetti e applicazioni”, 2005.
- Bendsøe, Martin P., and Ole Sigmund. “Optimization of structural topology, shape and material”, Vol. 414. Berlin etc: Springer, 1995.
- Bendsoe, Martin Philip, and Ole Sigmund, “Topology optimization: theory, methods, and applications”, Springer Science & Business Media, 2013.
- Campana, E.F., Liuzzi, G., S. Lucidi, D. Peri, V. Piccialli, “New Global Optimization methods for Ship design problems”, Rapporto Tecnico INSEAN n. 2005-1, 2005 (sottosezione 6.2.1).
- Cheng, N., “Learning Design with Digital Sketching”, in B. & Brown Martens, Computer Aided Architectural Design Futures 2005.
- D.R. Jones, “The direct global optimization algorithm”, In C. Floudas, P. Pardalos (edit.): “Encyclopedia of Optimization”, Kluwer Academic Publishers, Dordrecht.
- Dr.lr.F.A.A. (Frank) Huijben and lr. J.C. (Chris) van de Ploeg, “Computational Design in Engineering”.
- FabLab Venezia, “Design parametrico e progettazione algoritmica”.
- Flanagan, Robert H., “Generative logic in digital design”, Automation in Construction, 2005.
- Giornale dell'Architettura, “Digital Revolution: l’impatto del Computational Design sulla progettazione”, 2020.
- https://Bim e Computational Design: progettazione tramite algoritmi (01building.it).
- https://www.researchgate.net/publication/30869860_Teaching_Generative_Design - Fischer T., Herr C. M., 2001
- J.-B. Hiriart-Urruty, “Conditions for Global Optimality. In R. Horts, P. Pardalos (edit.): “Handbook of Global Optimization”, Academic Publishers, 1995 (sottosezione 3.2.1).
- Johaness Bader and Eckart Zitzler, “Hype: an algorithm for fast Hypervolume-Based-Objective Optimization”.

- Koutamanis, Alexander and Mitossi, Vicky, “Computer vision in architectural design”, Design Studies, 1993.
- L.C.W. Dixon, G.P. Szego (edit.), “Towards Global Optimization”, Vol.2. NorthHolland, Amsterdam, 1978.
- Maria Aghaei Meibodi, “Generative Design Exploration: computation and material practice”.
- Martini, K., “Optimization and parametric modelling to support conceptual structural design”, International Journal of Architectural Computing, 2011.
- MENDEZ E., TOMAS I., “Computational Search in Architectural Design”, Politecnico di Torino, 2014.
- Parmee, I. C., & Bonham, C, “Towards the support of innovative conceptual design through interactive designer/evolutionary computing strategies”, Artificial Intelligence for Engineering Design, Analysis, and Manufacturing, 2000.
- Philips S., “Parametric design: a brief History”.
- R. Horst, P.M. Pardalos, N.V. Thoai, “Introuction to Global Optimization”, Kluwer Academic Publishers, Dordrecht, 2000.
- Randy L. Haupt and Sue Ellen Haupt, *“*Practical genetic algorithms*”,* Wiley Interscience, 2004
- Rowe, Peter G., “Design thinking” (Cambridge, Mass.: MIT Press), 1987.
- S.N.Sivanandam and S.N Deepa*, “*Introduction to Genetic Algorithms*”.* Springer, 2008.
- Schnabel, Marc Aurel, “Parametric designing in architecture”, in Andy Dong, Andrew Vande Moere, and John S. Gero (eds.), Computer-aided architectural design futures,2007.
- Sebnem Y. C., Fulya O. A., Tugrul Y., “Computational design, Parametric modelling and architectural education”.
- Sivam Krish, “A practical generative design method”, 2011.
- Stangl, T., M. Pribek, and S. Wartzack, "Integration of structural optimization in the engineering design process." *DS 77: Proceedings of the DESIGN 2014 13th International Design Conference*, 2014.
- Steele, James, “Architecture and computers: action and reaction in the digital design revolution”, London: Laurence King, 2001.
- Terzidis, Kostas, “Algorithmic architecture”, Amsterdam; London: Architectural Press, 2006.

- Valerio Lacagnina, “Algoritmi genetici”, Università di Palermo, 2003.
- Vercellis, C., “Ottimizzazione, Teoria, metodi, applicazioni, Milano”, McGraw-Hill Companies, 2008.
- Vincent J. Amuso and Jason Enslin, “The Strength Pareto Evolutionary Algorithm2 (SPEA2) applied to simultaneous multi-mission Waveform design*”.*
- Woodbury Robert, “Elements of parametric design”, London, Taylor and Francis, 2010.
- Zubin Khabazi, “Algoritmi Generativi con Grasshopper”,2010.

# 8. Ringraziamenti

In queste righe colgo l'occasione per ringraziare brevemente tutti coloro che mi hanno supportata e accompagnata per il raggiungimento di questo importante traguardo della mia vita.
Vorrei ringraziare, prima di tutto, il Professore Giuseppe Marano per la possibilità concessami nello svolgere questo interessante lavoro di tesi, un lavoro grazie al quale ho avuto modo di mettermi in gioco e approcciarmi a nuove e innovative tematiche di studio.
Ringrazio inoltre Laura Sardone per il supporto tecnico e la grande disponibilità dimostrata in questi mesi, per essere stata parte integrante dello svolgimento dell'attività pratica in modo professionale e competente.
Questo lavoro di tesi rappresenta solo il traguardo di un percorso durato anni, anni durante i quali ho avuto modo di crescere e scommettermi in un contesto formativo, come quello offerto dal Politecnico di Torino, oltre che in una nuova città.
Vorrei quindi ringraziare la mia famiglia per la fiducia che ha sempre posto in me e, ancor di più, per aver reso possibile questa esperienza umanamente e tecnicamente così importante, incoraggiandomi e supportandomi nonostante la distanza che ogni giorno ci ha separati.
Ringrazio le mie coinquiline Laura e Stefania per essere state al mio fianco condividendo ogni mia gioia e sconforto ma, soprattutto, per aver reso questi anni migliori, facendomi sempre sentire a casa.
Un altro grazie va alle mie amiche di sempre Eleonora, Paola, Marilia e Greta che con il loro affetto mi sono state moralmente vicine e al mio caro amico Stefano, un importante punto di riferimento in questi anni lontana da casa.
Per ultimo, ma non per importanza, ringrazio Lorenzo per essere riuscito, più di chiunque altro, a sostenermi e affiancarmi, grazie alla sua costante presenza, in questi mesi così "diversi", regalandomi, ogni giorno, un sorriso con la sua simpatia e la sua dolcezza.