ReLighting spaces

Addestramento della cognizione all'accesso alla luce diurna in assemblaggi spaziali combinatori tramite Reinforcement Learning

Tesi di Laurea in Architettura e Composizione architettonica III
Corso di Ingegneria Edile-Architettura
Dipartimento di Architettura
Alma Mater Studiorum - Università di Bologna
di Giuseppe Massafra - Relatore: prof. Alessio Erioli

Alma Mater Studiorum - Università di Bologna

Scuola di Ingegneria e Architettura
DIpartimento di Architettua
Corso di Ingegneria Edile-Architettura

Tesi di Laurea
in Architettura e Composizione architettonica III

ReLighting spaces

Addestramento della cognizione all'accesso alla luce diurna in assemblaggi spaziali combinatori tramite Reinforcement Learning

Di:
Giuseppe Massafra

Relatore:
Prof. Alessio Erioli

Anno accademico: 2023/24
Sessione I

Contents

Abstract

Questa ricerca studia un processo combinatorio per la generazione di assemblaggi spaziali con controllo adattivo su fattori legati al clima, come l'accesso alla luce diurna, attraverso l'uso del *Reinforcement Learning* (RL).

Nella progettazione combinatoria, un insieme finito di parti e regole per il loro accoppiamento e aggregazione iterativa, genera assiemi più grandi le cui proprietà, prestazioni e funzioni a diverse scale di sistema, differiscono da quelle delle parti costituenti; Il tutto è composto da moltitudini[1], generate dalle interazioni che avvengono reciprocamente tra le parti coinvolte. Un ruolo chiave nello stabilire il potenziale per l'emergere di qualità olistiche risiede nella progettazione di parti e regole e nella politica che ne definisce la selezione in relazione agli obiettivi di progettazione.

La progettazione combinatoria è definita da Sanchez come un processo intrinsecamente aperto, in cui non è possibile ottenere alcun tipo di ottimizzazione. Tuttavia, è possibile operare all'interno di questo contesto per realizzare sistemi le cui proprietà olistiche, legate agli aspetti quantitativi dello spazio architettonico, seguono criteri progettuali predeterminati. Nella loro ricerca combinatoria, sia Sanchez, Alexander (*A Pattern Language*) e Stiny (nel suo studio sulle *Shape Grammars*) utilizzano criteri fissi (sia stocastici che euristici) per le scelte di aggregazione; Una strategia diversa consiste nell'utilizzare una politica che derivi le regole di aggregazione da applicare a ogni iterazione sulla base di un obiettivo e delle conseguenze delle azioni compiute.

Da questo punto di vista, Makoto Sei Watanabe nella sua serie *Induction Cities* ha sperimentato modelli "induttivi": un'ampia gamma di configurazioni spaziali vitali generate attraverso un processo stocastico accoppiato a condizioni target selettive basate su fattori climatici come l'accesso diretto alla luce solare (*Sun God City).*

Basandosi sul lavoro di Watanabe, questo studio mira a legare la logica combinatoria con considerazioni topologiche e feedback climatico-ambientali, per la generazione adattativa al contesto di assemblaggi di unità spaziali con controllo sull'accesso alle ore diurne sulle superfici esposte. L'approccio utilizza agenti RL addestrati invece di algoritmi iterativi di scelta/euristici casuali per adattarsi alle condizioni di illuminazione nel processo di selezione e aggregazione delle parti. Gli agenti spostano e posizionano le parti (voxel) all'interno di uno spazio

/1
Per Bogost il concetto di moltitudine si differenzia da quello di intero poiché nel primo caso il significato emerge dall'accoppiamento di unità senza appartenere ad un sistema olistico più ampio: le unità possono essere descritte nella loro autonomia in una struttura più ampia piuttosto che come parti di un tutto.

voxelizzato, con l'obiettivo di garantire la coerenza topologica e un accesso alle ore diurne target sulle superfici esposte nell'assemblaggio finale.

La ricerca introduce l'adattività basata sulle condizioni in un processo combinatorio per mezzo dell'addestramento RL, andando oltre sia la scelta casuale che gli insiemi euristici predeterminati; Sebbene entrambi possano riguardare condizioni al contorno, sono rispettivamente non controllabili e legati a uno specifico scenario ambientale. Attraverso la politica addestrata dell'agente, il sistema apprende una relazione stato-azione-ricompensa in un processo di feedback continuo tra i dati spaziali, ambientali e climatici che si applica a qualsiasi configurazione ambientale che può essere codificata nei termini del sistema.

Lo studio è implementato accoppiando librerie Python allo stato dell'arte (Stable Baselines 3, Gymnasium) per la parte di RL, e l'ambiente Rhino+Grasshopper per la modellazione, il calcolo del fattore di luce diurna[2] e la visualizzazione, costruendo un'infrastruttura personalizzata per la comunicazione bidirezionale dei dati tra ambienti di calcolo durante le fasi di addestramento e inferenza.

/2
Nello specifico si utilizza il plug-in di Ladybug tools al fine dell'analisi del numero di ore di luce diurne.

01

1.1 Architettura come sistema
1.2 Architecture intelligence
1.3 Induction design
1.4 Climate base generative process

Background

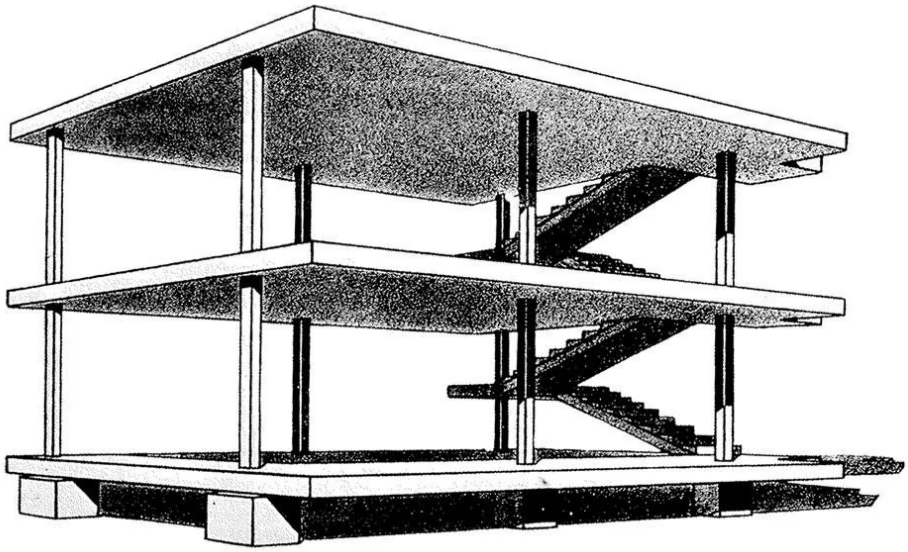
1.1 Architettura come sistema

Quando si parla di architettura spesso il discorso si catalizza su determinati modi di concepire, costruire e progettare legati alla concezione dell'opera unica e dell'architetto-creatore. In mezzo all'ampio panorama di approcci possibili, emergono due prospettive riguardo al modo di concepire l'architettura, il suo progetto ed il suo ruolo nella società, rappresentabili dai contributi di Bruno Taut e Le Corbusier. Il confronto tra il *Glaspavillon* e lo studio su la *Maison Dom-Ino*, infatti, pone in essere due concezioni antitetiche sulla natura dell'opera architettonica, nel primo caso intesa come artistica e totale, nel secondo, vista come un sistema.

Il *Glaspavillon*, eretto a Colonia nel 1914, è l'esempio archetipico dell'edificio inteso come opera d'arte singola e totale, in cui ogni elemento è attentamente regolato per definire un'unica, indivisibile ed inalterabile realizzazione. In questo contesto, l'opera trova valore nella sua unicità ed è legata ad una forma di costruzione artigianale in cui ogni pezzo è un prodotto unico fatto per avere uno specifico ruolo al suo interno. Ogni sua imitazione sarebbe considerata una copia e non contribuirebbe in nessun modo ad aumentare il valore percepito dell'originale.

Con la *Maison Dom-Ino*, Le Corbusier concepisce non un oggetto ma un sistema, figlio della logica della produzione industriale – forse il primo che in architettura ne coglie l'essenza – fatto per essere replicato a partire da un catalogo minimo di oggetti unici. Attraverso la replicazione combinatoria di 3 elementi (solai, pilastri e scale) si può accedere ad un numero potenzialmente infinito di configurazioni spaziali. L'architettura è trattata come una sorta di macchina che può essere facilmente adattata e moltiplicata. La copia stessa diventa la ragione di vita del sistema architettonico, aprendo la possibilità di adattamento al contesto, con esiti sempre diversi e accessibili. La *Maison Dom-Ino* dunque definisce uno dei primi modi in cui l'architettura

_f01
Glasspavillon - Bruno Taut 1914.



_f02
La Maison Dom-Ino - Le Corbusier 1914.

inizia ad essere vista come un sistema aperto, incompleto e adattabile. In questo caso il termine sistema può essere utilizzato nella sua accezione più generica in cui si configura come un insieme di parti che interagiscono tra loro permettendo a l'un l'altra di funzionare e formare un tutt'uno integrato. (Colchester, 2016)

Le premesse che portano allo sviluppo di questo genere di concezioni sono da ricercare all'interno delle tecniche produttive portate in auge dalla seconda rivoluzione industriale, dove l'attenzione si sposta molto verso gli effetti legati alle logiche di replicabilità. In ambito industriale esse erano utilizzate all'interno di processi economico-produttivi per provocare serialità e ridondanza. Un set di informazioni veniva condensato all'interno del singolo oggetto e riprodotto indefinitamente, generando risultati sempre uguali ed omogenei. D'altra parte, se il set di informazioni su cui si basa il sistema viene trasmesso in un processo in grado di replicarsi affrontando molteplici scenari, si può dar vita a sistemi in grado di adattarsi a contesti diversi e mutevoli.

_f03
Schizzo del progetto originale di Habitat 67 - Moshe Safdie.

Generare un'eterogeneità di risultati in grado di tener conto della complessità del contesto sociale in cui l'architettura si andava ad inserire, era una delle questioni che Moshe Safdie aveva affrontato con il suo progetto *Habitat 67* a Montreal, Canada. *Habitat 67* rappresentò un tentativo innovativo di creare un sistema abitativo prefabbricato e modulare, capace di accogliere le sfide legate alle esigenze di elevata densità urbana.

Safdie aveva compreso le problematiche che l'eccessiva espansione urbana stava provocando: esagerato consumo di suolo, energia e trasporti, erano tutti temi che potevano essere affrontati reinventando il concetto di abitazione urbana, cercando di offrire le qualità di vita di una casa con verde, privacy e accessibilità, anche all'interno delle città.

Il progetto originale per *Habitat 67* prevedeva moduli di appartamenti prefabbricati accatastati su strutture di 20 piani, con ogni unità disposta in modo tale da aver accesso ad un giardino. Le strutture dovevano librarsi sopra gli spazi pubblici sottostanti, con strade di collegamento poste ogni quattro piani. La disposizione dei moduli abitativi è stata studiata per definire una varietà di spazi esterni e interni, capace di conferire a ciascun appartamento un carattere unico e di rendere il sistema finale complesso ma organizzato (Walsh, 2023).
Nella realtà, con un budget di soli 15 milioni di dollari, il progetto realizzato è stato ridotto a 158 residenze, che rappresentano meno della metà della dimensione originale del piano generale.
L' altro aspetto che non ottenne un successo alla portata di quello de-

_f04
Ricostruzione del del progetto originale di Habitat 67 - Safdie Architects, Neoscape

_f05
Immagine del cantiere di costruzione di Habitat 67.

siderato riguarda i metodi di costruzione: il programma di produzione accelerato, testato per la prima volta in quell'occasione ha consentito di terminare i lavori in un periodo di circa 11 mesi ma ha visto lievitare i costi di realizzazione. Ciò che sarebbe dovuto essere prodotto in serie si è rivelato essere praticamente un campione realizzato a mano dal prezzo di oltre 100.000 dollari per unità (Jacobs, 2015).

Nonostante questo, l'idea di Safdie crea con successo per la prima volta un processo compositivo-spaziale combinatorio che fa leva su un sistema prefabbricato. Non si parla più di edifici ma di sistemi di architettura ad alta densità. "Non volevo costruire un edificio, volevo creare un sistema che potesse essere applicato a qualsiasi sito".(Murphy, 2016)

L'attenzione si sposta verso la gestione della relazione tra le parti in gioco oltre che sulla definizione delle stesse. In questo modo il progetto assume un forte carattere mereologico[4] nel senso che ci si interroga su come gli oggetti possono essere divisi in parti e come queste possono contribuire alla formazione di un insieme più ampio in relazione alle loro capacità. Il termine capacità fa riferimento ai modi in cui un oggetto può essere declinato o esercitato in relazione ad un altro. Sono le capacità [4] dei moduli che costituiscono il sistema a favorire l'emergere di proprietà nell'insieme. Allo stesso tempo tali capacità[5] rimangono potenziali fino a quando non vengono esercitate

/3
Il termine mereologia deriva dal greco *μέρος*, méros, "parte" e *-λογία*, logìa, "discorso", "studio". È utilizzato in filosofia e logica per descrivere la teoria delle parti e delle totalità.

/4
Le capacità di un oggetto dipendono dalle sue proprietà ma da esse si differenziano in quanto quest'ultime non hanno bisogno di essere specificate in relazione a qualcos'altro (De Landa, 2010).

_f07
Habitat 67 - Moshe Safdie.

_f06
Fotografia dell'ingresso di Habitat 67.

nel processo di aggregazione in cui si va a definire cosa può e cosa non può accadere tra le parti in gioco.

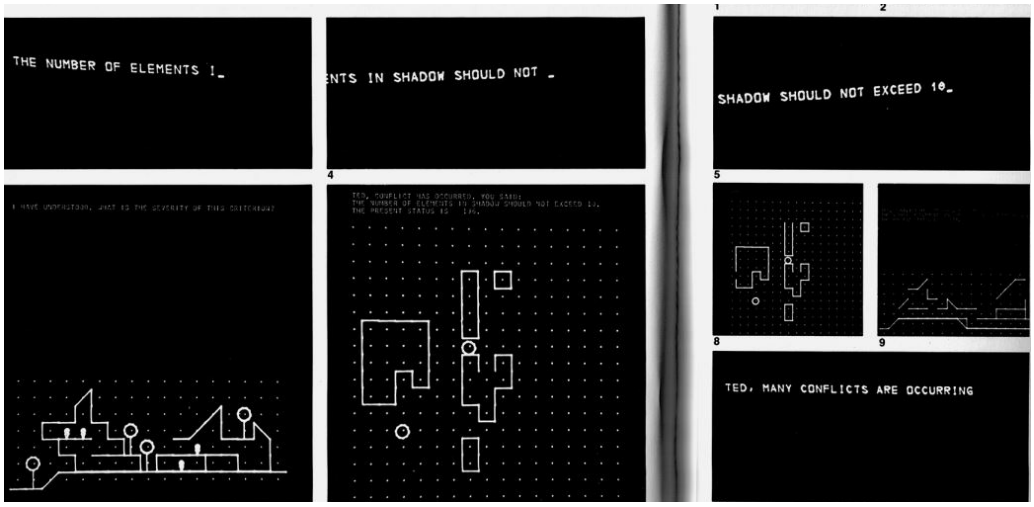
L'adozione di un approccio di tipo combinatorio fa sì che l'ordine generale non sia più il risultato di operazioni dirette o prestabilite, ma emerga organicamente attraverso i meccanismi di evoluzione del sistema stesso. I vantaggi che questo tipo di approccio propone sono molteplici: oltre all'aspetto puramente compositivo, progettare sistemi consente di adattarsi alla mutevolezza del contesto, consentendo di produrre una gamma di configurazioni potenzialmente infinita e scalabile a qualsiasi dimensione. Inoltre sistemi di questo genere si prestano bene alle tecniche di prefabbricazione, le quali possono essere implementate con vantaggi significativi in termini di utilizzo e gestione delle risorse in architettura.

1.2 Architectural intelligence

Tra le caratteristiche principali del metodo implementato da Watanabe per lo sviluppo delle Induction Cities spicca il suo approccio cibernetico all'architettura. Coniato da Norbert Wiener nel 1948 il termine cibernetica deriva dal greco kybernets, timoniere e inquadra lo studio della struttura, i processi di controllo, la comunicazione e i feedback all'interno di sistemi complessi, viventi e artificiali. L'architetto giapponese non fu il primo a pensare alla progettazione in questi termini: il rapporto tra architettura e cibernetica, infatti, iniziò ad assumere sempre più rilevanza negli anni '60 quando diversi architetti (tra cui Alexander, Wurman, Price, Negroponte etc..) con le loro ricerche delinearono nuovi modi di pensare alla progettazione, che aprirono la strada verso la ricerca sull'intelligenza artificiale in architettura (Steenson, 2017).

Nel 1969 l'articolo "*The Architectural Relevance of Cybernetics*" di Gordon Pask, suggeriva come la cibernetica avrebbe potuto influenzare le dinamiche di interazione tra il progettista e il sistema su cui lavorava. Ciò avveniva ridefinendo il pensiero sull'interazione tra spazi costruiti e utenti, in una maniera simile a quella che aveva approfondito Alexander in "*A Pattern Language*". Alexander infatti si focalizzava sul visualizzare sistemi architettonici e urbani come entità organiche e adattive, dove la progettazione avveniva attraverso l'interazione di elementi che si auto-organizzavano secondo principi e schemi ricorrenti. Il suo obiettivo era comprendere e gestire la complessità dei sistemi attraverso regole modulari e ripetibili, facilitando l'auto-organizzazione e l'adattamento degli stessi.

L'altro modo in cui l'approccio cibernetico avrebbe potuto influenzare il modo di leggere l'architettura era promuovendo concetti di adattabilità, interattività e risposta all'ambiente attraverso processi di feedback. Argomenti legati all'apprendimento e all'auto-riproduzione dei sistemi, che hanno avuto un ruolo fondamentale nell'estendere il campo della cibernetica a quello dell'intelligenza artificiale.



_f08
Interfaccia interattiva permetteva un diaolgo domanda e risposta tra l'utente e URBAN 5.

Fondamentale per la ricerca in questo senso fu il lavoro del MIT Architecture Machine Group in cui particolare rilievo ebbe la figura di Nicholas Negroponte, determinante nella ricerca sulle "Architecture machines".

Queste rappresentavano dispositivi pensati per trasformare il processo di progettazione in un dialogo, alterando la dinamica umano-macchina tradizionale e incorporando l'intelligenza artificiale per affrontare le procedure di progettazione. *URBAN 5*, ad esempio, consisteva in uno strumento di progettazione che aveva l'obiettivo di assistere il suo utente sviluppandone un modello cognitivo attraverso la conversazione. Il sistema funzionava su un computer che consentiva agli utenti di manipolare dei cubi con una penna ottica all'interno di un'ambiente che simulava un contesto urbano in cui era possibile svolgere attività di pianificazione semplificate.
Gli utenti di *URBAN 5* disegnavano e selezionavano quadrati su uno schermo e assegnavano modalità o attributi a ciascun blocco usando un set di trentadue pulsanti. Man mano che l'utente operava all'interno dell'ambiente il sistema poneva domande programmate al fine non solo di aiutarlo a portare a termine i task che stava svolgendo, ma anche di evidenziare gli eventuali errori che stava commettendo.
Seppure i limiti tecnologici non consentirono a Negroponte di raggiungere esattamente il risultato che avrebbe voluto, rimane l'idea innovativa di inseguire un modello di intelligenza degli strumenti che si identificava nella possibilità di dialogare, interagire e anche contrapporsi alle decisioni progettuali.
In termini architettonici ciò significava pensare ad un edificio come molto più che un'entità statica e inquadrare l'architettura come un processo interattivo di scambio delle informazioni. Proprio in quegli anni Richard Saul Wurman aveva enfatizzato la rilevanza del tessuto informativo che fa parte dello spazio costruito: il concetto di architet-

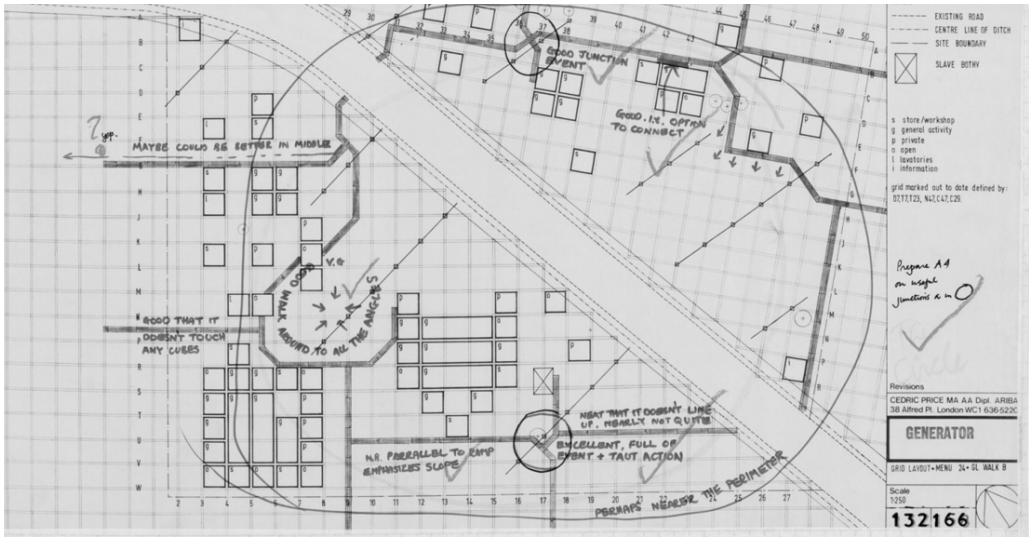
_f09
La postazione per l'utente di URBAN 5.

tura dell'informazione[5] sottolinea il bisogno di ambienti costruiti che non solo si adattino fisicamente ai loro fruitori ma che fungano anche da interfaccia intelligente tra utenti e la mole di dati generati dagli stessi spazi abitativi.

Temi che Cedric Price aveva perfettamente inquadrato nel 1965 con la sua ricerca su *The Generator Project*, un complesso, reattivo ai suoi fruitori, ideato per ospitare attività aziendali, eventi culturali e residenze di artisti.
Più che come un edificio, *Generator* era stato concepito come un sistema capace di recepire e memorizzare ogni comportamento dell'utente, permettendo a lui di interagire con lo spazio per modificarlo a proprio piacimento, e a se stesso di apprendere quali fossero le tendenze degli abitanti per anticiparne le mosse. Nel concreto ciò avveniva grazie a un sistema di unità mobili prefabbricate che potevano essere spostate, rimosse o sostituite per mezzo di gru, azionate da un operatore. (Steenson, 2017)

/5
Richard Saul Wurman ha inquadrato con grande lucidità la relazione tra architettura e informazione mettendo in luce come l'organizzazione dell'ambiente costruito costruisce un substrato di informazione che fa sì che esso possa essere interpretato dai suoi fruitori.

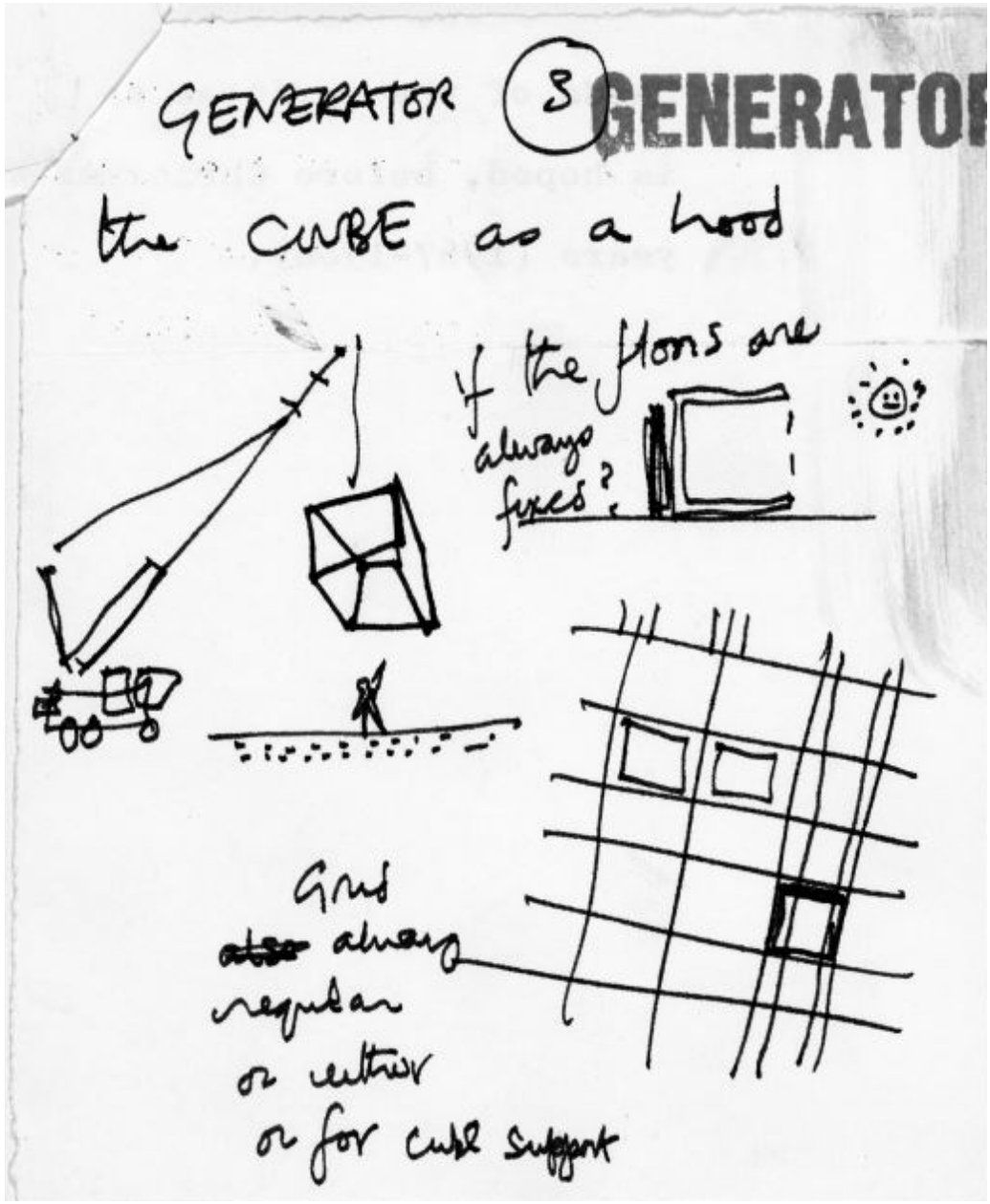
Con questo schema, *Generator*, esplorava la nozione di intelligenza artificiale, in cui il l'ambiente stesso diveniva un artefatto intelligente. Non contava più l'edificio come forma costruita o come oggetto, ma il processo di feedback utente-spazio che crea un'architettura fluida nel tempo.
L'aspetto importante in questo caso è come il sistema abitativo stesso diventa partecipante attivo al proprio cambiamento, non



_f10
Il layout a griglia di Generator.

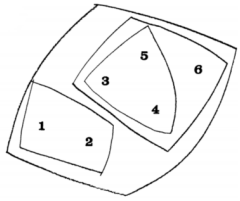
limitandosi a soddisfare i propri utenti, ma proponendo modifiche di propria iniziativa. Attraverso la cosiddetta "boredom function", se veniva rilevata inattività o mancanza di variazioni nell'organizzazione dello spazio per un periodo prolungato, venivano suggerite configurazioni alternative per stimolare l'interazione e la creatività degli utenti.

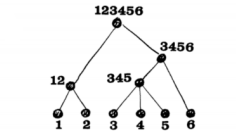
Con *Generator* si apre a un nuovo modo di intendere l'architettura e la tecnologia al suo interno: se la progettazione tradizionale ha come fine quello di compiere tutte le operazioni necessarie per definire la forma finale e stabile di un determinato progetto, qui l'obiettivo è proprio il cambiamento, risultato di un ciclo di feedback continuo tra spazio e ambiente, che genera in ogni istante un nuovo stato del sistema. È importante sottolineare come in questo caso il lavoro di Price si basa su due concetti fondamentali: per cominciare, la ricerca di un'architettura intelligente viene condotta attraverso un sistema spaziale più che di un edificio in senso tradizionale. Per poter rispondere all'interazioni degli utenti, *Generator* è stato concepito come un reticolo di blocchi che potevano essere spostati con l'ausilio di gru e collegati tra loro attraverso dei percorsi. Questo rende possibile sia quella flessibilità al cambiamento che quella despecializzazione dello spazio, necessaria per l'adattamento ad usi che siano molteplici.
In seconda battuta, come feedback per l'apprendimento viene scelto l'utente stesso il quale attraverso le proprie scelte, comportamenti e decisioni, offre dei dati che l'edificio può apprendere per auto organizzarsi.



_f11
In questo schizzo si ritrova il lavoro di Price su alcuni deglie elementi di Generator come la gru mobile e i supporti per i blocchi cubici.

_f12
Fotografia dall'alto della zona urbana in prossimità della Tokyo. Bay Area.





/6
La struttura ad albero è un termine che si riferisce alla teoria delle strutture di insiemi. In particolare Alexander definì la struttura ad albero come segue: "Una raccolta di insiemi forma un albero se e solo se, per due insiemi qualsiasi che appartengono alla raccolta, o uno è interamente contenuto nell'altro, oppure sono interamente disgiunti" (Alexander, 2019).

_f13
Diagramma di rappresentazione di un insieme con struttura ad albero - Christopher Alexander.

1.3 Induction Design

Il concetto di "*Induction Design*" è stato formulato nel 1990 dall'architetto Makoto Sei Watanabe, il cui intento di ricerca si focalizzava sulla definizione di una metodologia operativa per la creazione di sistemi urbani.

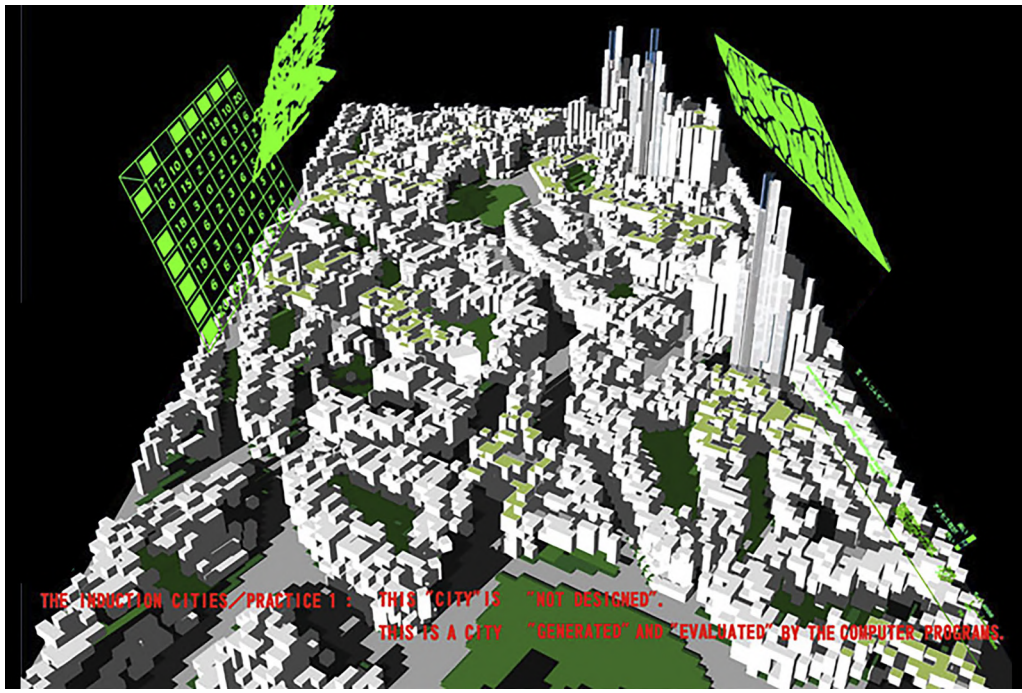
Presupposto dei suoi studi era l'assenza di un set soddisfacente di criteri (supportati da strumenti metodologici e teorici) che potesse avere un ruolo decisivo nel processo di costruzione della città.
Le problematiche legate all'approccio a questo tema, all'interno dello scenario di espansione urbana delle principali città giapponesi negli anni 90, erano evidenti in una serie di episodi in cui le scelte di pianificazione sembravano essere determinate da principi prevalentemente commerciali. Nello specifico il frutto della gestione dell'espansione urbana di grandi città come Tokyo era stato una serie di quartieri divisi in griglie ordinate, con file di isolati separati in base alla funzione, colmi di edifici totalmente estranei tra loro e intrecciati con passaggi pedonali.
In altre parole le attività di pianificazione producevano entità vicine al concetto di "*artificial cities*" definito da Christopher Alexander.
Città concepite rigidamente dalle mani di designers con una struttura ad albero[8], non in grado di accogliere la complessità generata dalla sovrapposizione dei diversi sistemi (fisici, sociali, spaziali, economici etc..) che la compongono. (Alexander, 2019)
La città, secondo Watanabe, doveva essere considerata come un siste-

ma in continuo cambiamento, formato da una serie di eventi interconnessi che si sviluppano in un network di relazioni tra persone e luoghi che si evolve nel tempo. Il suo processo di costruzione di conseguenza, doveva essere affrontato come un progetto per creare un vasto sistema e non come un disegno concettuale, plasmato dalle mani di un progettista la cui intenzionalità, seppur legata a determinati fondamenti teorici, tendeva a trattare un problema aperto come uno chiuso e controllabile.

Costruire il paesaggio urbano come un sistema aperto significava trovare un metodo per generare delle configurazioni spaziali parziali, incomplete e adattabili. Watanabe si poneva di raggiungere questo obiettivo partendo da premesse che inducessero ad una molteplicità di risultati possibili.
Le premesse in questione erano null'altro che le condizioni imposte su specifici dati fisici e climatici sotto cui far scaturire il processo di induzione con cui si sviluppava il sistema. Fattori come l'illuminazione naturale, l'esposizione al vento e la temperatura si prestavano in maniera adeguata a questo metodo in quanto analiticamente misurabili e assoggettabili a verifica per mezzo di algoritmi. "Ho selezionato luce sufficiente, brezze piacevoli, strade su cui è un piacere camminare, una topografia adeguatamente ascendente e discendente e varie funzioni disposte in rapporti ottimali. Poi ho iniziato a lavorare su programmi informatici per generare città che soddisfacessero meglio queste condizioni". (Watanabe, 1994)

Operativamente lo scopo non era quello di creare forme complesse, ma di applicare un metodo in cui molteplicità di esiti causata dalla ripetizione di forme semplici, generasse sistemi spaziali praticabili. I layout che se ne creavano erano diverse possibili forme capaci di soddisfare le condizioni richieste.
Fondamentale è come gli obiettivi di adattabilità del sistema potevano essere raggiunti solo focalizzandosi su una cerchia ristretta di parametri per volta, in modo da poter analizzare l'evoluzione del sistema all'interno di simulazioni e cogliere le relazioni tra le condizioni imposte e le interazioni che assumevano le varie parti all'interno del sistema.
In questo modo era possibile determinare che tipo di ricadute avesse la gestione delle condizioni su dati differenti (Illuminazione naturale in *Sun God City*, esposizione al vento *Wind God City*, etc..), in termini di qualità dello spazio architettonico.

Watanabe mette in evidenza come la sua metodologia acquisisce senso proprio nel momento in cui viene specificato l'insieme di fattori tramite cui far partire il processo di induzione: la possibilità di indurre diversi sistemi scaturiti da condizioni di tipo differente non implica la preferibilità di uno rispetto all'altro. Questa dipende dagli obiettivi di progetto, specificati durante un processo critico decisionale che è

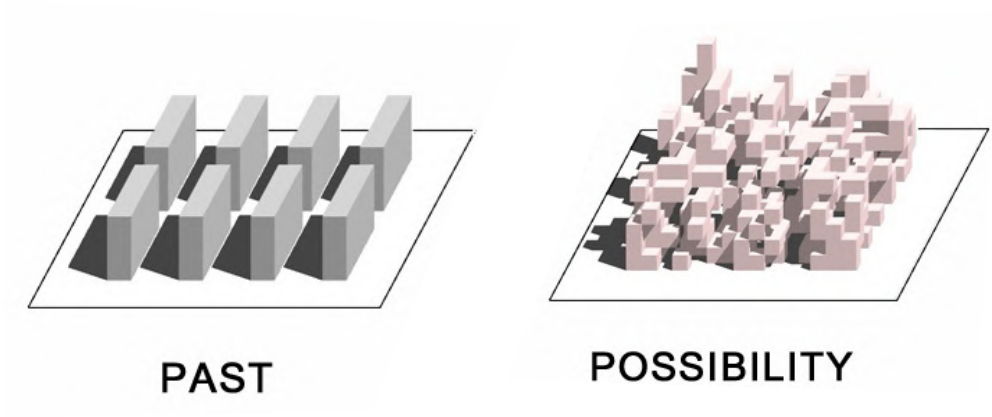


_f14
Induction cities series visualization - Makoto Sei Watanabe.

singolo per ogni determinato contesto.

Tra gli esperimenti più significativi troviamo *Sun God City*. Qui Watanabe parte dalle pratiche costruttive per edifici multi-unità in Giappone. Seppure l'aspetto legato alla massimizzazione dell'illuminazione naturale fosse di importanza condivisa nella tradizione architettonica giapponese, la ricerca di soluzioni che tenessero conto in maniera tangibile di questo aspetto non era evidente se non in rari casi. Infatti, in grandi metropoli come la già citata Tokyo, si assisteva maggiormente alla presenza di complessi residenziali monotoni, caratterizzati da edifici di qualità discutibile e di forma prettamente scatolare.

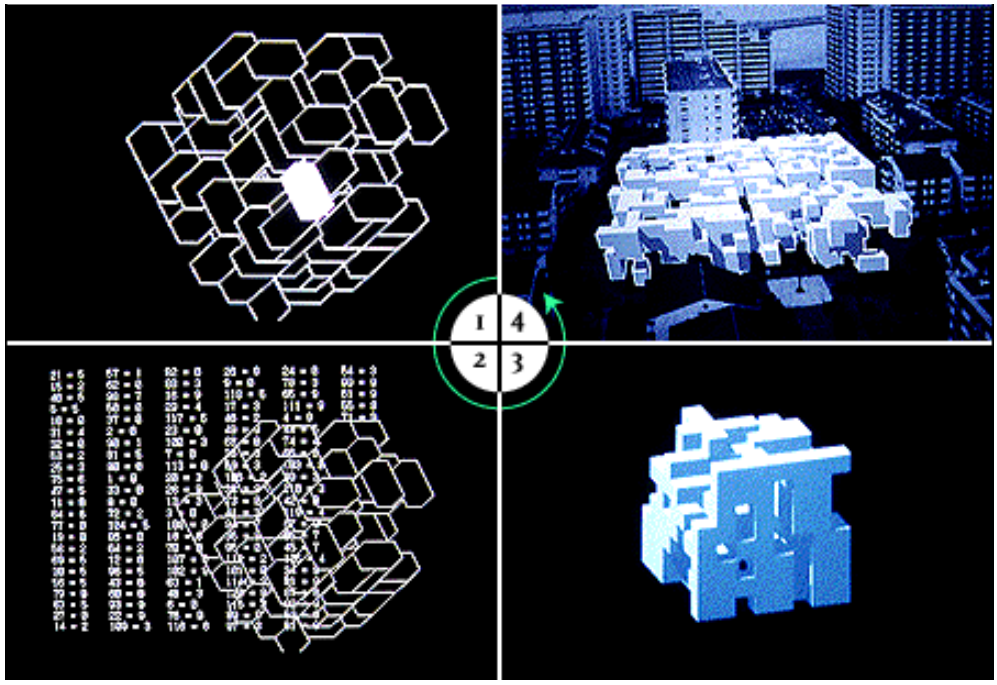
Sun God City era il tentativo di avviare un cambiamento, ipotizzando di trovare un metodo progettuale alternativo che tenesse in considerazione le stesse condizioni al contorno (massima luce solare per unità abitativa) ma, allo stesso tempo, potesse offrire una molteplicità di configurazioni spaziali praticabili. Fissando un numero n di ore di luce naturale per ogni unità abitativa e la densità di suolo da edificare, il sistema genera assemblaggi spaziali differenti, ottenuti attraverso delle operazioni di sottrazione volumetrica iterate e rispondenti analiticamente ai criteri di input. (Watanabe, 1995)
"La soluzione che abbiamo scoperto suggerisce uno spazio esterno riccamente vario con diverse rientranze, fornendo nuove relazioni tra la funzione da un lato e lo spazio (ad esempio, aperture, cortili, percorsi, terrazze e spazi comuni.) dall'altro."
Sebbene i limiti tecnologici ponessero un margine agli algoritmi



_f15
Diagramma di Sun God city - Makoto Sei Watanabe.

implementati da Watanabe, perlopiù basati su iterazioni randomiche, il suo lavoro mette in luce come i fattori climatici possano svolgere un ruolo fondamentale all'interno del processo compositivo. Non solo si mette in risalto la loro importanza come parametri per il giudizio e orientamento alle scelte progettuali, ma anche le ricadute dal punto di vista volumetrico e topologico che la loro gestione può offrire nel generare sistemi spaziali.

_f16
Sun God city - Makoto Sei Watanabe.



_f17
Interfaccia dell'algoritmo alla base di Sun God City - Makoto Sei Watanabe.

1.4 Climate based generative process

Pur non essendo stato costruito, *Generator* inquadra per la prima volta il concetto di architettura intelligente. La possibilità di apprendimento e di generare cambiamento da parte dello stesso spazio che la definisce sono i concetti che riassumono questa nozione.
Tuttavia, diverse considerazioni possono essere fatte in merito alla sorgente dell'apprendimento.
Se ad esempio il tipo di feedback fosse differente? Se questo processo di apprendimento si basasse su dati diversi, che tipo di sistema si potrebbe ottenere?

Definito un insieme di parti potremmo far sì che le regole di interazione tra esse tengano conto di dati di tipo ambientale o climatico. Esposizione al vento, radiazione solare, esposizione luminosa, sono tutti dati legati più o meno direttamente alle qualità architettoniche di uno spazio.
La presenza di una determinata quantità di luce naturale può rendere uno spazio più o meno luminoso e caldo, l'esposizione ad un certo tipo di vento può favorire la ventilazione naturale e allo stesso tempo avere ricadute di tipo acustico sullo spazio vissuto. In altre parole, questi dati concorrono a qualificare lo spazio architettonico sia nei suoi aspetti percettivi che in quelli prettamente tecnici.
Dunque, inglobare questi dati come feedback per il processo di generazione di un'architettura, significa far sì che il sistema possa apprendere dal suo auto-generarsi informazioni che possono essere catalizzate per rendere le qualità architettoniche, che ad essi sono associabili, adattabili a condizioni ambientali di progetto ritenute più opportune, al termine di un processo critico decisionale che è specifico per ogni singolo caso.
Non è intento di questo ragionamento, infatti, stabilire rigidamente cosa possa essere più o meno preferibile in una specifica situazione. Le "condizioni" scelte inizialmente possono essere associabili ai compor-

tamenti dell'utente di Generator: il sistema non si cura delle motivazioni della scelta dell'utente, ma solo di apprendere un modo per poterle soddisfare assieme a lui.

A questo punto è bene fare una precisazione: sebbene il modello di *Generator* offra un'idea di architettura nuova, allo stesso tempo pone problematiche nella sua realizzazione. Le idee ambiziose di uno spazio in continuo cambiamento impedimenti di diverso genere, costruttivi, economici e manutentivi, che portano a renderne la realizzazione effettiva quanto meno difficoltosa, se non in casi esigui e particolari.
I limiti realizzativi attuali non cancellano le opportunità del concetto di architettura intelligente. Gli spunti di generator possono allora essere ricalati ad esempio nel solo processo compositivo di un sistema spaziale.
Questo è l'intento di questa ricerca.
La capacità di apprendimento del sistema, fa sì che esso acquisisca una cognizione sulle ricadute che ogni suo stato abbia sui dati ambientali legati alla luce naturale, e di conseguenza sulle sue stesse qualità.
In questo modo il modello accumula informazioni utilizzabili per generare una configurazione del sistema che può essere considerata definitiva, ma al termine di un processo in cui ambiente e architettura sono simbiotici, alimentandosi l'un l'altro per dar vita ad una delle possibili diverse soluzioni che soddisfi gli obiettivi iniziali.
Potremmo pensare a questo approccio come alla possibilità di sfruttare tutta l'esperienza accumulata dal nostro modello per produrre delle istanze più compatibili ad un determinato tipo di ambiente, in cui l'apprendimento stesso del modello avviene.

02 Metodo

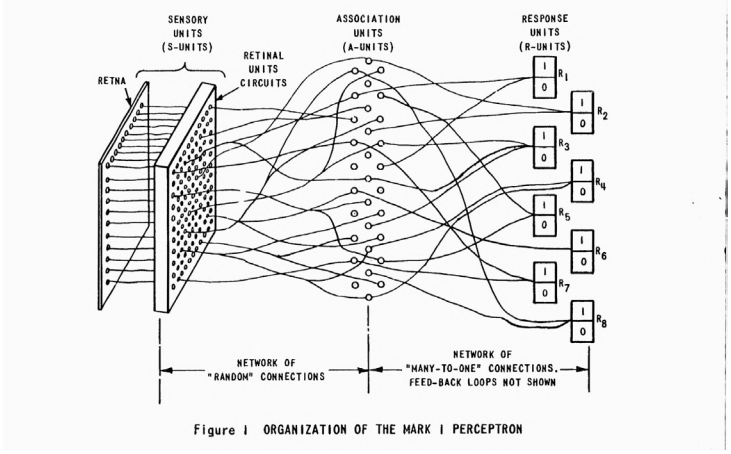
2.1 Artiicial Intelligence
2.2 Reinforcement Learning
2.3 RL in architecture

2.1 Artificial Intelligence

Nel 1956 l'informatico John McCarthy in occasione di una conferenza al Dartmouth College conia il termine artificial intelligence (AI) definendola come "la scienza e l'ingegneria della creazione di macchine intelligenti, in particolare di programmi informatici intelligenti. Si tratta di un compito simile a quello di utilizzare i computer per comprendere l'intelligenza umana, ma l'AI non deve limitarsi a metodi biologicamente osservabili." (IBM, 2021). Malgrado lo stesso McCarthy ammise di aver definito il termine come espediente per raccogliere fondi per finanziare una ricerca universitaria, esso apriva all'idea di un campo di studio in cui le macchine potessero emulare il pensiero umano.

Sebbene l'AI sia talvolta percepita come un campo emergente, le sue radici possono essere rintracciate ad una lunga storia profondamente legata all'analisi statistica. La sua evoluzione come disciplina è fatta di molteplici lignaggi, ma una prima generale distinzione sui filoni di ricerca che l'hanno definita sta nella differenza tra *Symbolic AI* e *Cognitive AI*.
L'intelligenza artificiale simbolica è stata il paradigma dominante della ricerca sull'AI dalla metà degli anni '50 sino alla metà degli anni '90 e si concentra sulla rappresentazione della conoscenza e del ragionamento attraverso simboli e regole logiche. Durante questo periodo sono stati studiati e sviluppati sistemi basati sulla conoscenza simbolica (in particolare sistemi esperti[7]) in grado di risolvere problemi legati alla semantica e alla logica. Nonostante questo le complicazioni relative a difficoltà nell'acquisizione della conoscenza e della fragilità della gestione di problemi fuori dominio, seguite dall'aumento della

/7
Nel campo dell'intelligenza artificiale i sistemi esperti sono dei sistemi computerizzati in grado di emulare le abilità di decision -making di umani esperti (con una profonda competenza e conoscenza).



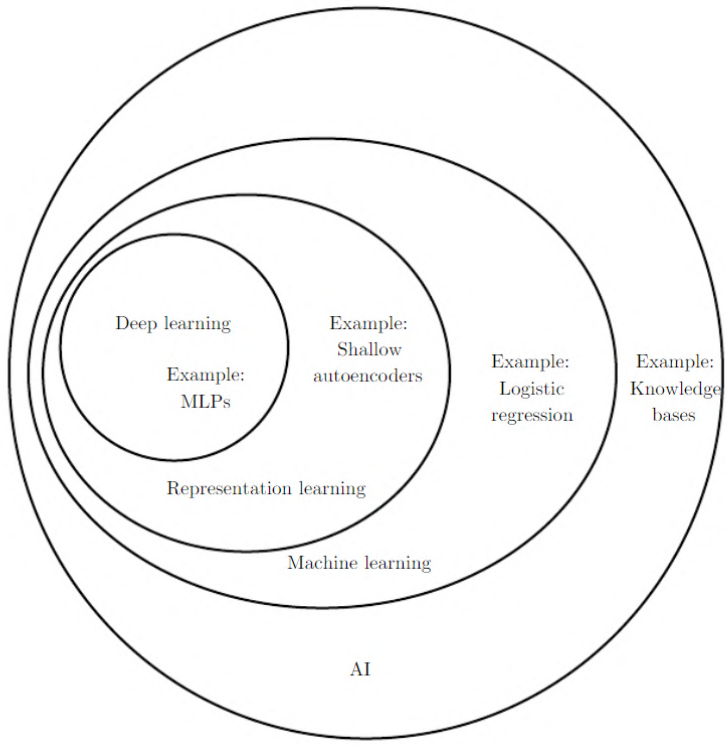
_f18
Il diagramma di funzionamento del Perceptron di Franck Rosenblatt.

potenza di calcolo e alla disponibilità di grandi quantità di dati, hanno riacceso l'interesse verso la Cognitive AI a partire dal 2012. L'intelligenza artificiale cognitiva si basa su modelli neurali e tecniche di apprendimento automatico per rappresentare la conoscenza.

La descrizione della prima rete neurale risale al 1957 attraverso il lavoro di Frank Rosenblatt.

Il suo programma *Perceptron* fu fondamentale non tanto per la creazione di dispositivi per l'intelligenza artificiale ma più che altro come occasione per lo studio dei principi neuro dinamici e delle strutture fisiche su cui si fonda l'intelligenza naturale.

Perceptron, era infatti un modello cerebrale artificiale, tradotto in una macchina neurale capace di imparare a riconoscere forme diverse ispirandosi alla struttura dei neuroni dell'occhio. Fu già all'epoca che lo stesso Rosenblatt aveva inquadrato le potenzialità di questa disciplina oltre che alcuni dei suoi futuri campi di applicazione: "Ci si aspetta che dispositivi di questo tipo alla fine siano in grado di formare concetti, tradurre lingue, confrontare informazioni militari e risolvere problemi attraverso la logica induttiva"(Rosenblatt, 1961).



_f19
Il diagramma di Venn in figura mostra le relazioni e le interesezioni tra le sotto discipline che definiscono L'AI come branca di studio

AI Domains

L'AI è un campo di studi che combina l'informatica e l'analisi statistica a robusti data set per creare sistemi in grado di apprendere dai dati e eseguire previsione e azioni basate sulla conoscenza acquisita per raggiungere obiettivi specifici.
Essa è il dominio più ampio che comprende diverse sotto discipline, tra cui particolare rilevanza assumono il Machine Learning (ML), il Representation Learning e il Deep Learning (DL) in ordine crescente di specificità, come illustrato all'interno del diagramma di Venn (Goodfellow, Bengio and Courville, 2016).

Il Machine Learning si focalizza su algoritmi che consentono a sistemi informatici di apprendere dai dati con lo scopo di effettuare previsioni e classificazioni. In questo contesto il ML restituisce dei modelli statistici, o più precisamente si pone come metodo per la automatizzazione della modellazione statistica di data set attraverso algoritmi di apprendimento (Pasquinelli, 2020).

All'interno del ML il Representation Learning si occupa di trovare metodi efficaci per la rappresentazione dei dati, al fine di facilitare l'apprendimento delle caratteristiche necessarie a descriverli meglio in relazione alla tipologia di problema affrontato. Un esempio da questo

punto di vista sono le reti convoluzionali CNN, un tipo di rete neurale utilizzata nel campo del riconoscimento delle immagini e della visione artificiale, profondamente efficace per il trattamento di dati con una struttura a griglia, come immagini e segnali audio (Goodfellow, Bengio and Courville, 2016).

Il Deep Learning (DL) è un'ulteriore specializzazione del Representation Learning. Ciò che qualifica il Deep Learning riguarda le modalità di apprendimento da parte dei modelli. Il Deep Learning implica l'utilizzo di reti neurali profonde (composte da più di tre strati): queste vengono utilizzate per automatizzare il processo di estrazione delle caratteristiche determinante per comprendere le differenze tra i dati di input. In altre parole attraverso l'uso di reti neurali il modello può acquisire dati non strutturati nella loro forma grezza e può determinare automaticamente la gerarchia delle caratteristiche che distinguono le diverse categorie tra loro. Il DL sfrutta reti neurali profonde di diversa tipologia come le Reti Neurali Ricorrenti, Reti Generative Avversarie (GAN), Reti Trasformatori, che hanno contribuito a rafforzare l'efficacia su applicazioni correlate al campo dell'IA come Visione artificiale, La Natural Language Processing (NLP) e il Riconoscimento Vocale.

Supervised learning, unsupervised learning, Reinforcement Learning

Gli algoritmi di ML vengono utilizzati per scopi come la previsione e la classificazione. Le tecniche di apprendimento comprese all'interno del dominio del ML sono solitamente distinte in tre categorie: Supervised Learning, Unsupervised Learning e Reinforcement Learning.

I modelli di supervised learning sfruttano dataset etichettati (classificati) al fine di cogliere le correlazioni esistenti tra gli elementi dei dati che producono in output previsioni corrette. Il concetto di supervisione è definito dunque dal fatto che il modello viene addestrato con dati in input le cui caratteristiche sono note e determinanti per il corretto processo di apprendimento da parte del modello. Modelli di questo genere vengono utilizzati perlopiù per fini che riguardano le operazioni di regressione (previsione di un valore numerico) come ad esempio il calcolo del prezzo di un immobile dati numero di stanze e location e di classificazione (previsione di appartenenza ad una categoria) come l'identificazione di mail spam a partire dal testo contenuto nell'oggetto e dalla sintassi.

Diversamente i modelli di Unsupervised Learning effettuano previsioni a partire da dataset non etichettati. L'obiettivo di un modello di questo genere è dedurre ricorrenze o pattern significativi all'interno dei dati di input. È comunemente usato per le attività come il clustering (identificazione di somiglianze tra insiemi) o la riduzione della

dimensionalità (riduzione del numero di variabili) (Google, 2018).

Il Reinforcement Learning si basa invece su una metodologia che utilizza algoritmi o "agenti" per navigare nel processo decisionale sequenziale all'interno di un ambiente che fornisce feedback. Questo approccio è particolarmente adatto per definire politiche e comportamenti ed è parte di quello che comunemente viene chiamato agent Agent-Based-Modeling (ABM) (Royal Society, 2017).

L'Agent-Based Modeling più in generale può essere inteso come un metodo di studio per sistemi basato sulla simulazione computazionale di azioni e interazioni tra agenti autonomi all'interno di uno specifico ambiente. Gli agenti sono addestrati con regole semplici che ne governano il comportamento solitario e reciproco nel processo di simulazione che viene esaminato per rivelare i pattern o gli schemi ricorrenti che definiscono il sistema (Colchester, 2016).

2.2 Reinforcement Learning

Il Reinforcement Learning (RL) è un framework utilizzato per risolvere problemi decisionali costruendo agenti in grado di interagire con uno specifico ambiente e di ricevere ricompense come feedback per comprendere l'efficacia del loro operato, al fine di migliorare la propria strategia decisionale nel tempo. L'idea è quella di trasporre il modo di apprendere degli esseri umani e animali in termini computazionali, specie attraverso l'utilizzo di reti neurali.

Reinforcement Learning background

Le radici del RL sono da ritrovare all'interno delle prime teorizzazioni del concetto di apprendimento per tentativo ed errori offerta da Edward Thorndike agli inizi del Novecento. La teoria nasce dall'idea, di matrice psicologica, che le azioni, quando seguite da risultati positivi o negativi, abbiano la tendenza ad essere riselezionate e modificate di conseguenza. Questa viene chiamata "*Legge dell'effetto*" dallo stesso Thorndike perché descrive l'effetto del rinforzo degli eventi sulla tendenza a selezionare le azioni (Sutton and Barto, 2014).
La legge dell'effetto racchiude in se i due aspetti che qualificano il RL, ossia che esso è selettivo in quanto implica provare alternative e selezionare tra esse testando le conseguenze di ciascuna e associativo, nel senso che le alternative trovate nella selezione sono associate a situazioni particolari (Sutton and Barto, 2014).

A partire da questo concetto l'evoluzione della ricerca nel RL è passata per diversi step che partono dall'introduzione della nozione di Reinforcement Learning in "*Steps Toward Artificial Intelligence*" (Minsky, 1961) che discuteva diverse questioni rilevanti per il RL, incluso il problema dell'assegnazione dei crediti che tratta come distribuire credito (o ricompense) per il successo tra le molte decisioni che potrebbero essere state prese.
Da allora diversi filoni del RL hanno affrontato diverse problematiche

concettuali legate alla disciplina. Queste furono completamente riunite nel 1989 con lo sviluppo del Q-Learning[8] da parte di Chris Watkins. Il Q-Learning segna il punto di inizio degli algoritmi di RL classico che precedono l'utilizzo di reti neurali all'interno del Deep Reinforcement Learning[9] (Deep RL), sviluppato principalmente dal team di ricercatori di DeepMind guidato da David Silver nel 2014. Tra gli algoritmi di Deep RL troviamo PPO (Proximal Policy Optimizazion) utilizzato all'interno di questa ricerca, e che sarà descritto in dettaglio nel capitolo successivo..

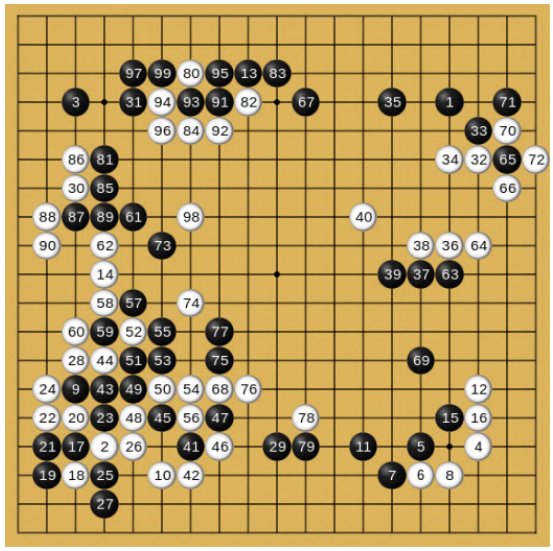
Il vasto panorama dei campi di applicazione del RL comprende la robotica, il gaming, la finanza e molti altri settori. Uno dei risultati che già nel 2016 ha dimostrato le enormi capacità di questo metodo è stato *AlphaGo* di DeepMind, un programma basato sul RL che ha sconfitto il campione mondiale di Go Lee Sedol. Il modello alla base di *Alpha Go* è stato allenato in un primo step attraverso supervised learning con un'innumerevole quantità di partite di Go giocate ad alto livello e sottoposto successivamente ad un processo di RL in cui ha utilizzato le conoscenza acquisite per giocare una vasta serie di partite contro se stesso e migliorare ulteriormente le sue strategie.

La mossa 37 con cui il modello sviluppato da DeepMind ha battuto l'allora campione mondiale è considerata iconica nel mondo del RL in quanto sembrava, agli occhi degli esperti del gioco, essere apparentemente erronea e banale, ma si è rivelata brillante e creative nonché dimostrativa di come le modelli di RL possano fare previsioni non intuitive e agire al di là delle consuete strategie umane (Silver et al., 2016).

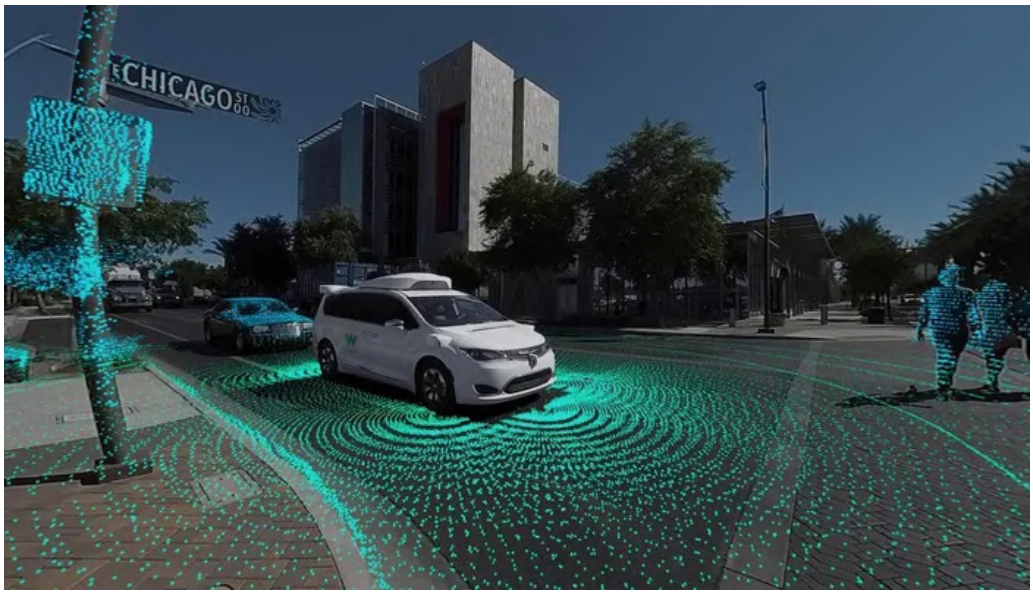
Un altro esempio è il sistema di guida automatica su cui si basano i veicoli *Waymo* creati da Alphabet. Questi sfruttano algoritmi di RL per elaborare i dati provenienti dai sensori installati a bordo e prendere decisioni in tempo reale sulla velocità, la direzione e il comportamento di guida da tenere.

/8
Il Q-Learning è un algoritmo RL che addestra una funzione Q codificata da una tabella contente tutti i valori delle coppie stato azione percorribili da un agente all'interno di uno specifico environment. Dato uno stato e un'azione la funzione Q identifica il valore corrispondente all'interno della suddetta tabella detta anche Q table. Durante il processo di addestramento l'agente esplora l'environment migliorando i valori delle coppie stato-azione di conseguenza. Ciò porta, finito l'addestramento, ad ottenere una funzione Q ottimizzata che ci porta a conoscere per ogni stato dell'ambiente l'azione migliore da intraprendere.

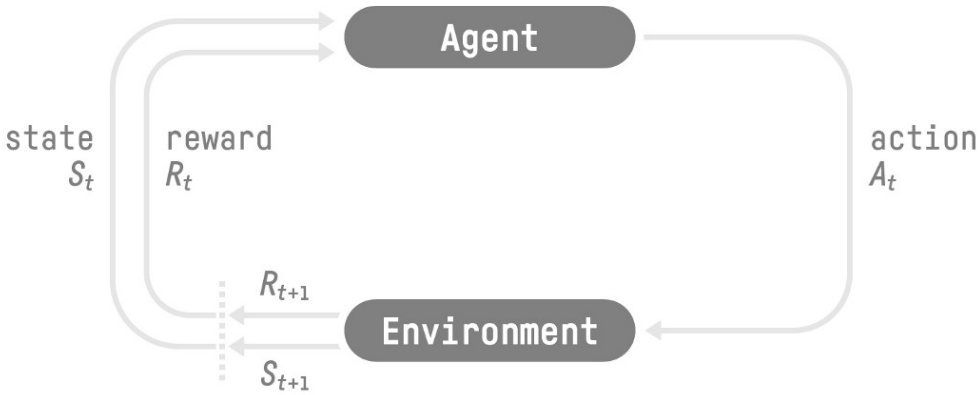
/9
Il Deep RL nasce dalle difficoltà computazionali nella rappresentazione tabellare su cui si basa il Q-Learning ad ambienti connotati da un elevatissimo numero di coppie stato. Questi algoritmi sfruttano l'utilizzo di reti neurali, che vengono addestrate per approssimare le funzioni che determinano le azioni ottimali da intraprendere in un determinato ambiente. Esistono diverse tipologie di algoritmi di Deep RL tra cui il Deep Q-Network (DQN), Deep Deterministic Policy Gradient (DDPG), Proximal Policy Optimization (PPO), Actor-Critic Methods, Twin Delayed Deep Deterministic Policy Gradient (TD3). (OpenAI, 2021).



_f20
La mossa 37 che ha portato Alpha Go a sconfiggere il campione Lee Sedol.



_f21
I veicoli autonomi Waymo di Alphabet.



_f22
Diagramma di Markov alla base del RL.

RL Framework

Il processo di RL avviene in cicli che producono una sequenza di *state* (stato), *action* (azione), *reward* (ricompensa), *next state* (stato successivo)[10].

Per *state* (St) si intende il set di informazioni (fornite all'agente sotto forma di *observation*[11] (osservazioni)) necessarie a descrivere l'assetto dell'ambiente in cui l'*agent* può compiere azioni. Si pensi all'esempio di un'agente di RL il cui obiettivo è apprendere strategie per vincere una partita di scacchi: lo stato in questo caso è rappresentato dalla posizione di tutte le pedine in un determinato istante della partita.
Per *action* (At) si intende invece una delle possibili interazioni che l'agent può decidere di compiere con l'*environment* (ambiente) ed in risposta a cui riceverà un *reward* o un *punishment* (punizione) a seconda dell'efficacia che ha prodotto. L'insieme delle possibili azioni che l'agent può compiere in un determinato environment è detto *action space*[12]. Se il nostro *agent* agisce in maniera da creare condizioni vantaggiose il *reward* (Rt) sarà positivo, viceversa, se la sua mossa ha un esito sfavorevole, riceverà una puniziona. In ogni caso l'*action* produrrà una modifica nell'ambiente e di conseguenza un nuovo stato del sistema (St+1) da cui rip artirà l'iterazione successiva del ciclo. Nel nostro esempio l'agente accumulerà tanta più ricompensa quanto più riuscirà a catturare scacchi dell'avversario e riceverà tante più punizioni quanto si metterà sotto scatto o si farà catturare pezzi importanti.
L'insieme di iterazioni successive costituisce un episodio, la cui lunghezza varia a seconda del tipo di *task* (compito) che l'*agent* sta compiendo. Nell'ambito degli scacchi l'episodio si può considerare terminato nel momento in cui termina la partita.
Fondamentale è come tutto il processo di RL si basa sul fatto che l'obiettivo dell'agente è massimizzare la sua ricompensa cumulativa

/10
Per chiarezza espositiva i termini tecnici legati all'ambito del RL verranno mantenuti in lingua inglese.

/11
Le observation si differenziano dallo stato in quanto contengono solo le informazioni riguardanti l'environment che sono effettivamente fornite all'agente come feedback. A seconda del tipo di observation si differenziano degli observation space di differente tipologia (generalmente continui o discreti).

/12
A seconda del tipo di environment si differenziano action space discreti e continui.
Le librerie di Stable-baselines offrono quattro tipi di action space utiliizzabili: Discrete (discreto), Box (continuo), MultiDiscrete (discreto), Multibinary (discreto).

lungo tutto l'episodio. Per ricompensa o ritorno cumulativo si intende la sommatoria dei reward ad ogni iterazione scontate per un tasso di sconto γ:

$$R(T) = r_{(t+1)} + \gamma r_{(t+2)} + \gamma^2 r_{(t+3)} + \cdots + \gamma^n r_{(t+n)}$$

Questo consente di descrivere qualsiasi obiettivo, legato a diversi tipi di problema, purché formulabile in termini matematici,, racchiudendolo in un parametro che è indicatore dell'efficienza delle azioni che l'agente sta compiendo.
In questi termini assume significativa importanza la progettazione di un'ottimale funzione di reward, necessaria a far sì che ci sia una corrispondenza tra la performance del modello ed il raggiungimento di una strategia funzionale a risolvere il problema iniziale.

La funzione che determina l'azione da intraprendere dato una stato dell'ambiente è detta *policy* (politica). A seconda del tipo di algoritmo utilizzato, questa funzione definisce una mappatura da ciascuno stato alla migliore azione corrispondente o in alternativa, una distribuzione di probabilità sull'insieme delle azioni che è possibile compiere in quel particolare momento. In termini più intuitivi, la *policy* determina l'atteggiamento dell'*agent* (curioso, prudente, aggressivo, etc.) nella scelta dell'*action* da compiere tra quelle possibili.
Esistono diversi metodi attraverso cui è possibile addestrare l'agent ad apprendere la *policy* più conveniente possibile in termini di ricompensa cumulativa.

La formazione dell'*agent* avviene durante il processo di *training* (addestramento) mediante cui il modello, sperimentando le azioni all'interno dell'ambiente accumula dati riguardanti le relazioni tra ogni stato dell'ambiente e l'effetto che una determinata azione potrebbe avere su di esso. Man mano che l'*agent* compie iterazioni, queste informazioni vengono utilizzate per aggiornarne la *policy* (in step temporali detti *epochs* (epoche)), con lo scopo di aumentare la possibilità di compiere l'azione più conveniente quando il sistema assume un particolare stato.

Durante l'addestramento, l'aspetto che gioca un ruolo fondamentale per il processo di aggiornamento della *policy* è legato al cosiddetto *explotaition/exploration trade off*. Uno dei problemi del RL infatti, è ricercare una *policy* che consenta di massimizzare i *rewards* senza rinunciare a compiere azioni mai sperimentate, esplorando strategie che potrebbero essere ancora più profittevoli per l'*agent* coinvolto. In un certo senso si potrebbe pensare a questa problematica come alla scelta di un ristorante in cui cenare. Se, ad esempio, ipotizzassimo di dover scegliere tra un ristorante conosciuto, in cui ci abbiamo già avuto un'esperienza discreta ed un altro nuovo aperto da poco, che non abbiamo mai provato, la nostra decisione potrebbero avere due esiti: il

primo ci vede rinunciare ad esplorare il nuovo (*exploitation*) basandoci sulla nostra esperienza positiva in quello vecchio. Il secondo caso ci vede esploratori in quanto rischiamo scegliendo il ristorante di cui non abbiamo informazioni esperienziali. Ciò che non vogliamo è che l'esperienza negativa o positiva con il nuovo ristorante abbia un impatto troppo forte sul nostro comportamento, convincendoci a decidere di non esplorare più, se l'esperienza sia stata pessima o a provare sempre posti nuovi, nel caso in cui l'esperienza sia stata gradevole. Analogamente, nel framework del RL è preferibile che la policy del nostro agent tenda a non essere né esageratamente deterministica (conservatrice), né troppo esplorativa.
Questo è il principio su cui si basa il metodo *PPO, Proximal Policy Optimization*, uno dei principali algoritmi usati nel Reinforcement Learning e su cui si basa questa ricerca.

PPO

L'idea alla base di PPO è quella di migliorare la stabilità dell'addestramento dell'*agent* evitando aggiornamenti delle *policy* troppo grandi in ogni epoca di formazione. Questo avviene misurando la variazione tra la *policy* corrente rispetto alla sua precedente. Solitamente i principali algoritmi di RL basati sull'utilizzo di reti neurali (come Soft Actor Critic (SAC), Advantage Actor Critic (A2C), etc..) prevedono che l'aggiornamento della *policy* sia guidato dalla cosiddetta *objective function* (funzione obiettivo) progettata per massimizzare i ritorni attesi dall'agente nel corso del tempo.
In particolare la funzione obiettivo viene utilizzata per calcorare la discrepanza (detta anche *loss*) tra le predizioni della rete neurale e i valori di ritorno che effettivamente vengono prodotti durante il training. Calcolando il gradiente della funzione di perdita la rete neurale modifica i suoi pesi per minimizzare questa discrepanza, aggiornando di conseguenza la *policy*.

$$L^{PG}(\theta) = E_t[\log \pi_\theta(a_t|s_t) * A_t]$$

θ = parametri della nuova politica
$\pi\theta(a_t|s_t)$ = probabilità di prendere l'azione at nello stato st.
A_t = vantaggio stimato al tempo t.

Per riassumere, in questa maniera l'*agent* cercherà di imparare a selezionare le azioni che massimizzino questa funzione consentendogli di migliorare l'efficacia nel suo compito.
L'aspetto problematico in questo caso è che se i valori che la funzione assume dovessero variare in modo significativo da un'epoca di addestramento all'altra, si potrebbe produrre un aggiornamento della *policy* eccessivo o addirittura distruttivo per l'operato dell'*agent*. Per questo motivo in PPO l'aggiornamento della *policy* avviene confrontando la funzione obiettivo con una funzione detta *clipped surrogate objective function*.

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t\left[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)\right]$$

θ = parametri della nuova politica
$\pi\theta(a_t|s_t)$ = probabilità di prendere l'azione at nello stato st.
rt(θ) = ratio function
At = vantaggio stimato al tempo t.
clip(rt(θ), 1−ϵ,1+ϵ) = funzione di clipping
ϵ = iperparametro che determina quanto la nuova politica può deviare dalla vecchia politica.

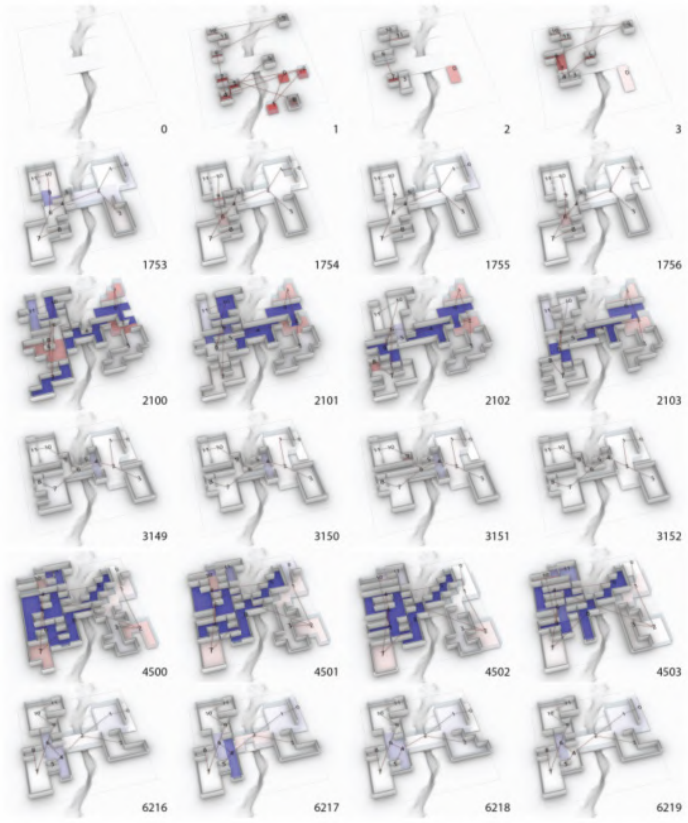
Questa si compone di due termini: la funzione di vantaggio e la funzione di verosimiglianza delle azioni. La prima (non ritagliata) rappresenta quanto una determinata azione è migliore rispetto alle azioni alternative in uno specifico stato. Il vantaggio è calcolato utilizzando una stima dell'importo di ritorno atteso ottenuto dalla *policy* corrente rispetto a una stima basata sulla politica precedente. La seconda (ritagliata) misura quanto le azioni suggerite dalla *policy* attuale sono simili a quelle suggerite dalla *policy* precedente.
Fondamentalmente la scelta del minimo tra la parte ritagliata e quella non ritagliata, funge da paracadute, consentendo di avere aggiornamenti della *policy* non eccessivi, ma selezionando sempre un'opzione che si muova verso una situazione di vantaggio.

2.3 RL in architettura

Dal punto di vista della sua applicazione in architettura, il Reinforcement Learning si pone come un metodo in grado di fornire strumenti avanzati per il supporto di architetti e ingegneri nel processo di progettazione.

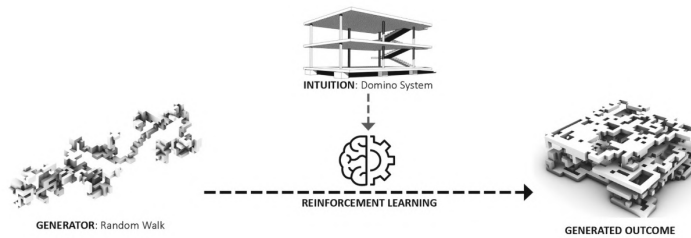
Mediante la simulazione e la loro efficacia, i modelli RL offrono l'opportunità di esplorare metodi innovativi e adattivi, in grado di rispondere a contesti mutevoli e esigenze diversificate, migliorando l'efficienza progettuale. Nonostante questo, spesso i modelli esistenti sono costruiti per problemi non architettonici, difficili da adattare alla progettazione spaziale: nelle sue applicazioni attuali infatti, il RL si è posto perlopiù come strumento per sviluppare strategie di controllo il cui primo obiettivo è quello di incorporare requisiti architettonici al loro interno, affrontando le limitazioni dei modelli esistenti.

Pedro Veloso e Ramesh Krishnamurti nel loro lavoro "*An Academy of Spatial Agents*" (Veloso and Krishnamurti, 2022), hanno affrontato questo tema sviluppando un sistema di agenti RL rappresentanti entità spaziali (come un edificio o una stanza) capaci di controllare localmente la loro forma e la loro interazione con l'ambiente.
Operando su un ambiente condiviso da cui ricevono segnali, essi sono in grado di prendere decisioni per modellare lo spazio seguendo degli obiettivi di vicinato, ossia ricercando l'adiacenza con gli altri agenti e obiettivi di forma ed area che traducono requisiti architettonici in incentivi a creare un determinato tipo di spazio. Questo approccio consente la creazione di configurazioni spaziali che possono essere adattate in tempo reale e rispondere a perturbazioni, supportando l'interazione fine e la co-creazione con il design.

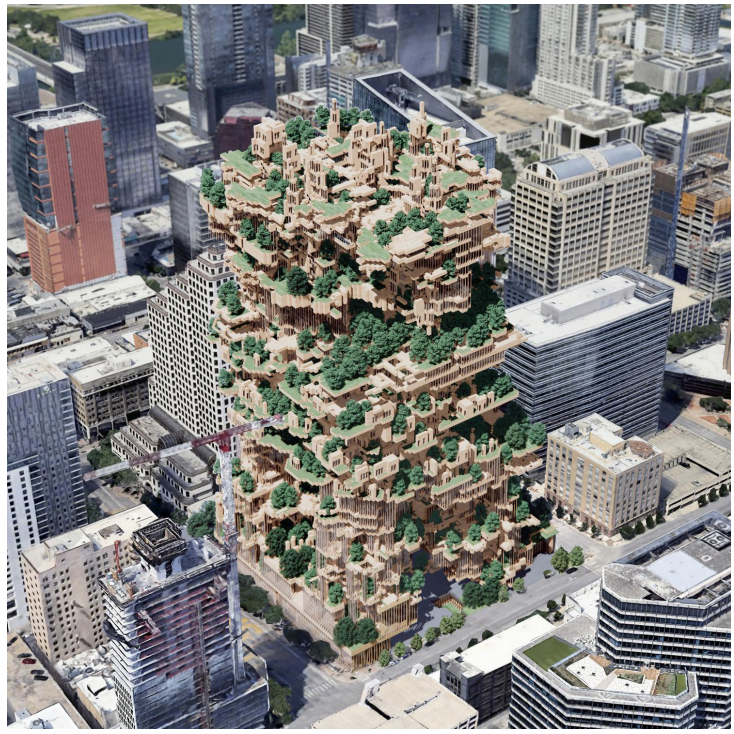


_f23
An academy of spatial agents
- Pedro Veloso

Un'altra funzione del RL può essere quella di integrazione ad algoritmi generativi esistenti per supportare la progettazione combinatoria: lo studio *"Artificial Intuitions of Generative Design: An Approach Based on Reinforcement Learning "*(Wang and Snooks, 2021) mira a combinare l'utilizzo del RL come mezzo di controllo per processi generativi esistenti come il metodo algortimico Random Walk. l'obiettivo di questo esempio è quello di addestrare un RW con una serie di intuizioni architettoniche di base, inizialmente ispirate al sistema Dom-ino di Le Corbusier e ulteriormente sviluppate con intenzioni progettuali riguardanti la logica spaziale e strutturale introducendo criteri di ricompensa che creino strutture simili a tipologie edilizie predeterminate.



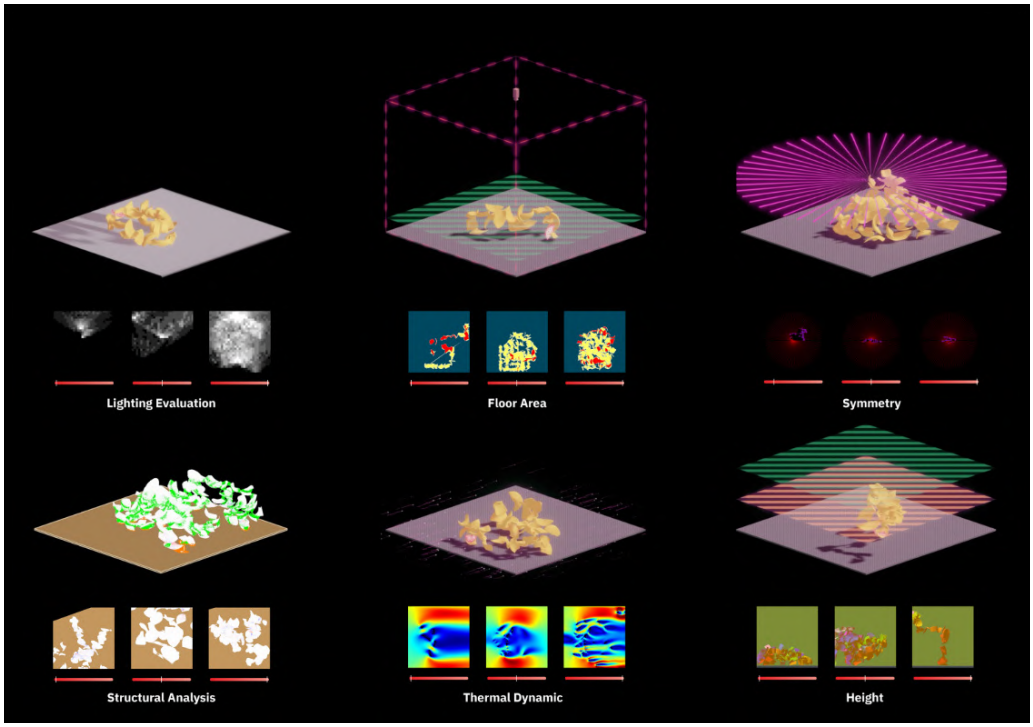
_f24
Artificial Intuitions of Generative Design: An Approach Based on Reinforcement Learning
- Wang



_f25
TheThreeNine building - Daniel Koehler

_f26
Stairs and Slabs - Daniel Koehler

Sempre nell'ambito della progettazione combinatoria, la *One Minute Architecture* (OMA) (Koehler, 2021a), studiata da Daniel Koehler simula il processo di costruzione di edifici planetari guidato da una macchina architettonica basata sull'intelligenza artificiale. Utilizzando i dati prelevati dalle piattaforme immobiliari come sorgente di apprendimento, il modello alla base di OMA è in grado di muoversi nello spazio delle combinazioni possibili, per riorganizzare lo spazio architettonico istante dopo istante. Ciò che lo studio di Koehler mette in mostra è il ruolo che l'IA artificiale può rivestire nello sviluppare macchine capaci di incrementare la velocità dei processi di costruzione combinatori.
L'utilizzo di modelli in grado di automatizzare le procedure di aggregazione e assemblaggio di componenti viene messo in evidenza in ulteriori progetti dello stesso Koehler, come *TheThreeNine building* (Koehler, 2022), un edificio formato da più di 380.000 parti costituenti assemblate a partire da 3 componenti di base, e *Stairs and Slabs* (Koehler, 2021b), che produce configurazioni di possibili edifici formato da due macchine assemblanti che posizionano rispettivamente scale e solai all'interno di una struttura unica.



_f27
Reform Standard - Chien-Hua Huan

Un ulteriore campo di applicazione del Rl è quello dell'automazione del processo di costruzione. Un caso studio in questo senso è il progetto *"Reform Standard"* (Huang, 2021) che si basa sull'utilizzo di un sistema di Machine Learning che per produrre nuove forme a partire da rifiuti. *Reform Standard* si occupa di ordinare e trasformare gruppi di materiale di scarto plastico in nuove forme. L'idea è quella di svincolarsi dai processi di riciclo standard ed energivori, seguendo un approccio material oriented che consenta di arrivare in maniera quanto più possibile vicina al concetto di circolarità "cradle-to-cradle. Partendo da delle scansioni digitali dei rifiuti ottenute con Real Capture, si riesce a creare un inventario ordinato di quelli che sono i materiali di scarto, estraendo da ogni modello scansionato dei dati geometrici e topologici. Dopo questo set up tutti i frammenti e i relativi dati vengono dati in input a Unity, e usati dagli ML agents che operano, ruotando e spostando gli elementi sequenziali dell'inventario per posizionarli. Ad ogni iterazione vengono computate le observation sull'environment analizzando proprietà strutturali dell'assieme, incentivando gli agents a creare assemblaggi stabili e resistenti.

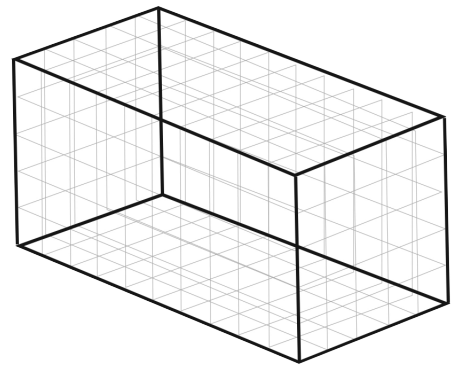
Per concludere, Il Reinforcement Learning si pone anche come mezzo di controllo e gestione della componente energetica dell'architettura. Nel loro studio Hirou Karimi e Mohammad Anvar Adibhesami hanno approfondito modelli per far fronte a scelte riguardanti layout planimetrici e gestione delle sottosistemi come quello dell'illuminazione per ridurre il consumo energetico all'interno di edifici per uffici (Karimi and Anvar Adibhesami, 2024).

03 Ricerca

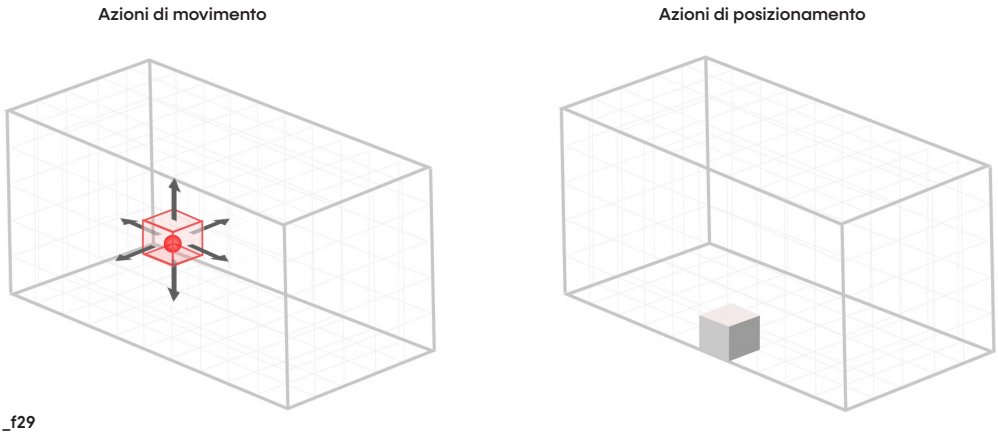
3.1 Apparato di ricerca

Nella fase di ricerca si mira ad addestrare agenti RL capaci di muoversi all'interno di un ambiente dove potranno posizionare e aggregare un numero preimpostato di parti elementari cercando di garantire all'assemblaggio che si sviluppa in questo processo, un valore di ore di illuminazione diretta medio nel range del 10% di un target di progetto, e una adeguata connettività interna (tale da ottenere sistemi spaziali assemblati in non più di 3 gruppi di unità mutuamente isolati tra loro).

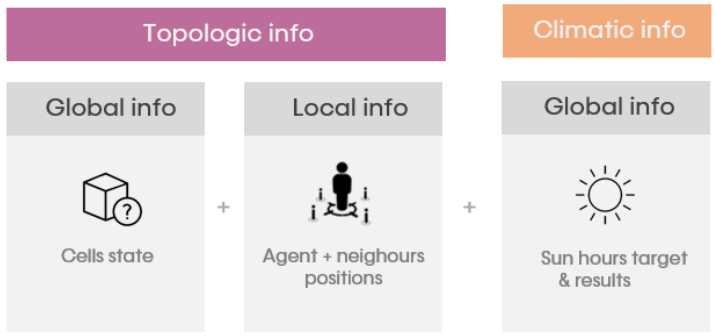
L'ambiente di aggregazione è composto da *voxel* (o celle), volumi elementari cubici, che definiscono matrici tridimensionali dette *voxel space*, spazi volumetrici, di estensione variabile.



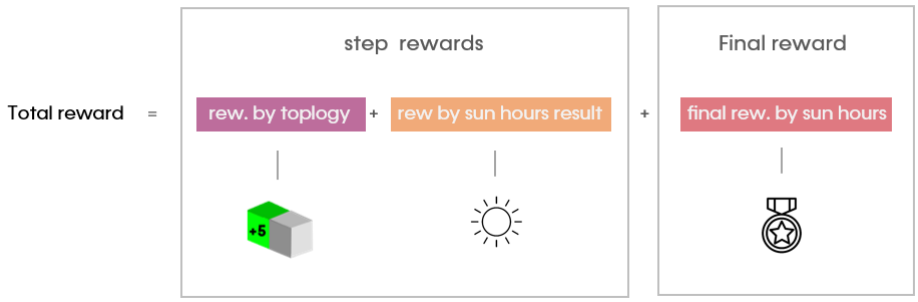
_f28
Diagramma dell'ambiente
di addestramento dell'agente



_f29
Diagramma dell'action space utilizzato.



_f30
Diagramma dell'observation space utilizzato.



_f31
Reward function.

Le azioni che l'agente potrà compiere al suo interno sono di doppia natura: quelle di movimento sono associate al suo spostamento in uno delle 6 possibili celle adiacenti alla sua posizione attuale, mentre quella di posizionamento riguardano la scelta di collocare una delle n componenti nel *voxel* in cui si trova. L'insieme delle 7 azioni che l'agente può compiere definisce il nostro *action space* (**_f29**).

In risposta ad esse l'ambiente fornisce all'agente due tipologie di feedback, *observations e rewards*.

Le *observations* (**_f30**) consistono in un vettore costituito da un numero di dati variabile ad ogni esperimento. In una prima distinzione si dividono in *observation topologiche*, volte a descrivere la topologia dello stato del *voxel space*, e *climatiche,* in quanto contengono target e risultati a riguardo della solar analysis. A loro volta le *observation* possono assumere un carattere *globale,* se funzionali a descrivere lo stato dell'intero ambiente, o *locale*, se ritraggono lo stato della sola cella in cui si colloca l'agente e delle sue vicine ad ogni istante temporale. L'insieme di questi dati compone diversi *observation space*.

I *rewards* (**_f31**) sono le informazioni che l'agente riceve per valutare l'efficacia prodotta dalla sue azioni. In maniera analoga alle *observation* si distinguono in reward climatici, ricompense tanto più alte tanto quanto i risultati della *solar analysis* rientrano nel target specificato, e topologici, variabili nei vari esperimenti, ma che premiano le condizioni di adiacenza tra le nuove parti collocate con quelle già presenti. Un'importante differenziazione è quella tra *step reward*, che vengono forniti ad ogni iterazione, e *final reward*, forniti solo alla fine dell'episodio come feedback dell'operato complessivo dell'agente.

Per quanto concerne la *solar analysis* viene valutato il numero di ore di esposizione a luce diurna diretta. Come risultato l'analisi restituisce un valore compreso tra 0h e 15h per ogni superficie coinvolta. Questo valore indica per quante ore (indipendentemente dalla intensità luminosa) ognuna delle superfici che formano l'assemblaggio riceve luce solare all'interno di una giornata.

Per il nostro scopo divideremo l'intervallo 0-15h in 30 fasce di illuminazione (2 fasce per ogni ora di luce) e condenseremo i dati relativi a tutte le superfici in un fattore medio per il singolo assemblaggio. Questo ci consente di racchiudere i dati climatici in un singolo parametro discreto, facilmente trattabile all'interno del problema di RL. Infatti per svolgere il suo compito l'agente riceverà come informazioni legate al clima la fascia luminosa "target", impostata ad inizio training, e la fascia di illuminazione corrente, calcolata ad ogni iterazione .

Una precisazione va fatta sul periodo in cui viene condotta l'analisi. Nel nostro caso a questo fine è stato impostato il solstizio d'estate a Bologna (alba = 5:30, tramonto = 21:03). Tuttavia questa scelta può essere considerata secondaria o di relativa importanza all'interno del nostro algoritmo: analogamente alla fascia target di illuminazione, la scelta del periodo in cui condurre l'analisi acquisisce significato solo nel momento in cui si specifica un determinato obiettivo che è variabile per ogni specifico caso. In altre parole questa scelta influenza l'istanza ma non il processo, che è stato pensato per essere aperto e adattabile a diverse possibilità in questo senso.

Dal punto di vista strumentale, si è scelto di sviluppare l'intero processo combinando Rhinocheros, Grasshopper (GH) Ladybug, Stable-Baselines3 e Gymnasium. Rhino + Grasshopper sono stati scelti per la loro versatilità nel visualizzare e manipolare modelli architettonici oltre che per la loro compatibilità con gli strumenti di analisi climatica di LadyBug. Le librerie Stable-Baselines3 e Gymnasium consentono di implementare in maniere agevole e con un'ottima documentazione algortimi RL.
Quello che segue è un elenco complessivo degli strumenti utilizzati in questa ricerca:

- Rhino è un software di modellazione 3D usato per creare, modificare, analizzare, documentare, renderizzare curve, superfici e solid NURBS

- Grasshopper è un editor di algoritmi grafici integrato in Rhino che consente agli utenti di creare script visivi per la generazione e manipolazione di geometrie complesse attraverso un'interfaccia basata su nodi.

- Stable Baselines3 è una libreria di Python che offre implementazioni stabili e ben documentate di vari algoritmi di RL. È costruita su PyTorch e offre strumenti per addestrare, testare e confrontare agenti di RL.

- Gymnasium è una libreria di Python che fornisce un toolkit per sviluppare e confrontare algoritmi di apprendimento per rinforzo. Offre una vasta gamma di ambienti di simulazione standardizzati, dove gli agenti di RL possono essere addestrati e testati. Facilita la creazione di nuovi ambienti e supporta l'integrazione con altre librerie di RL.

- Ladybug Tools è una suite di plug-in open-source per Grasshopper che facilita l'analisi ambientale e climatica per l'architettura e il design urbano. Fornisce diversi strumenti per l'analisi e la visualizzazione dei dati climatici.

- Hoopsanke è un plug-in che viene utilizzato per compiere cicli o loop all'interno di Grasshopper. Di fatti Grasshopper è stato sviluppato in assenza di controllo di flusso, il che implica che per compiere operazioni cicliche o ripetute è necessario fare affidamento a plug-in esterni, come Hoopsnake.

- TensorBoard è uno strumento di visualizzazione sviluppato da TensorFlow che consente di monitorare e analizzare gli esperimenti di machine learning.

3.2 Strumenti e pipeline

Il quadro del problema di RL in questa ricerca è stato implementato dal punto di vista strumentale nel seguente modo:

- L'*environment* è generato in Grasshopper ed è visualizzabile direttamente all'interno del software Rhino; è costituito da un *voxel space* formato da celle di dimensione e numero variabile all'interno di cui possono essere collocati dei blocchi cubici.

- L'agent è rappresentato dal modello addestrato mediante l'uso delle librerie Python di Gymnasium e Stable-Baselines3. Le sue azioni vengono computate in Python ed eseguite nell'environment in GH.

- Le *observations* e i *rewards* vengono computati ad ogni iterazione all'interno di Grasshopper e comunicate a Python.

In particolare, questo è stato reso possibile dall'utilizzo combinato (tramite comunicazione socket[13]) degli algoritmi di Reinforcement Learning, implementati in Python, e di Grasshopper (GH).

/13
Una comunicazione socket è un metodo di comunicazione bidirezionale tra due processi (applicazioni o componenti software). In questo modello, i processi comunicano scambiandosi messaggi attraverso un'interfaccia di rete, nota come "socket", formato da una combinazione di un indirizzo IP e un numero di porta.

L'utilizzo di questa pipeline offre due vantaggi in particolare:

1 La possibilità di definire, gestire e visualizzare l'environment di un sistema RL in tempo reale all'interno di uno dei software più utilizzati per la processazione di dati in architettura.

2 Per il fine proposto Grasshopper, differentemente a Unity e altri software comunemente utilizzati per costruire modelli di RL, consente di eseguire analisi di tipo solare ed energetico in maniera pratica e quanto mai efficace grazie agli strumenti di LadyBug Tools.

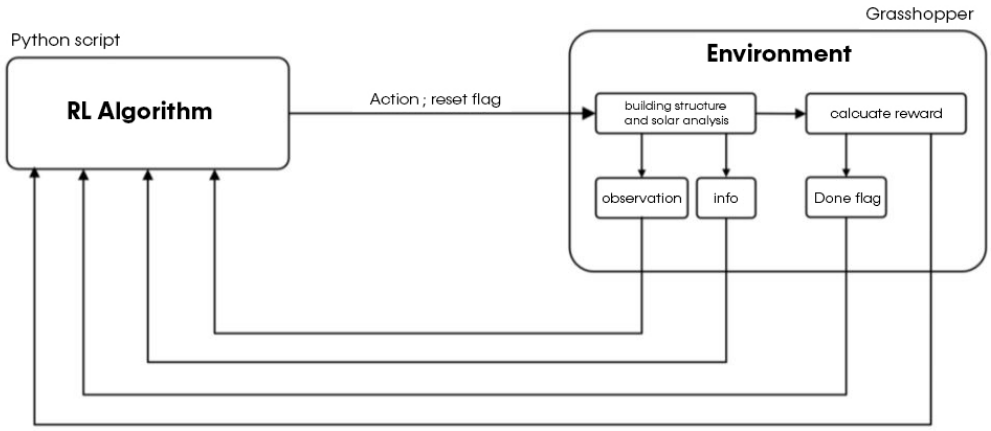
Per il nostro scopo si è deciso di implementare la pipeline sviluppata in *Reinforcement Learning for Architectural Combinatorial Optimization* (Wietschorke, Liu and Chen, 2021).

Essa si basa sull'utilizzo combinato del plug-in Hoopsnake[14] che consente di poter eseguire cicli iterativi all'interno di Grasshopper e di una comunicazione socket eseguita tra Grasshopper e Python, necessaria per poter scambiare i dati computati dalle librerie di RL di Stable-bBaselines3 e Gymnasium da e verso Grasshopper.

In particolare un'iterazione del ciclo di RL avviene come segue:

Nella prima iterazione viene aperta la comunicazione socket tra Python e GH. In Grasshopper vengono computate le observation, ossia i feedback sullo stato dell'environment (modellato in GH), e i reward, che vengono racchiusi in un messaggio e inviati all'algoritmo Python che gli elabora a seguito di ogni iterazione per computare l'action successiva, a sua volta mandata a GH, dove una volta eseguita, produrrà delle nuove observation e dei nuovi reward.

/14
Hoopsanke è un plug-in che viene utilizzato per compiere cicli o loop all'interno di Grasshopper. Di fatti Grasshopper è stato sviluppato in assenza di controllo di flusso, il che implica che per compiere operazioni cicliche o ripetute è necessario fare affidamento a plug-in esterni, come Hoopsnake.



_f32
Diagramma di flusso della comunicazione tra Python e Grasshopper

3.3 Block game 2D

Block game 2D - RL framework

Per testare il corretto funzionamento della pipeline Gh-python, il primo passo è stato costituire una versione semplificata del nostro problema di RL. Questo primo test è stato determinante per le seguenti motivazioni:

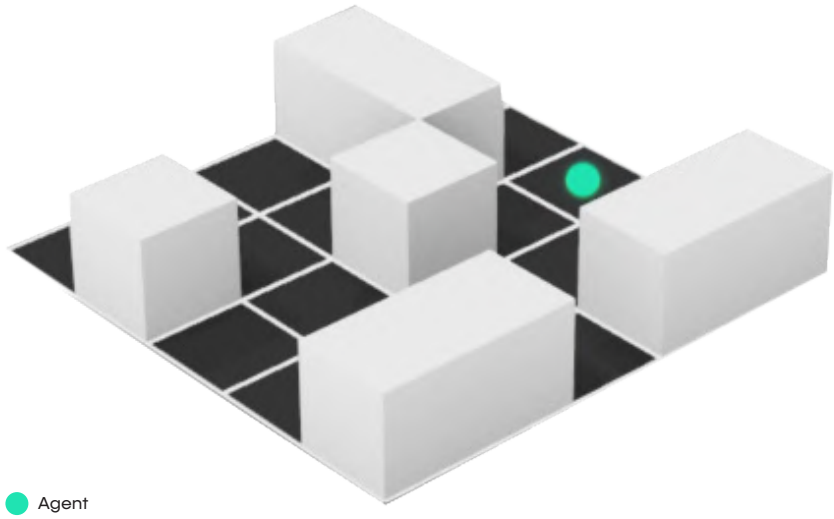
- Verificare il corretto funzionamento della connessione socket tra Grasshopper e Python.

- Testare diversi tipi di algoritmi di RL al fine di inquadrare quello più compatibile alle caratteristiche del nostro problema.

- Individuare gli iperparametri cruciali per l'apprendimento del modello. Gli iperparametri sono le variabili modificabili dall'utente (o da un modello di ottimizzazione degli iperparametri) durante le successive esecuzioni di training di un modello. La loro modifica influisce sulle prestazioni legate a caratteristiche differenti del processo di training

- Costruire un'ambiente di base estendibile a quello finale, più avanzato.

L'ambiente sviluppato è rappresentato da una griglia bidimensionale (di dimensioni 5x5) all'interno di cui un agente potrà muoversi e posizionare blocchi di forma cubica. Il suo obiettivo sarà quello di riempire totalmente la griglia evitando al tempo stesso di inserire un nuovo blocco sulle celle già precedentemente occupate. Ogni qualvolta un blocco sarà aggiunto su una cella, il suo stato passerà da vuoto a pieno. L'episodio si considererà concluso nel momento in cui l'agente porterà a termine il suo obiettivo (*win condition*) o se si verificherà una sovrapposizione all'interno di una delle celle (*game over condition*).



_f33
Block game 2D - Environment

Action space, Observation space, Reward Function

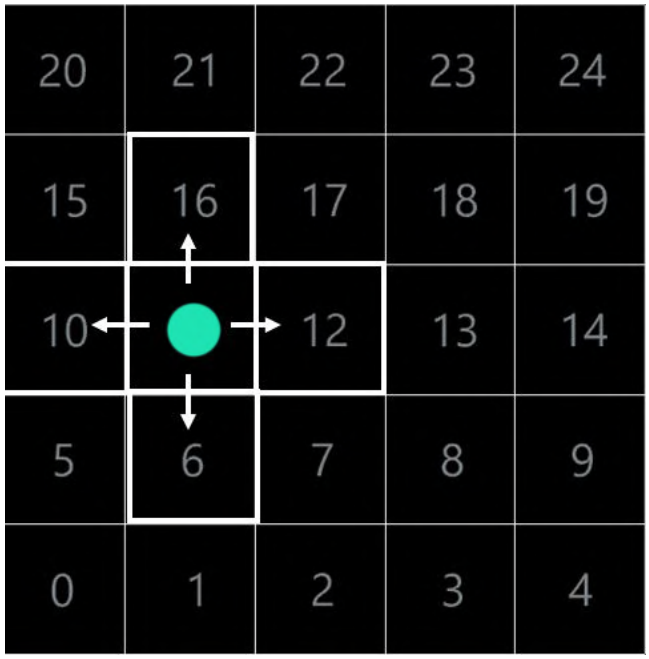
In questo caso è stato definito un *action space*[15] discreto tale da permettere all'agente di compiere i quattro spostamenti cardinali all'interno dello spazio 2D (destra, sinistra, su, giù), posizionare un blocco oppure decidere di rimanere fermo nella sua posizione corrente.

Come observation space è stato definito un vettore di dimensioni uguali a quella della griglia, in cui ogni valore rappresenta lo stato della cella: è stato definito come una matrice di dimensioni uguali a quella della griglia, in cui ogni valore rappresenta lo stato della cella.

Per addestrare il modello è stata progettata una funzione di ricompensa capace di premiare il posizionamento dei blocchi e punirne la sovrapposizione. Le ricompense sono state assegnate in base alle azioni come segue:

/15
La libreria di RL Gymnasium mette a disposizione diverse tipologie di action space. Nel nostro caso è stato usato un action space Discrete (6) in cui ogni azione corrisponde ad un numero intero tra 0 e 5.

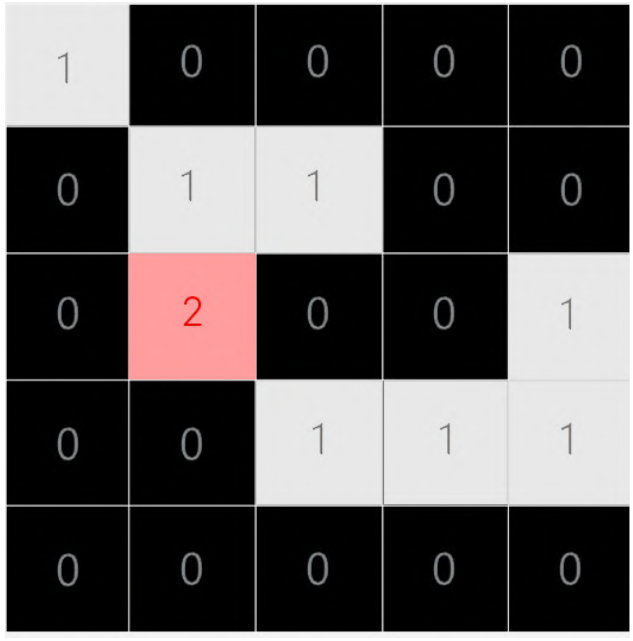
Reward Function
Add box = +1
Other action = -2
Box overlap (game over) = -10
Goal = +1000



_f34
Block game 2D - Action space

Actions
0 = stop
1 = right
2 = left
3 = up
4 = down
5 = add block

Action space
self.action_space = DIscrete (6)



_f35
Block game 2D - Observation space

Observations
Cell state:
0 = empty
1 = not empty
2 = overlapping

Observation space
self.observation_space = Multi-DIscrete [(3)*num_cells]

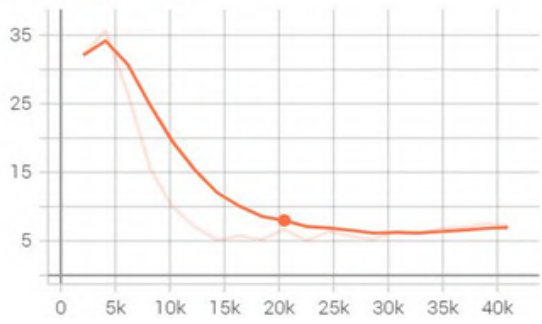
Training e risultati

Una volta definito il modello, è stata condotto un primo training della durata di 40.000 iterazioni. Per le caratteristiche del nostro environment è stato scelto l'algoritmo PPO, compatibile sia con la tipologia dell'*observation space* che dell'*action space utilizzati.*
I risultati del training sono racchiusi all'interno dei seguenti grafici che mostrano l'andamento sulle metriche riguardanti il comportamento dell'agente nel tempo.
I grafici sono ottenuti attraverso l'utilizzo di TensorBoard[16].

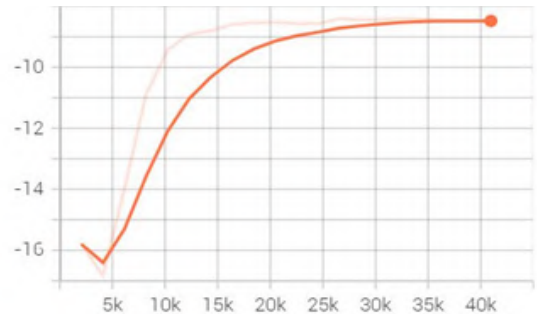
Il grafico *ep_len_mean* si riferisce alla durata media dell'episodio al procedere delle epoche di addestramento. Una curva discendente indica che la durata dell'episodio cala progressivamente nel corso del training. Ciò può essere interpretato positivamente laddove l'*agent* riesca a portare a termine l'episodio con un numero di azioni ridotto (diventa

/16
TensorBoard è una libreria open-source per il ML sviluppata da Google. È uno strumento utile alla visualizzazione e al monitoraggio dei modelli di ML.

ep_len_mean
tag:rollout/ep_len mean



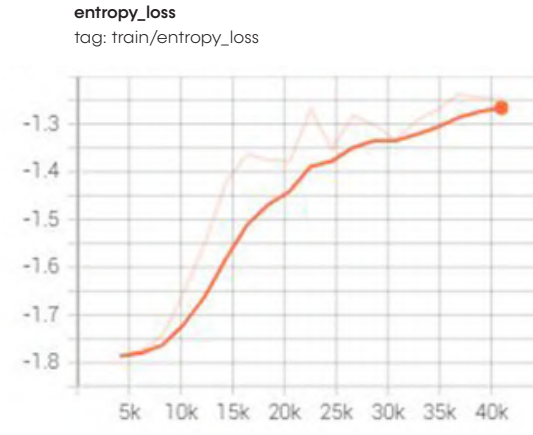
ep_rew_mean
tag:rollout/ep_rew_mean



sempre più efficiente) o negativamente se invece se questo trend indica un troncamento dell'episodio al raggiungimento del *reward* massimo senza che i task siano stati effettivamente risolti.

Il grafico *ep_rew_mean* indica invece l'andamento del *reward* medio tra gli episodi all'interno di ogni epoca di addestramento. Una curva crescente indica prestazioni positive in quanto il reward cresce durante l'addestramento ma non è condizione sufficiente a determinare il successo dell'*agent*. Fondamentale per questo è il raggiungimento di valori di reward medi rispetto al range di reward possibile in un episodio.
Determinante per il giudizio di un training è dunque la visione combinata di questi due grafici in quanto offrono informazioni parziali e non capaci di stabilire singolarmente il grado di successo o di insuccesso di un training.

In questo caso si nota come il modello durante l'addestramento abbia aumentato la ricompensa media riducendo il numero di azioni intraprese. Una policy di questo genere ha privilegiato ricompense a breve termine piuttosto che a lungo termine. In un certo senso la policy del modello si è aggiornata in modo tale da portarlo ad intraprendere azioni che tendessero a interrompere l'episodio il prima possibile, piuttosto che continuare ad accumulare reward negativi.
Ciò è visibile anche nella voce *entropy loss*. L'*entropy loss* è un iperparametro che indica quanto il modello sta esplorando o quanto le sue logiche di scelta siano deterministiche. Nel nostro caso l'aumento di *entropy loss* avviene in campo negativo, il che indica un'iniziale fase di esplorazione del modello che progressivamente si è appiattita sulla ripetizione deterministica di alcune azioni.
Questo fenomeno è dovuto probabilmente ad una incorretta proget-



tazione della reward function: avendo attribuito un'eccessiva punizione alla condizione di overlap ed una discreta all'esecuzione degli spostamenti cardinali, il modello è stato incentivato a compiere meno iterazioni possibili, interrompendo l'episodio subito dopo ottenuto qualche reward positivo.
Nonostante questo primo esperimento abbia avuto un esito negativo, è stato ritenuto sufficiente per validare la pipeline utilizzata e per evidenziare l'importanza di una corretta gestione della funzione di reward e delle informazioni contenute all'interno dell'observation space, necessarie per garantire un processo di formazione più veloce ed efficace.

3.4 Sun hours based generative system

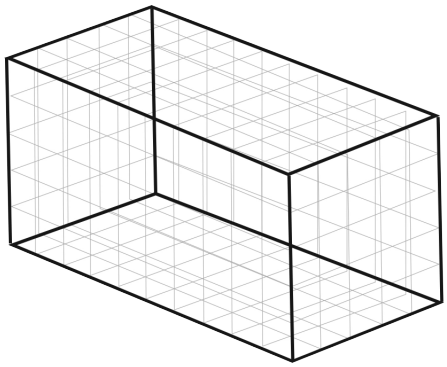
Una volta testata l'efficacia del workflow in Grasshopper, lo step successivo è quello di addestrare un agente a portare a termine un compito analogo a quello che svolgeva l'algoritmo alla base di *Sun God City* (Watanabe, 1995).
In quel caso un algoritmo iterativo, settato su condizioni di illuminazione solare predefinite, era in grado di generare una molteplicità di configurazioni spaziali, in target con l'obiettivo climatico previsto. Nel nostro caso le decisioni sul posizionamento delle unità che definiscono i diversi assemblaggi saranno affidate ad un *agent*.

Il modello è stato allenato progressivamente in diverse versioni, progettate per risolvere strada facendo le problematiche riguardanti in primis la corretta gestione della funzione di reward e in secondo luogo quelle inerenti il tipo di *observation* assegnate all'*agent* come feedback per il suo operato.
È importante specificare come al fine valutativo dei modelli sviluppati nella ricerca sia stato definito un set di criteri generali, riguardanti la natura architettonica dei sistemi spaziali creati nel processo, essenziale per tracciare in maniera più precisa possibile il grado di successo o insuccesso dei vari test eseguiti, delineando il percorso di ricerca.
I criteri riguardano gli ambiti di clima, connettività e scalabilità.
Il primo è stato introdotto come rispondenza di ogni assemblaggio prodotto a condizioni di illuminazione naturale di progetto (target) all'interno di un margine del 10%.
Il secondo riguarda i concetti di percorribilità e circolazione che sono stati analizzati sinteticamente come numero di *cluster* (gruppi separati di blocchi aggregati) di circolazione. Nel tentativo di ottenere assemblaggi totalmente connessi, le considerazioni su cui si sono basate le scelte relative al quadro di RL (*observation, reward*) si sono basate sul cercare di ridurre questo valore all'interno dei test effettuati nei vari modelli.
Il terzo criterio è inteso come la possibilità di applicare ed estendere il sistema a dimensioni potenzialmente infinite. Operativamente, questo implica la mediazione tra aspetti concettuali e tecnici per arrivare ad un metodo adattabile ad *environment* di dimensione e natura variabile.

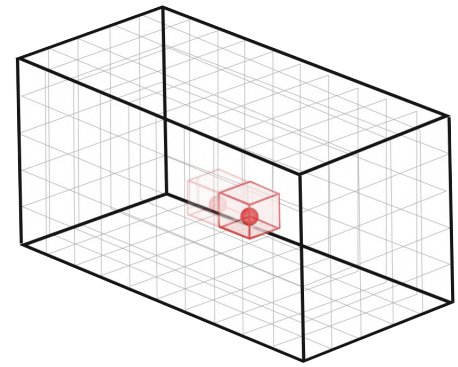
PPO model 0.4

A fronte di tre tentavi di setup, in questa prima versione l'agente si trova all'interno di uno spazio voxelizzato tridimensionale, formato da 250 celle distribuite 10x5x5 (lxpxh). Estendendo l'*action space* del game test 2D, all'interno di esso potrà compiere 7 azioni. Le prime 6 consentono i movimenti tra una cella e le sue vicine all'interno dello spazio, mentre la settima prevede il posizionamento di un'unità.

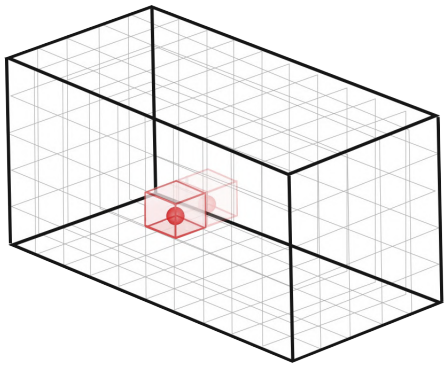
Environment



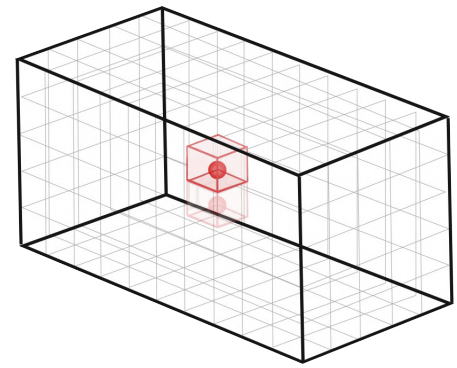
Action 0
right



Action 3
down

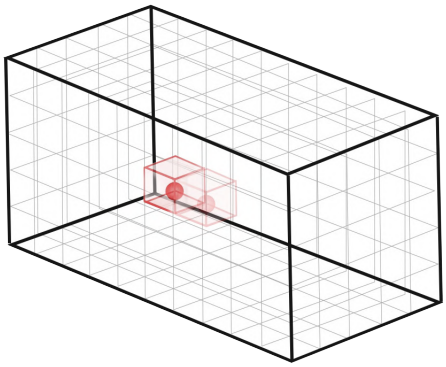


Action 4
up (z)

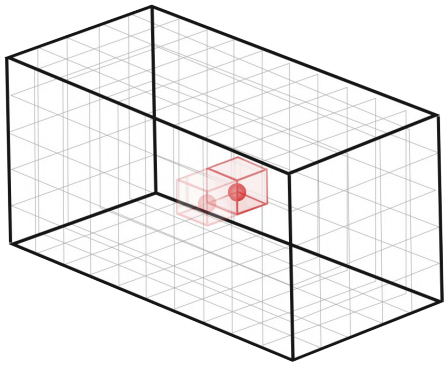


Per terminare un episodio l'agente dovrà disporre un numero n di blocchi preimpostato, facendo in modo che i risultati della *Direct sun hours analysis* (calcolati come media dei valori analizzati sulle singole superfici di ogni blocco - **_f36**) tendano ad un valore target preimpostato di progetto (in questo caso di 7h).

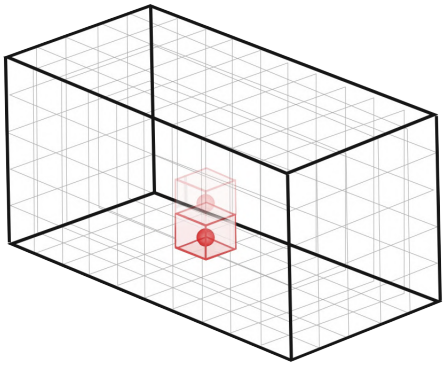
Action 1
left



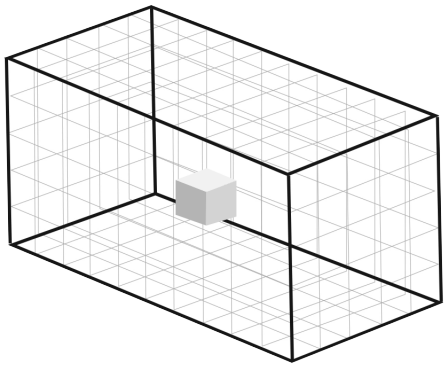
Action 2
up



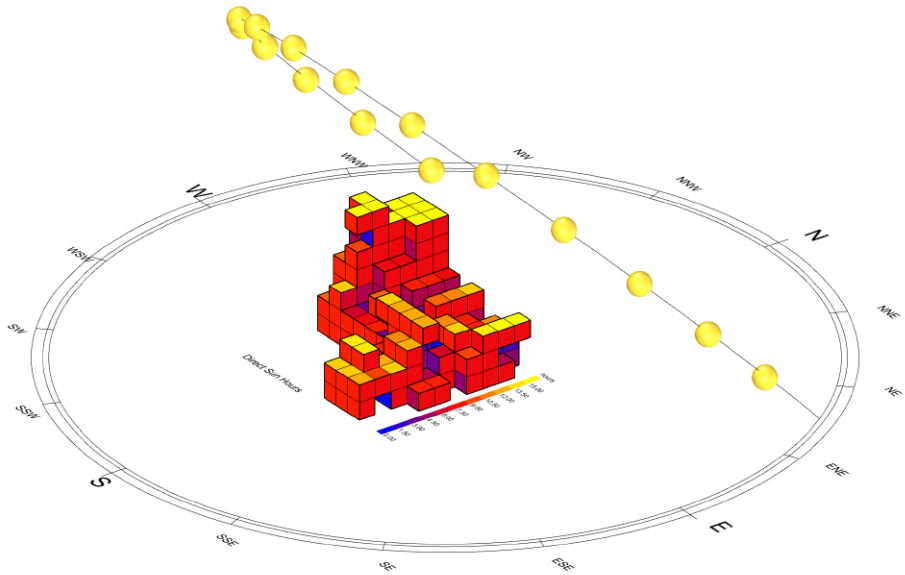
Action 5
down (z)



Action 6
add block

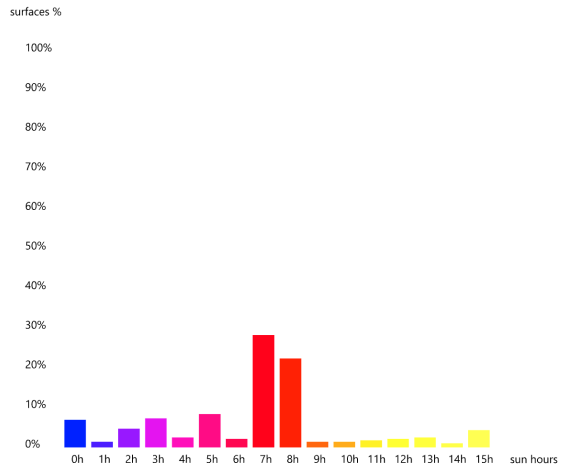


Direct sun hours analysis



_f36
Direct sun hours analysis Grasshopper

Data distribution



step reward

```
Gap = (target - results)

Gap < 1        ⟶  + 10
1 < Gap < 2    ⟶  + 5
2 < Gap < 4    ⟶  - 5
Gap > 4        ⟶  - 10
```

final reward

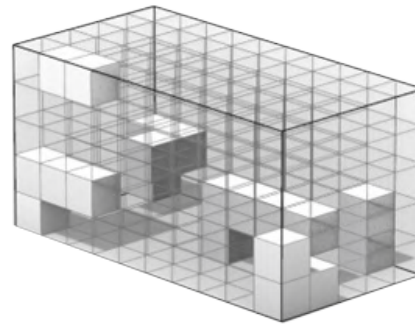
```
results = target ± 1    ⟶  + 1000
results ≠ target ± 1    ⟶  - 1000
```

Per farlo, il modello come feedback per le azioni compiute nell'*environment* riceve esclusivamente il risultato dell'analisi ad ogni iterazione (in un valore tra 1 e 30 che rispecchia la corrente fascia di illuminazione).

Al fine di evitare di scoraggiare l'*agent* a compiere l'azione di collocamento delle unità, la funzione di reward è stata definita in maniera da attivarsi solo nel momento del posizionamento di un blocco. All'interno di essa vengono premiati posizionamenti che riscontrino una variazione nei risultati dell'analisi solare in direzione del valore target di progetto, e puniti quelli che si discostano eccessivamente da esso. La *win condition* è rappresentata da una ricompensa di +1000 all'ultima iterazione[17].
L'analisi del training di questo modello ha mostrato molte imperfezioni nella progettazione sia dello spazio delle osservazioni che nella funzione di reward utilizzata.

Infatti gli assemblaggi finali (**_f37**) mostrano evidenti problematiche di concezione spaziale e topologica, determinate dalla presenza di elementi sconnessi o staticamente inconcepibili all'interno dell'insieme. Questo fenomeno è conseguenza dell'assenza di un sistema di ricompense che tenga in considerazione gli aspetti topologici/architettonici dell'assemblaggio.

/17
L'utilizzo di una ricompensa finale molto più alta rispetto alle altre rappresenta sia un'informazione inequivocabile sul raggiungimento dell'obiettivo da parte dell'agente, che un dato utile a comprendere in maniera più efficace e veloce l'andamento del training nella fase di analisi delle sue metriche.



_f37
Assemblaggio creato dal modello PPO 0.4

I risultati deludenti in termini di reward e di efficacia del modello, evidenziano inoltre la necessità di fornire all'interno dei feedback dati in output all'*agent* sottoforma di *observation*, il valore target dell'analisi solare. Questo aiuterà il modello a costruire un'associazione tra le ricompense che riceve e lo scostamento tra il risultato dell'analisi e il target impostato.

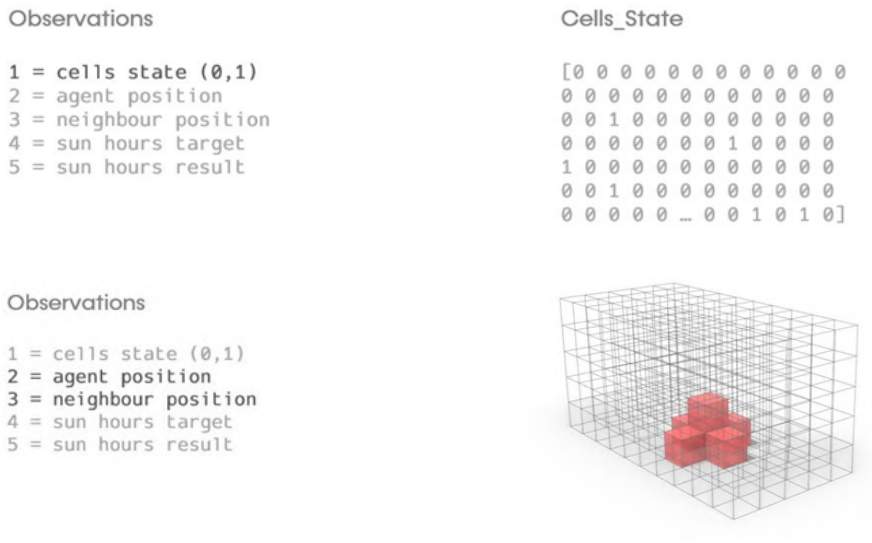
Results & Topologic feedback

Nella ricerca di un modello le cui performance positive legate alle ricompense fossero accompagnate dalla produzione di assemblaggi spaziali al contempo in target con l'obiettivo climatico specificato e architettonicamente plausibili per aspetti topologici legati a densità e struttura, si è passato ad addestrare sei nuovi modelli concentrandosi su tre aspetti fondamentali:

1 Per rendere più efficiente il processo di formazione dell'agent, sono stati aggiunti diversi altri parametri nella lista delle *observations* restituita ad ogni iterazione.

2 Per colmare le lacune evidenziate nei modelli precedenti dal punto di vista dell'organizzazione spaziale degli assemblaggi, sono state progettate diverse funzioni di reward idonee ad assegnare premi e ricompense in base a differenti logiche riguardanti lo stato delle celle adiacenti ad ogni blocco aggiunto.

3 Al fine di incrementare le performance degli agenti, si è passati a processi di addestramento più duraturi arrivando alla soglia delle 100.000 iterazioni.



_f38
Diagramma delle observation aggiunte al modello 0.5

Observation space

Punto di partenza per la ricerca di un modello più performante sotto gli aspetti precitati è stato lo studio di un *observation space* ancor più denso di informazioni utili all'agente per cogliere le relazioni tra le mosse compiute, il nuovo stato del sistema, e le ricompense/punizioni ricevute in risposta ad esse.

Il primo set di informazioni necessarie a descrivere l*'environment,* sono quelle riguardanti lo stato delle celle che lo compongono. Queste informazioni sono state racchiuse all'interno della lista *cells_state,* i cui elementi possono assumere un valore tra 0 e 1 a seconda che al loro interno sia stato o meno inserito un blocco.

Alla lista c*ell_state*, sono state poi aggiunte le informazioni riguardanti gli indici della cella rappresentante la posizione dell'agente e di quelle ad essa adiacenti. Questo consente al modello di comprendere la sua posizione all'interno della lista degli stati di ogni cella, e di conseguenza cogliere quale azione può essere più funzionale in quel determinato stato. Inoltre, avere a disposizione la lista dei proprio vicini, consente di conoscerne lo stato. Questo diventa fondamentale per il processo di posizionamento dei blocchi in ottica di una funzione di reward che premia situazioni di densità all'interno dell'assemblaggio.

L'ultimo set di dati che giocano un ruolo fondamentale nel rappresentare lo stato dell'ambiente riguarda i risultati dell'analisi dello ore di luce diurna: in aggiunta ad essi (sempre calcolati come media di tutte le superfici e riparametrizzati in 30 fasce di illuminazione) viene forni-

ta anche la fascia di illuminazione target. Ciò è funzionale al modello per comprendere in che modo la variazione dello scarto tra target e risultati è correlata ad aumento/decremento delle ricompense.

Tutte queste informazioni vengono condensate all'interno di un observation space *MultiDiscrete*[18] fatto come in figura (**_f39**).

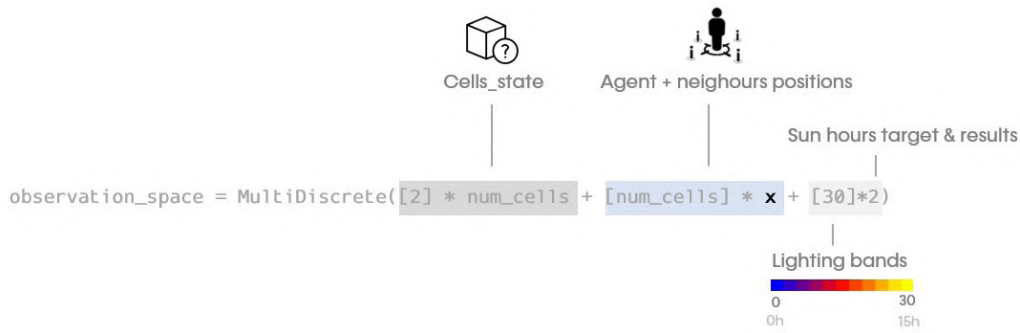
Reward function

Lo studio di una *reward function* adeguata agli obiettivi di questa ricerca è stata una delle problematiche principali da gestire. Definito un *observation space* comune per i vari modelli testati, si è avviato un processo di progressiva raffinazione del sistema di ricompense utilizzato.

La difficoltà maggiore sta nel tradurre un problema aperto come quello architettonico in una funzione matematica. Sebbene dal punto di vista concettuale sia impossibile racchiudere in un numero chiuso qualità incomputabili e soggettive legate all'architettura, è possibile utilizzare approcci matematici per trattare gli aspetti quantitativi dello spazio.

Nei diversi modelli testati la reward function è stata sempre suddivisa in tre differenti attributi: il primo legato agli aspetti di carattere spaziale/topologico, il secondo riguardante puramente i dati climatici, il terzo legato al raggiungimento dell'obiettivo finale (dunque anch'esso legato alla analisi delle ore di luce diurna).
Quella che segue è una presentazione di diverse proposte studiate e dei risultati che hanno prodotto in termini di performance legate agli obiettivi.

/18
Il parametro "x "nei diversi modelli può variare da 6 a 7 a seconda che venga effettivamente fornita o meno la posizione dell'agente come informazione aggiuntiva.



_f39
Observation space - PPO
model 0.5

PPO model 0.5

I risultati ottenuti dalla prima fase di training hanno mostrato come previsto delle carenze spaziali e/o topologiche all'interno degli assemblaggi finali di blocchi.
In questa versione si è cercato di arginare queste problematiche introducendo delle ricompense tali da premiare la densità dell'assemblaggio e da punire la creazione di cluster isolati di blocchi.
La reward function è stata dunque modificata come segue:

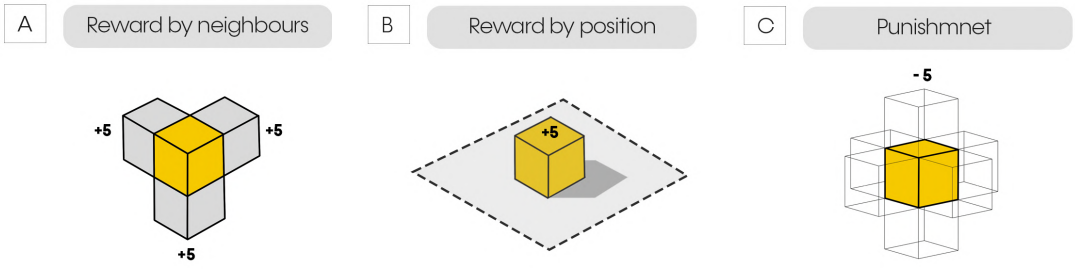
A. Ogniqualvolta un blocco viene posizionato in una cella che ha i suoi vicini occupati, viene corrisposta una ricompensa di +5 per ogni vicino con cui il blocco è adiacente. Essendo 6 il numero massimo di vicini per ogni cella, il range ad ogni iterazione per cui questa componente può oscillare è tra 0 e +30.

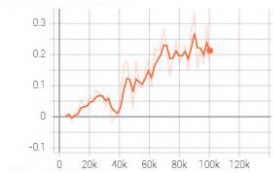
B. Per stimolare la crescita progressiva dal basso verso l'alto dell'assemblaggio viene riconosciuta una ricompensa di +5 nel momento in cui il blocco viene posizionato al livello 0.

C. Viene punito il collocamento di blocchi isolati (privi di vicini) con una punizione di -5.

Le ricompense riguardanti i dati climatici restano invariate rispetto a quelle utilizzate nel PPO model 0.4.

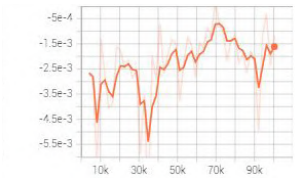
È stato inoltre utilizzato un *observation space* che non tiene conto dello stato delle celle adiacente alla posizione corrente dell'agente. Impostando le condizioni di illuminazione target sulla fascia oraria relativa alle 7 ore di luce e con un numero di 50 blocchi da posizionare





explained_variance
tag: train/explained_variance

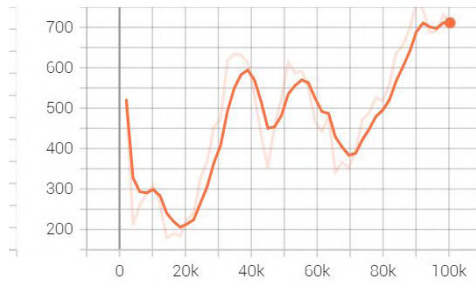
è stato avviato una sessione di addestramento da 100.000 iterazioni. Nel complesso, i grafici indicano che si è verificato un processo di apprendimento, come evidenziato dalle tendenze nella *explained_variance* e nella ricompensa media. Tuttavia, alcune incongruenze nei parametri, come la perdita del gradiente politico e la perdita di valore, suggeriscono che il processo di apprendimento potrebbe essere piuttosto volatile.
L'instabilità del modello è ancor più evidente all'interno degli assemblaggi testati a partire da esso:



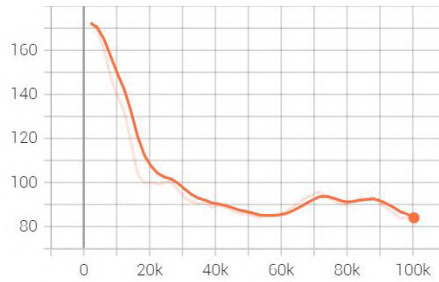
policy_gradient_loss
tag: train/policy_grad_loss

Il grafico model score è uno *scatterplot* i cui punti sono associati ai diversi assemblaggi ottenuti da ogni relativo modello. In particolare le ordinate indicano il reward totale ottenuto dall'agent nel singolo test (*Total reward*) e le ascisse si riferiscono alla quota parte di ricompense accumulate in relazione ai soli requisiti climatici (*reward by sunhours*). In questo senso man mano che ci si sposta dal quadrante in basso a sinistra verso quello in alto a destra troviamo esperimenti sempre più compatibili con gli obiettivi topologici e climatici specificati.

ep_rew_mean
tag:rollout/ep_rew_mean

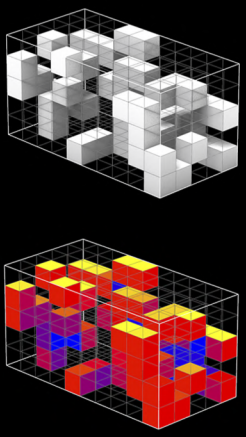


ep_len_mean
tag:rollout/ep_len mean



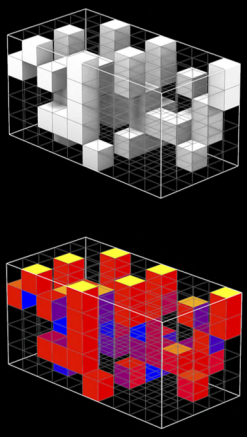
PPO model 0.5

test A



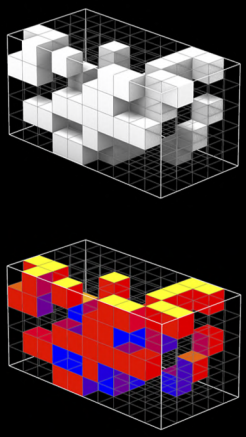
Total reward **- 1085**
Rew by sunh. **-870**

test B



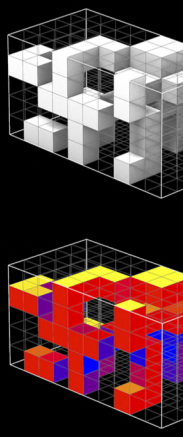
Total reward **-1010**
Rew by sunh. **-820**

test C



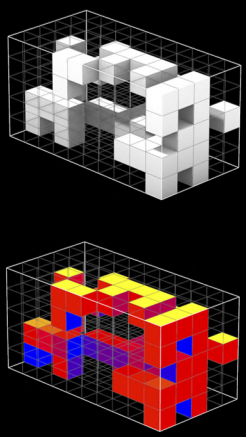
Total reward **-1250**
Rew by sunh. **-920**

test D



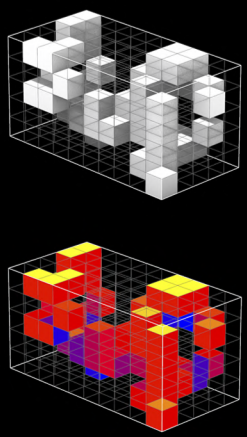
Total reward **-1005**
Rew by sunh. **-830**

test F



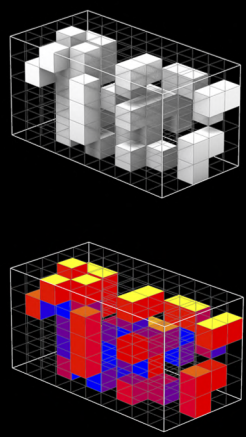
Total reward **-1080**
Rew by sunh. **-825**

test G



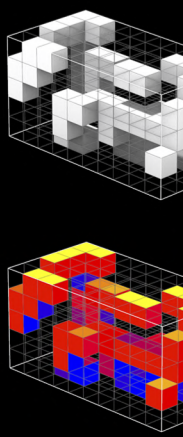
Total reward **-1045**
Rew by sunh. **-820**

test H



Total reward **-1035**
Rew by sunh. **.845**

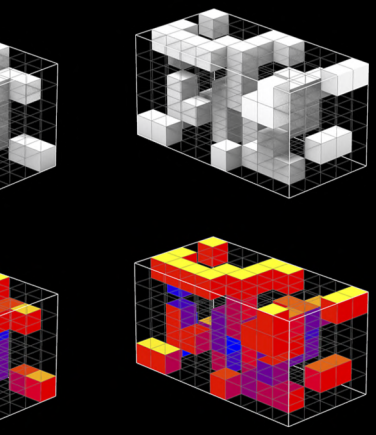
test I



Total reward **-1060**
Rew by sunh. **-820**

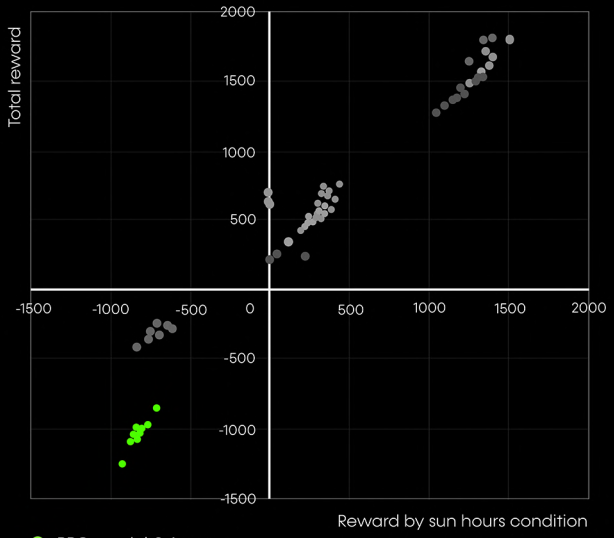
0h 15h

test E



Total reward **-970**
Rew by sunh. **-765**

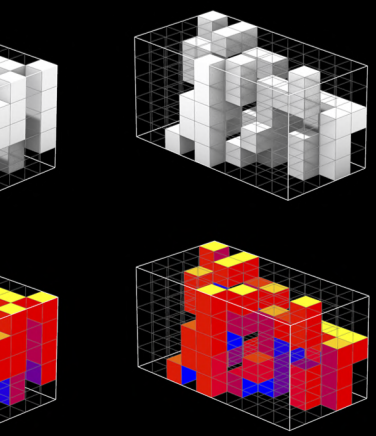
model score



PPO model 0.6 tests

Training timestep = 100.000
Sun hours target = 7h
Assemblage n of part = 50

test J



Total reward **-850**
Rew by sunh. **-710**

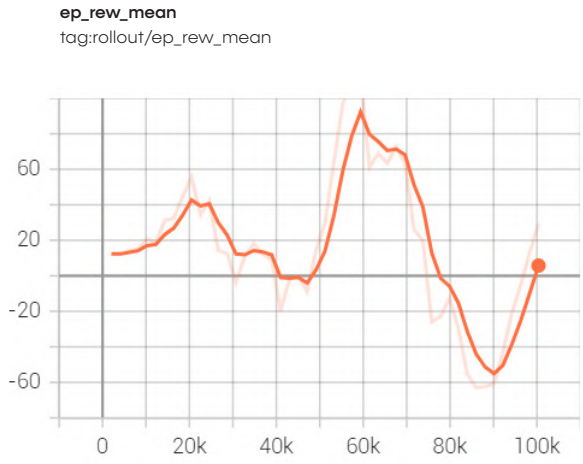
Come evidente, in questo caso, l'agent non riesce a portare a termine né l'obiettivo legato alle sun hours, né quello legato alla densità dell'assemblaggio. I sistemi spaziali sono spesso formati da cluster sconnessi, talvolta accompagnati dall'eccessiva presenza di elementi flottanti. Le motivazioni che spiegano in parte questo comportamento sono da ricercare in due temi:

- La reward function risulta essere eccessivamente complessa. La gestione di un sistema di reward frazionato in tre contributi introduce incertezza sulla relazione tra azione, stato e ricompensa.

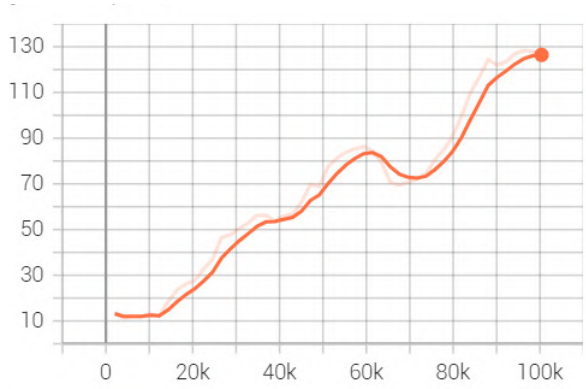
- L'observation space è ancora troppo poco informativo per l'agente.

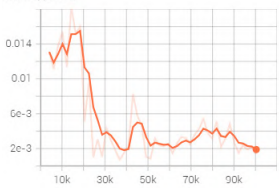
PPO model 0.6

Tenendo in considerazione i risultati del training precedente, nella versione 0.6 è stato introdotto un singolo cambiamento: all'interno dell'observation space sono state inserite le informazioni relative allo stato delle celle adiacenti alla posizione dell'agent. L'idea è quella di concentrarsi su uno dei due problemi precitati per volta al fine di localizzare la fonte della volatilità del training.
Sia la reward function che le impostazioni di training sono rimaste inalterate producendo i seguenti risultati:
Anche in questo caso le metriche mostrano una certa instabilità nell'apprendimento: La media delle ricompense per episodio mostra una volatilità significativa, con un picco negativo verso il centro del training. Ciò potrebbe suggerire che l'agente sta esplorando nuove strategie che non sempre portano a risultati positivi.

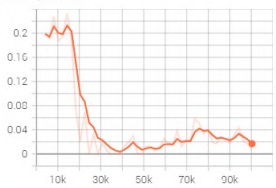


ep_len_mean
tag:rollout/ep_len mean

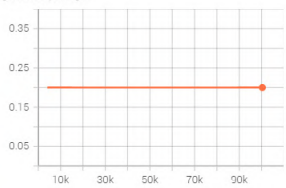




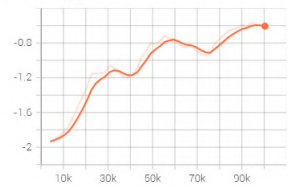
approx_Kl
tag: train/approx_kl



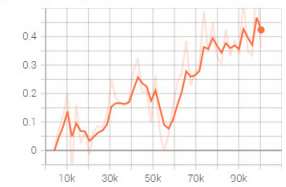
clip_fraction
tag: train/clip_fraction



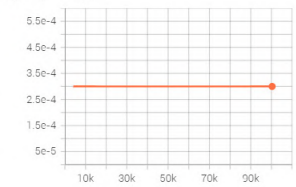
clip_range
tag: train/clip_range



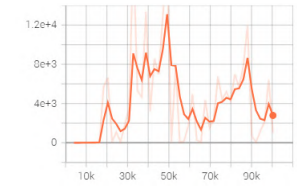
entropy_loss
tag: train/entropy_loss



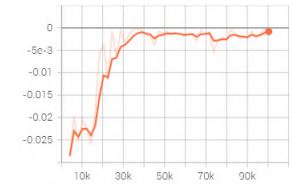
explained_variance
tag: train/explained_variance



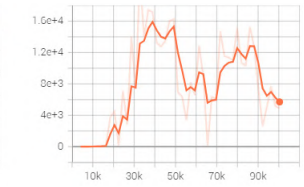
learning_rate
tag: train/learning_rate



loss
tag: train/loss



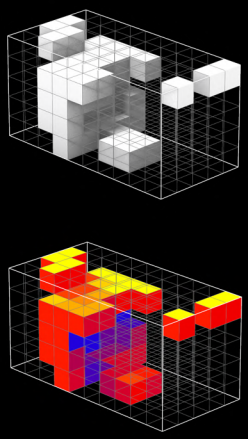
policy_gradient_loss
tag: train/policy_grad_loss



value loss
tag: train/value_loss

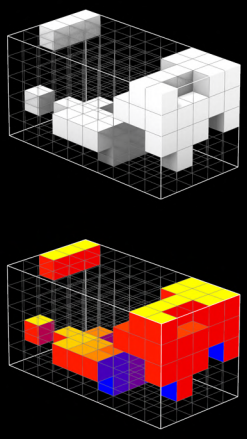
PPO model 0.6

test A



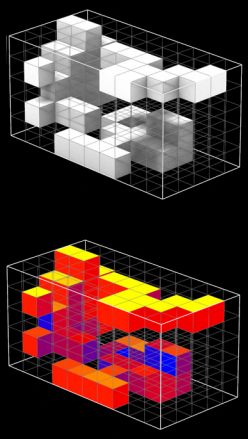
Total reward - **-755**
Rew by sunh. **-355**

test B



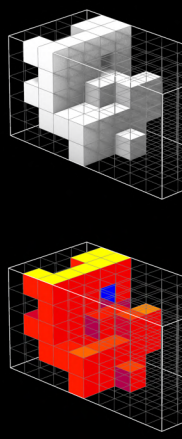
Total reward **-640**
Rew by sunh. **-265**

test C



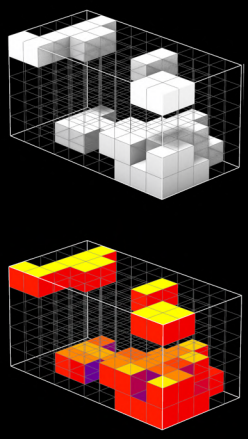
Total reward **-690**
Rew by sunh. **-320**

test D



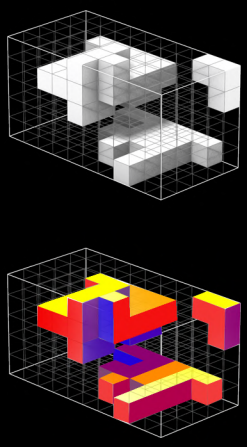
Total reward **-745**
Rew by sunh. **-305**

test F



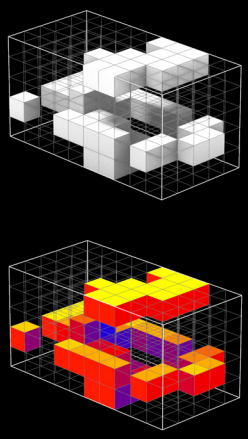
Total reward **1800**
Rew by sunh. **1330**

test G



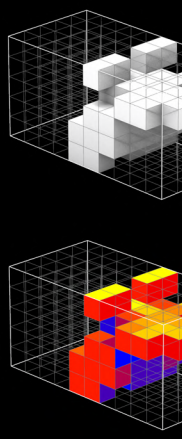
Total reward **1805**
Rew by sunh. **1375**

test H



Total reward **-615**
Rew by sunh. **-270**

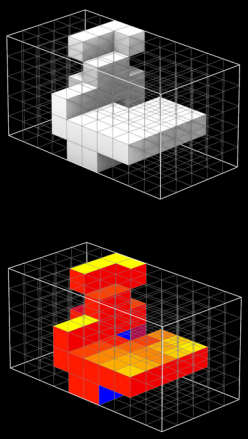
test I



Total reward **-830**
Rew by sunh. **-415**

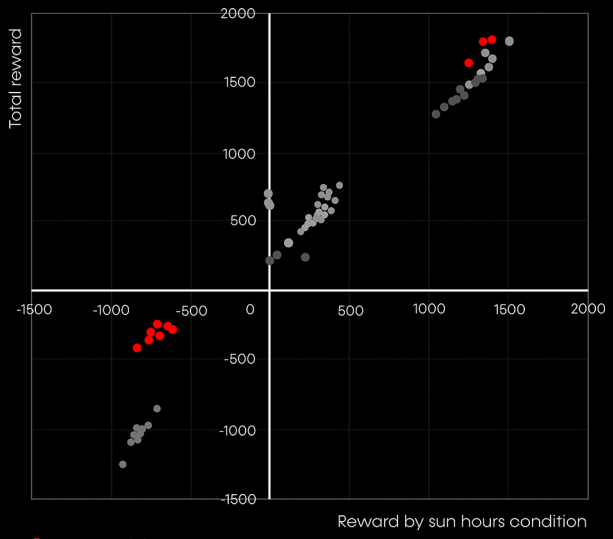
0h 15h

test E



Total reward **-710**
Rew by sunh. **-245**

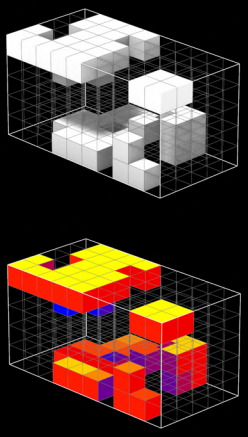
model score



PPO model 0.6 tests

Training timestep = 100.000
Sun hours target = 7h
Assemblage n of part = 50

test J



Total reward **1640**
Rew by sunh. **1250**

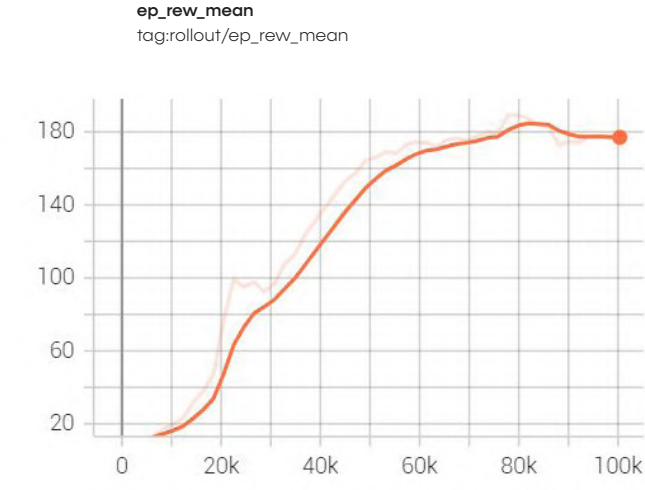
Evidente è come all'interno dei test condotti utilizzando questo modello ci sia una grande variabilità con un tasso di successo/insuccesso per episodio vicino al 50%.

PPO model 0.7

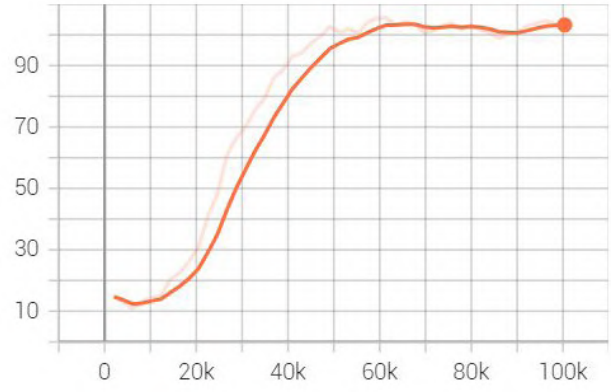
Come previsto, la mancanza di un adeguato *observation space* non è l'unica fonte dell'instabilità dei training. Di conseguenza ci si è concentrati sul secondo problema relativo al modello 0.5, ossia l'eccessiva complessità della funzione di *reward*.

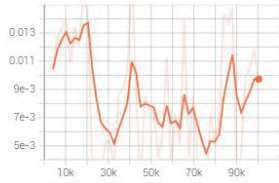
A tal fine la logica che si è voluta perseguire è stata quella di semplificare il sistema di ricompense, tenendo invariati i premi, ma rimuovendo le punizioni. Essendo infatti il modello per sua natura programmato a massimizzare la ricompensa cumulativa, potrebbe non avere bisogno di sanzioni per ottimizzare il suo comportamento, anzi, talvolta esse potrebbero risultare fuorvianti ai fini della ricerca della migliore azione da compiere in un determinato stato.

Seppure i dati relativi al *reward* medio per episodio non mostrano progressi significativi, l'evoluzione del training mette in luce come la rimozione di contributi negativi all'interno del sistema di ricompense abbia aumentato la stabilità del training.

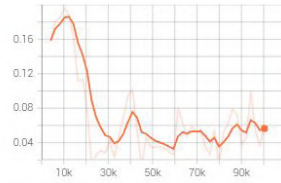


ep_len_mean
tag:rollout/ep_len mean

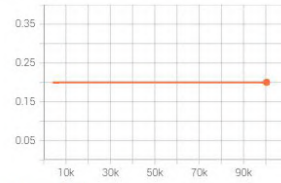




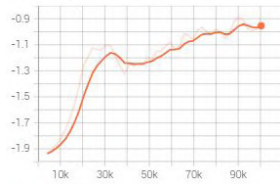
approx_Kl
tag: train/approx_kl



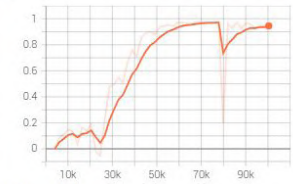
clip_fraction
tag: train/clip_fraction



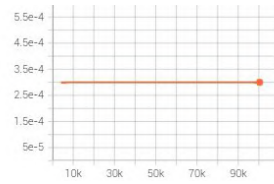
clip_range
tag: train/clip_range



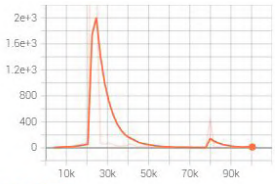
entropy_loss
tag: train/entropy_loss



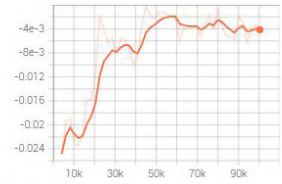
explained_variance
tag: train/explained_variance



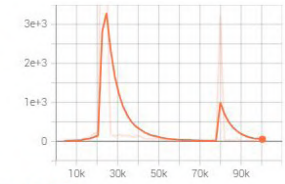
learning_rate
tag: train/learning_rate



loss
tag: train/loss



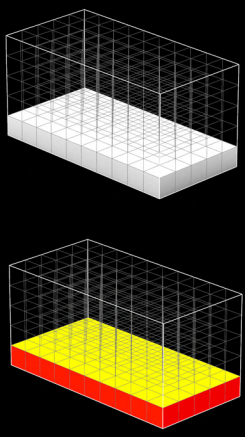
policy_gradient_loss
tag: train/policy_grad_loss



value loss
tag: train/value_loss

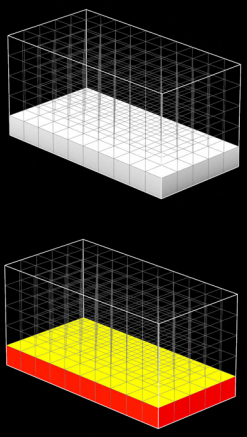
PPO model 0.7

test A



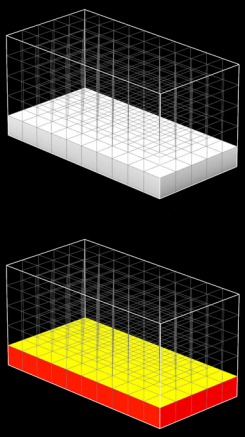
Total reward **-665**
Rew by sunh. **-10**

test B



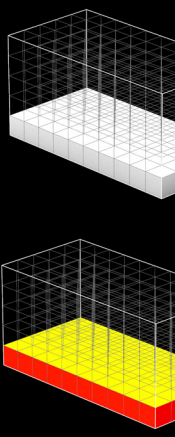
Total reward **-665**
Rew by sunh. **-10**

test C



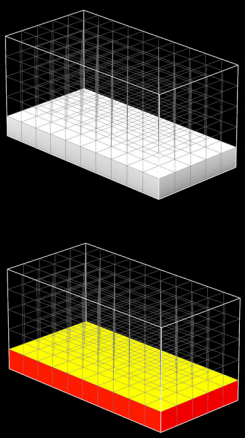
Total reward **-665**
Rew by sunh. **-10**

test D



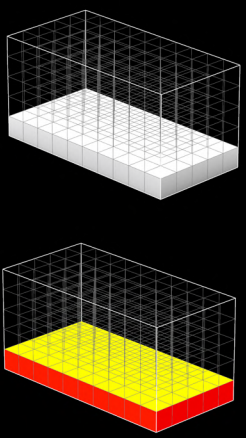
Total reward **-665**
Rew by sunh. **-10**

test F



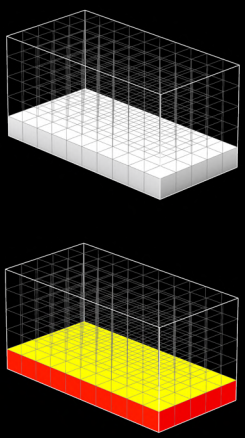
Total reward **-665**
Rew by sunh. **-10**

test G



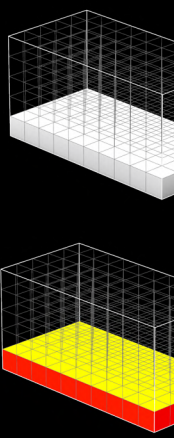
Total reward **-665**
Rew by sunh. **-10**

test H



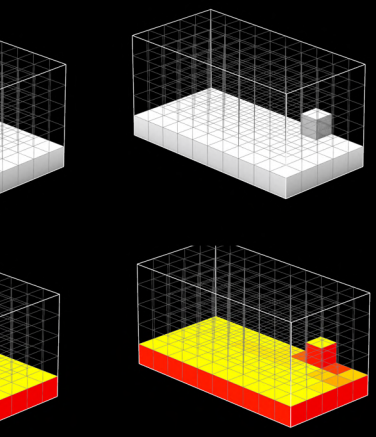
Total reward **-665**
Rew by sunh. **-10**

test I



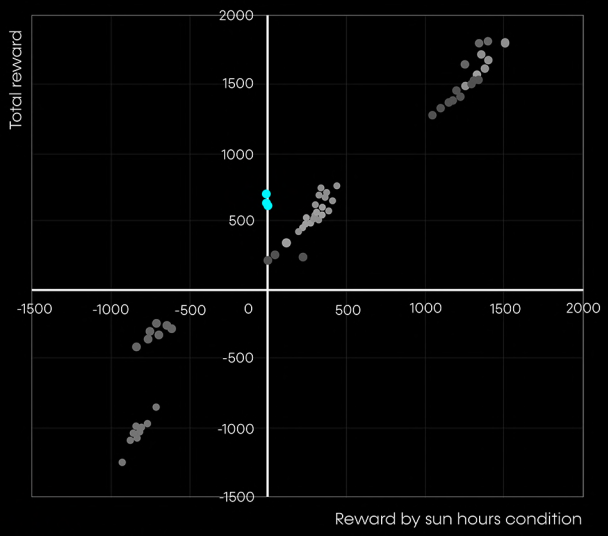
Total reward **-665**
Rew by sunh. **-10**

test E



Total reward **-660**
Rew by sunh. **-10**

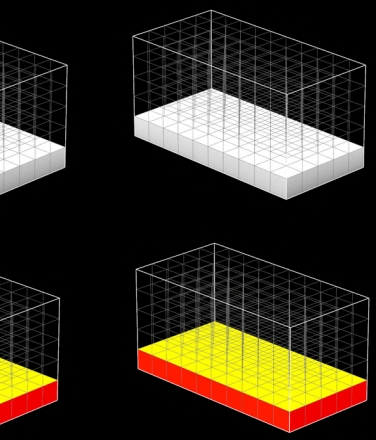
model score



PPO model 0.7 tests

Training timestep = 100.000
Sun hours target = 7h
Assemblage n of part = 50

test J



Total reward **-665**
Rew by sunh. **-10**

Questo è riscontrabile all'interno del set di test effettuato in cui è palese un atteggiamento prettamente deterministico da parte dell'agente, che restituisce modelli molto simili tra loro.

L'addensamento dei blocchi a livello zero denota ulteriori problematiche all'interno della funzione di reward: in particolare l'agente tende a sfruttare molto il reward positivo conseguito per il posizionamento dei blocchi a livello zero, disinteressandosi della sua compatibilità alle condizioni climatiche imposte.

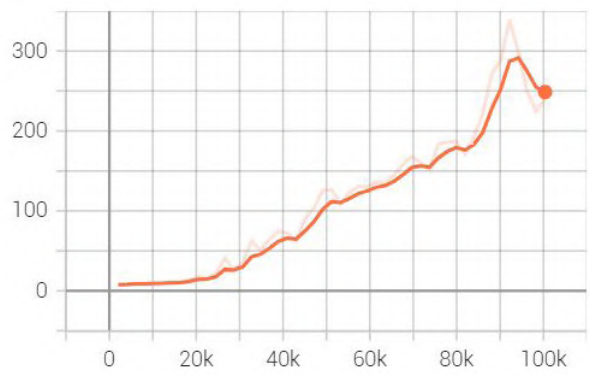
PPO model 0.8

La direzione perseguita nel modello 0.8 è stata quella di snellire ulteriormente la *reward function*, rimuovendo il premio legato alla disposizione dei blocchi al livello zero, al fine di contrastare l'eccessivo consumo di spazio alla base dell'*environment*.

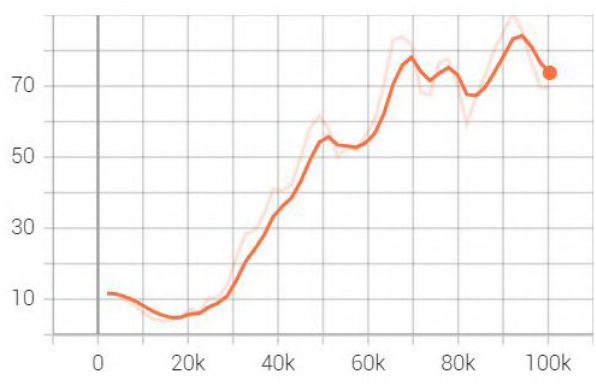
L'andamento della funzione *reward_mean* mostra un picco di reward medio introno ai 300 punti, dato notevole in quanto mette in luce un incremento delle prestazioni del 90% rispetto al picco del modello precedente.

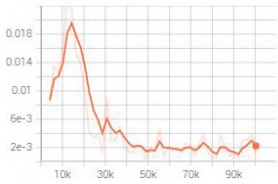
Nonostante questo, il tasso di successo resta ancora molto basso (sotto al 30%). I test inoltre si configurano in grandi cluster sconnessi e ancora lontani dall'obiettivi topologici iniziali.

ep_rew_mean
tag:rollout/ep_rew_mean

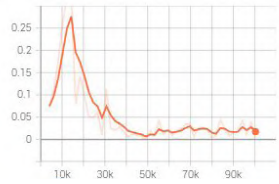


ep_len_mean
tag:rollout/ep_len mean

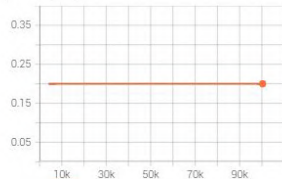




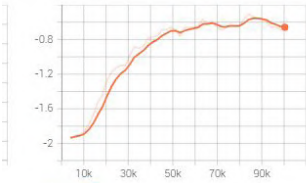
approx_Kl
tag: train/approx_kl



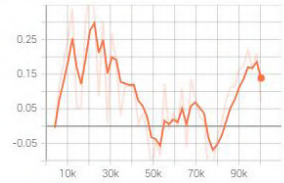
clip_fraction
tag: train/clip_fraction



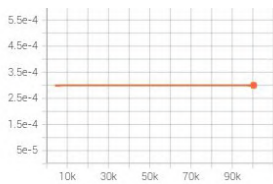
clip_range
tag: train/clip_range



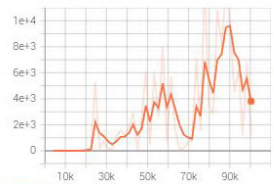
entropy_loss
tag: train/entropy_loss



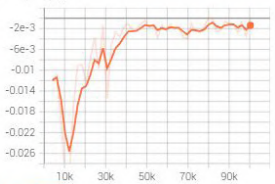
explained_variance
tag: train/explained_variance



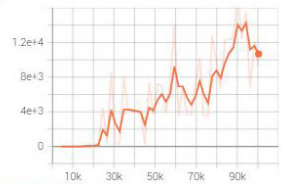
learning_rate
tag: train/learning_rate



loss
tag: train/loss



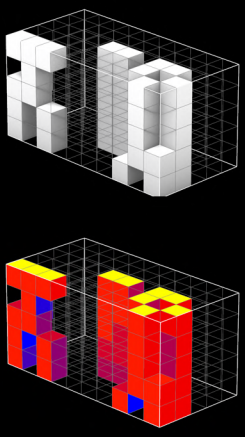
policy_gradient_loss
tag: train/policy_grad_loss



value loss
tag: train/value_loss

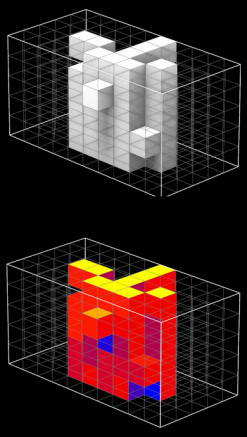
PPO model 0.8

test A



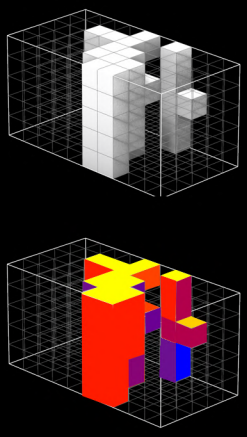
Total reward **760**
Rew by sunh. **435**

test B



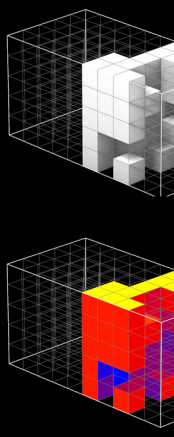
Total reward **695**
Rew by sunh. **330**

test C



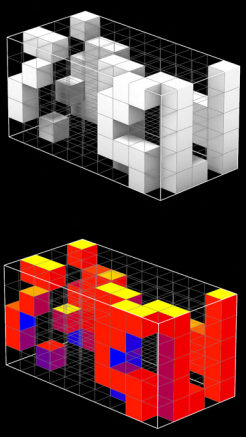
Total reward **725**
Rew by sunh. **345**

test D



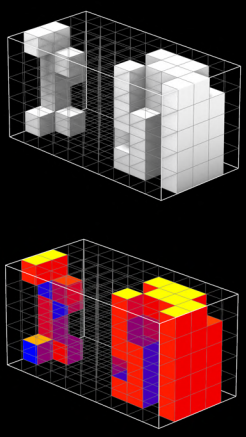
Total reward **700**
Rew by sunh. **345**

test F



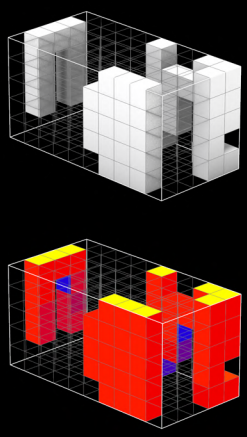
Total reward **535**
Rew by sunh. **305**

test G



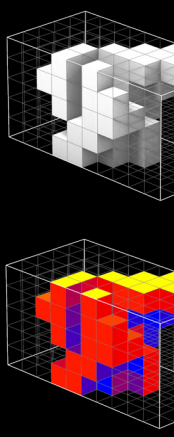
Total reward **620**
Rew by sunh. **300**

test H



Total reward **1795**
Rew by sunh. **1500**

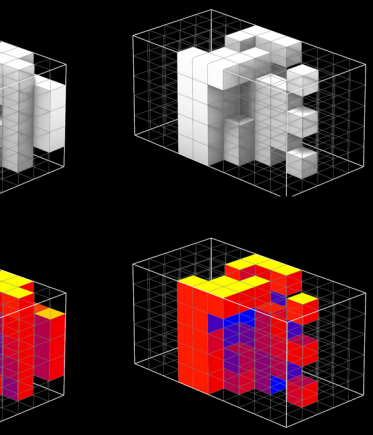
test I



Total reward **715**
Rew by sunh. **355**

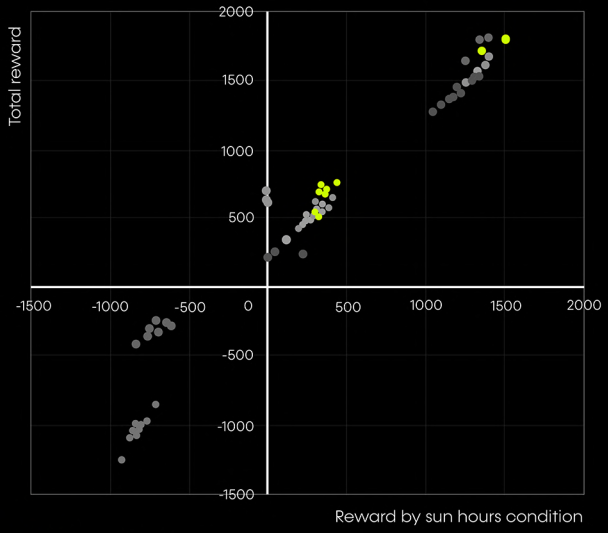
0h 15h

test E



Total reward **685**
Rew by sunh. **360**

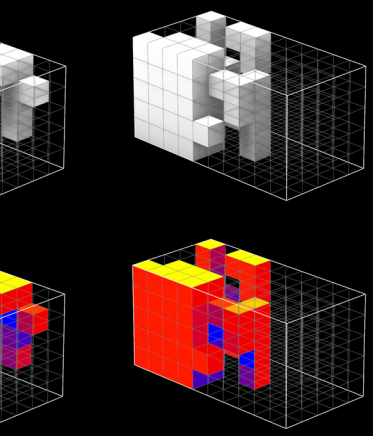
model score



PPO model 0.8 tests

Training timestep = 100.000
Sun hours target = 7h
Assemblage n of part = 50

test J



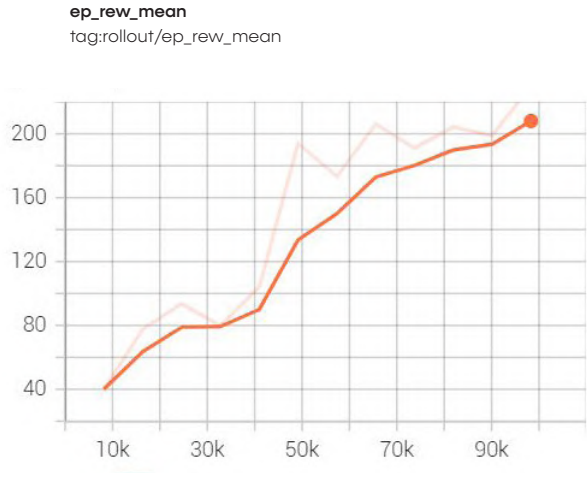
Total reward **1710**
Rew by sunh. **1350**

PPO model 0.9

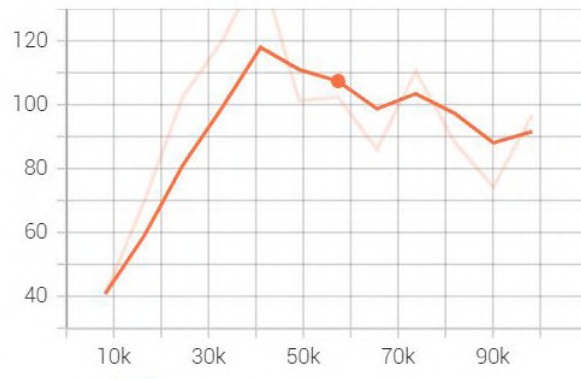
Il problema su ci si è principalmente concentrati in questa nuova versione è stato quello di ridurre la clusterizzazione degli assemblaggi andando ad intervenire sulla ricompensa legata alla presenza di vicini nel posizionamento di un blocco.

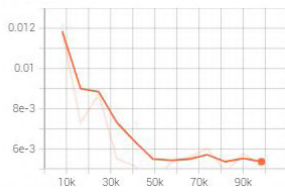
Di conseguenza è stata eliminata la componente cumulativa che assegnava un reward di +5 per ogni vicino presente attorno al blocco appena posizionato. Al suo posto è stata imposta una più semplice condizione if che premia l'agente con una ricompensa di +5 se almeno una delle celle adiacente a quella in cui viene collocato un blocco sia piena.

Questa operazione ha prodotto un training moderatamente più performante rispetto al precedente. Seppure il picco di reward medio sia intorno al 200 (più basso rispetto al modello precedente di circa +300) non bisogna dimenticare che la nuova funzione di reward ammette ricavi ben più bassi ad ogni step temporale.

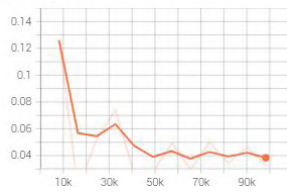


ep_len_mean
tag:rollout/ep_len mean

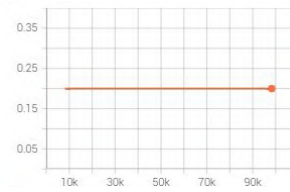




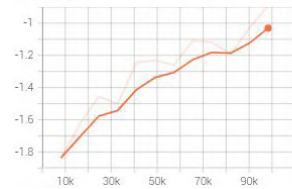
approx_Kl
tag: train/approx_kl



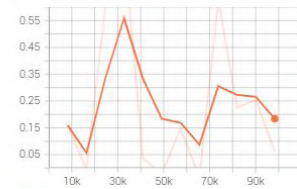
clip_fraction
tag: train/clip_fraction



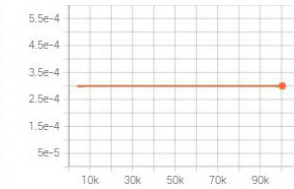
clip_range
tag: train/clip_range



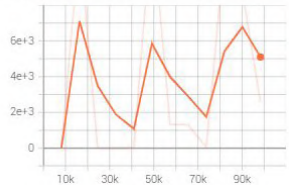
entropy_loss
tag: train/entropy_loss



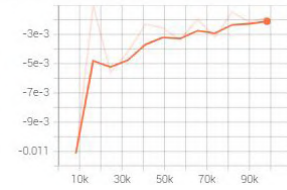
explained_variance
tag: train/explained_variance



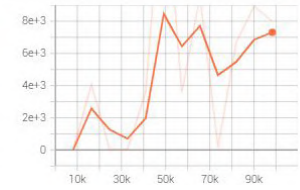
learning_rate
tag: train/learning_rate



loss
tag: train/loss



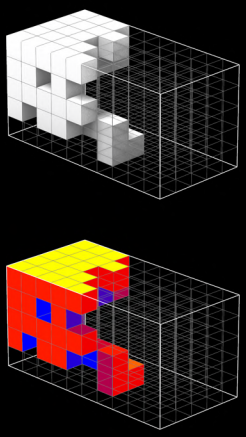
policy_gradient_loss
tag: train/policy_grad_loss



value loss
tag: train/value_loss

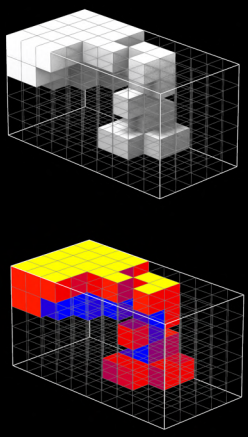
PPO model 0.9

test A



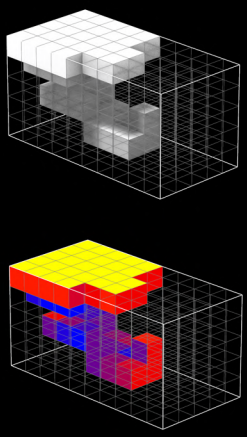
Total reward **260**
Rew by sunh. **500**

test B



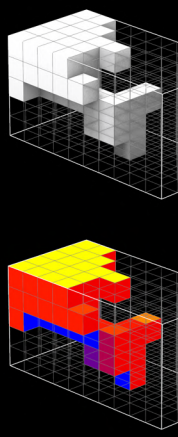
Total reward **250**
Rew by sunh. **485**

test C



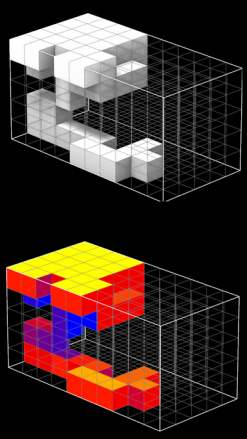
Total reward **475**
Rew by sunh. **235**

test D



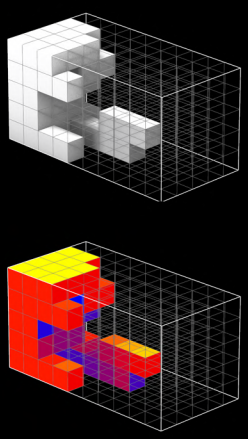
Total reward **345**
Rew by sunh. **120**

test F



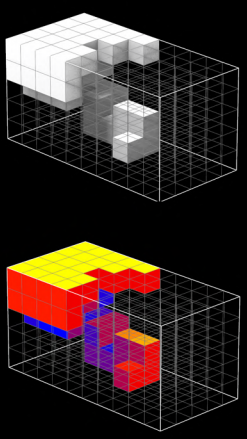
Total reward **555**
Rew by sunh. **310**

test G



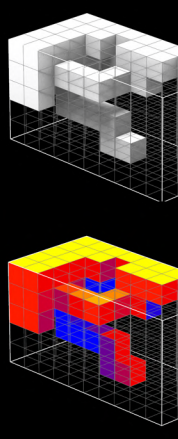
Total reward **1560**
Rew by sunh. **1325**

test H



Total reward **520**
Rew by sunh. **245**

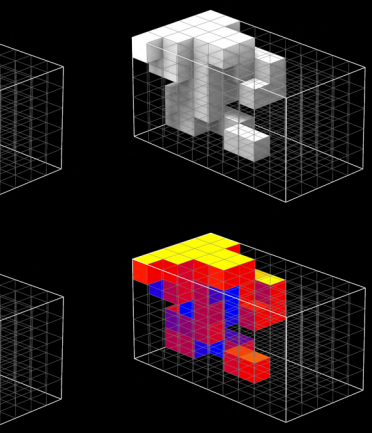
test I



Total reward **520**
Rew by sunh. **245**

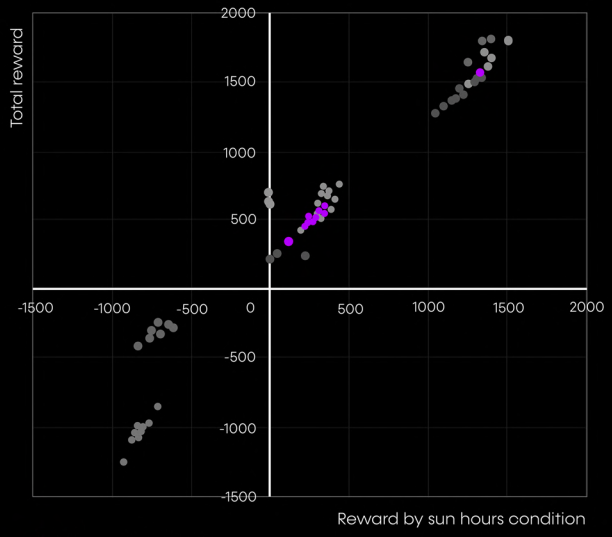
0h 15h

test E



Total reward **455**
Rew by sunh. **220**

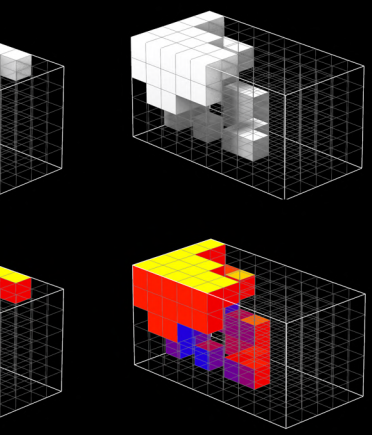
model score



PPO model 0.9 tests

Training timestep = 100.000
Sun hours target = 7h
Assemblage n of part = 50

test J



Total reward **605**
Rew by sunh. **335**

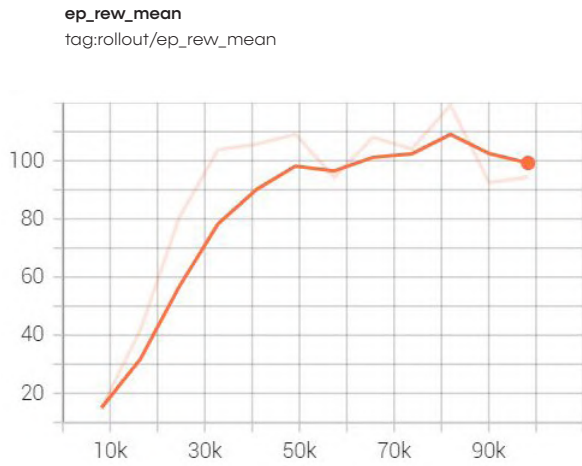
Il range di reward possibile in questo modello è infatti 0,+15 mentre nello scorso 0, +40, a testimoniare un progressivo miglioramento. Sebbene le performance del modello siano in incremento, la strada verso la restituzione di assemblaggi totalmente connessi dal punto di vista della circolazione interna è ancora in salita i quanto continuano ad essere presenti cluster spaziali isolati ed elementi privi di connessioni.

PPO model 0.10

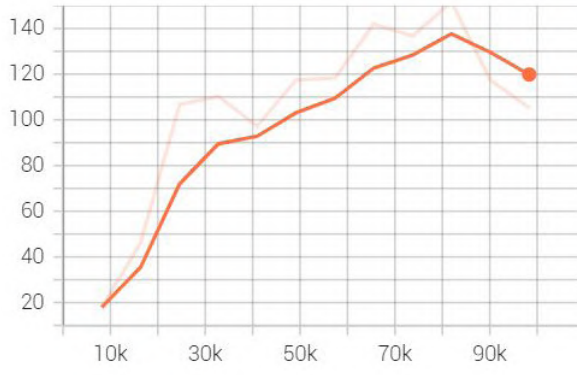
Arrivati a questo punto, per incentivare la creazione di sistemi spaziali più coerenti con i nostri obiettivi iniziali, si è deciso di condurre un nuovo training, implementando una *game over condition*, e mantenendo inalterata la *reward function.*

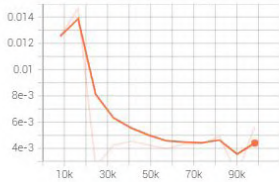
La *game over condition* funziona nel seguente modo: nel momento in cui un blocco viene posizionato isolato nello spazio, l'episodio si interrompe. Unica eccezione è fatta per i blocchi isolati posizionati a livello zero. L'idea è di introdurre questa condizione nel solo processo di training al fine di promuovere la creazione di assemblaggi più densi e connessi internamente.

L'andamento del training mette in luce come il modello progressivamente abbia imparato a mantenersi in vita più a lungo al fine di accumulare più reward possibili. Sebbene l'andamento del grafico *mean reward* sia sospettosamente in calo rispetto ai modelli precedenti bisogna fare una grossa precisazione. Il fatto di aver introdotto una *game over condition* fa si che la lunghezza media degli episodi sia notevolmente minore rispetto a quella dei casi precedenti. Ciò rende sostanzialmente incomparabile dal punto di vista delle metriche questo modello con i suoi predecessori.

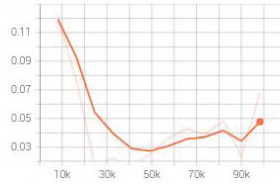


ep_len_mean
tag:rollout/ep_len mean

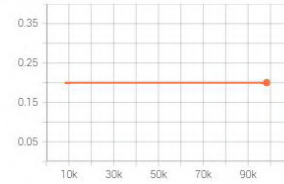




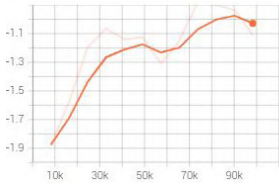
approx_Kl
tag: train/approx_kl



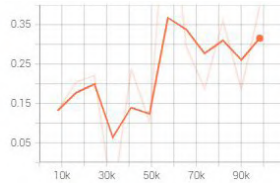
clip_fraction
tag: train/clip_fraction



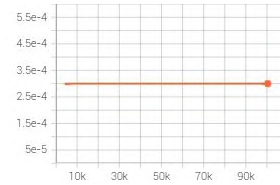
clip_range
tag: train/clip_range



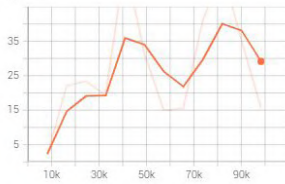
entropy_loss
tag: train/entropy_loss



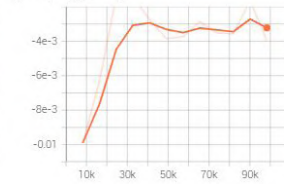
explained_variance
tag: train/explained_variance



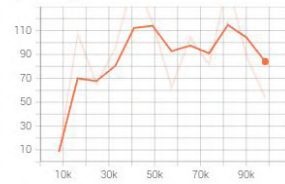
learning_rate
tag: train/learning_rate



loss
tag: train/loss



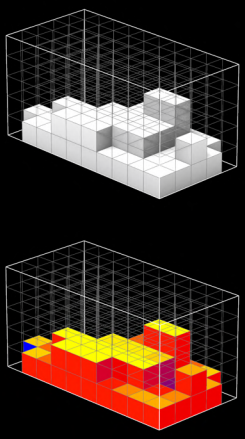
policy_gradient_loss
tag: train/policy_grad_loss



value loss
tag: train/value_loss

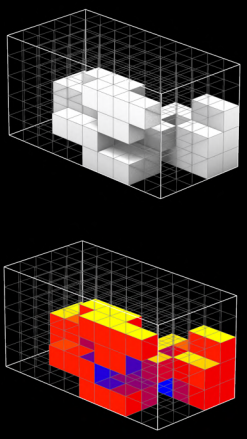
PPO model 0.10

test A



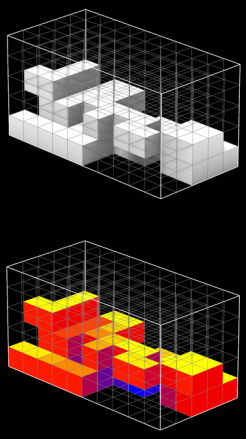
Total reward **645**
Rew by sunh. **410**

test B



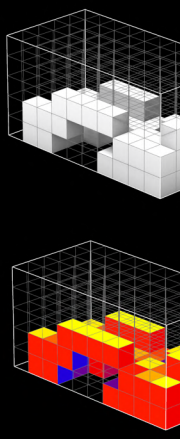
Total reward **1485**
Rew by sunh. **1255**

test C



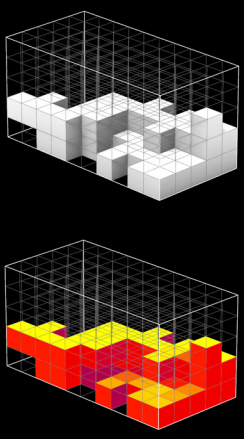
Total reward **1520**
Rew by sunh. **1300**

test D



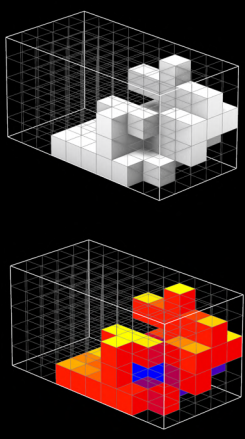
Total reward **1530**
Rew by sunh. **1320**

test F



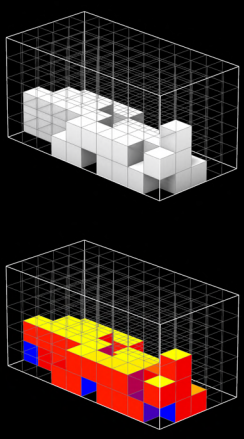
Total reward **435**
Rew by sunh. **205**

test G



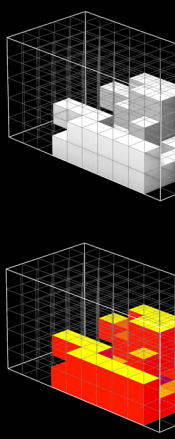
Total reward **520**
Rew by sunh. **295**

test H



Total reward **1680**
Rew by sunh. **1400**

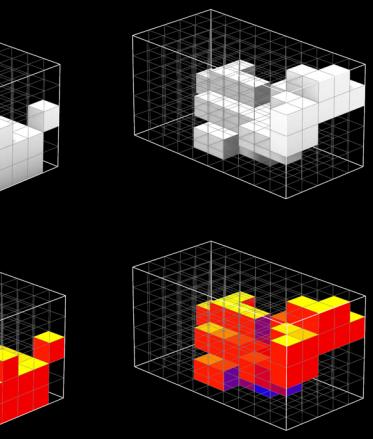
test I



Total reward **1680**
Rew by sunh. **1440**

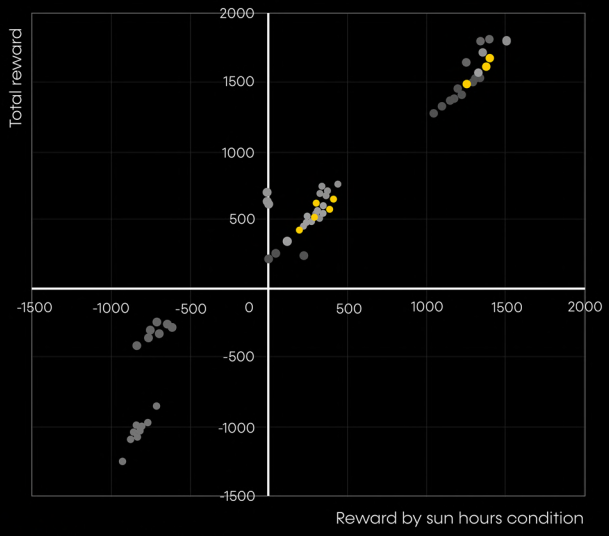
0h 15h

test E



Total reward **535**
Rew by sunh. **300**

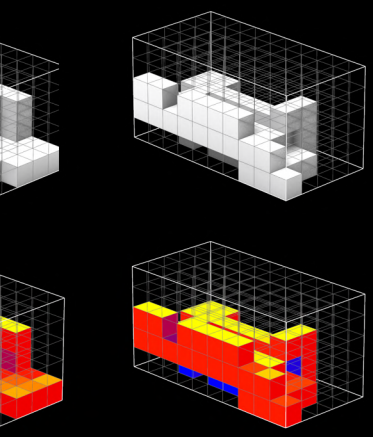
model score



PPO model 0.10 tests

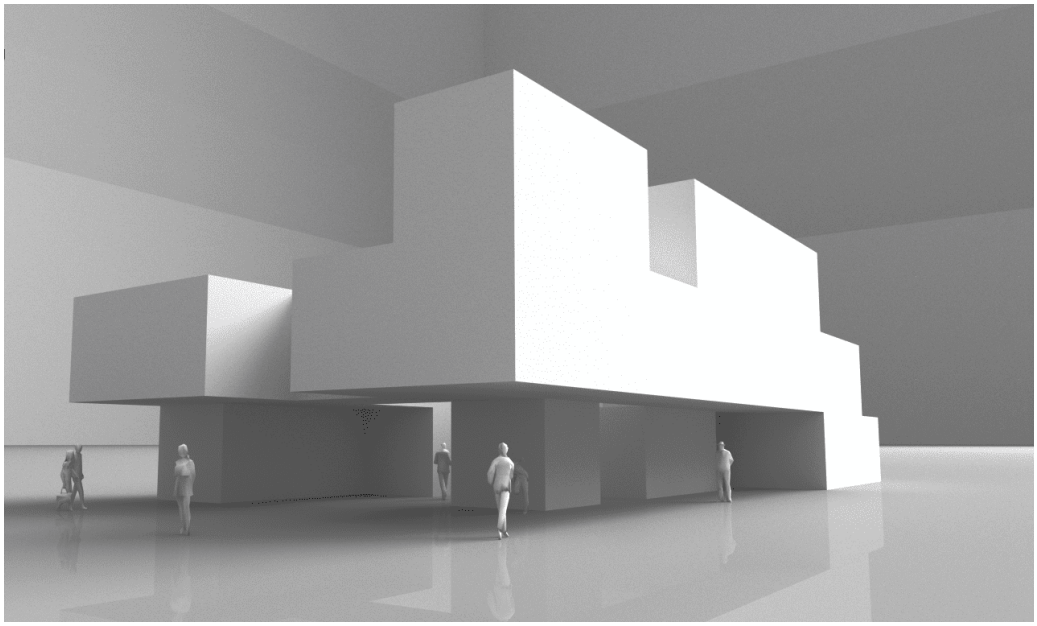
Training timestep = 100.000
Sun hours target = 7h
Assemblage n of part = 50

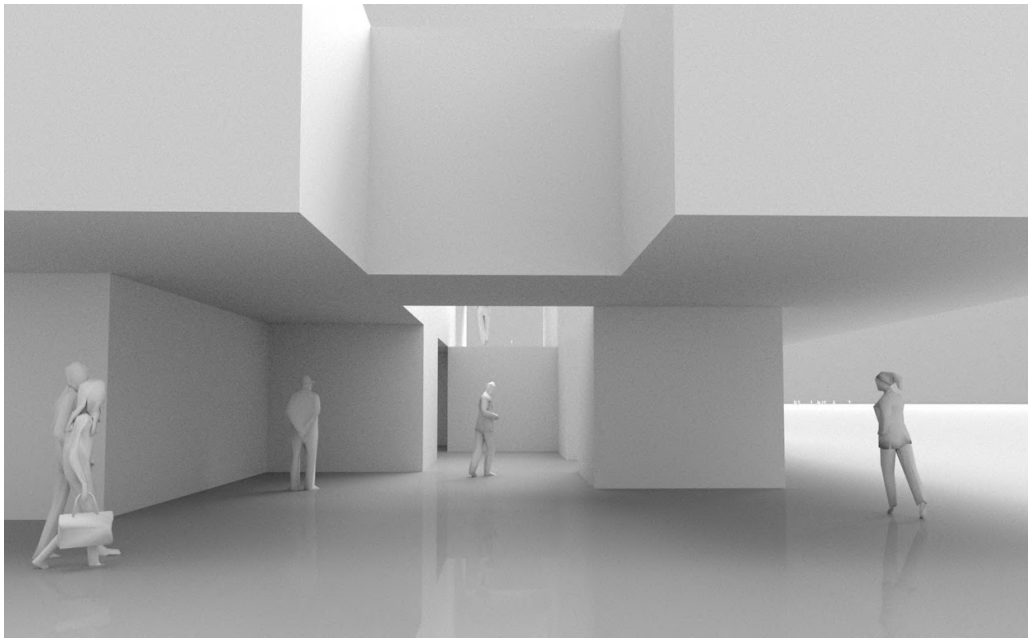
test J

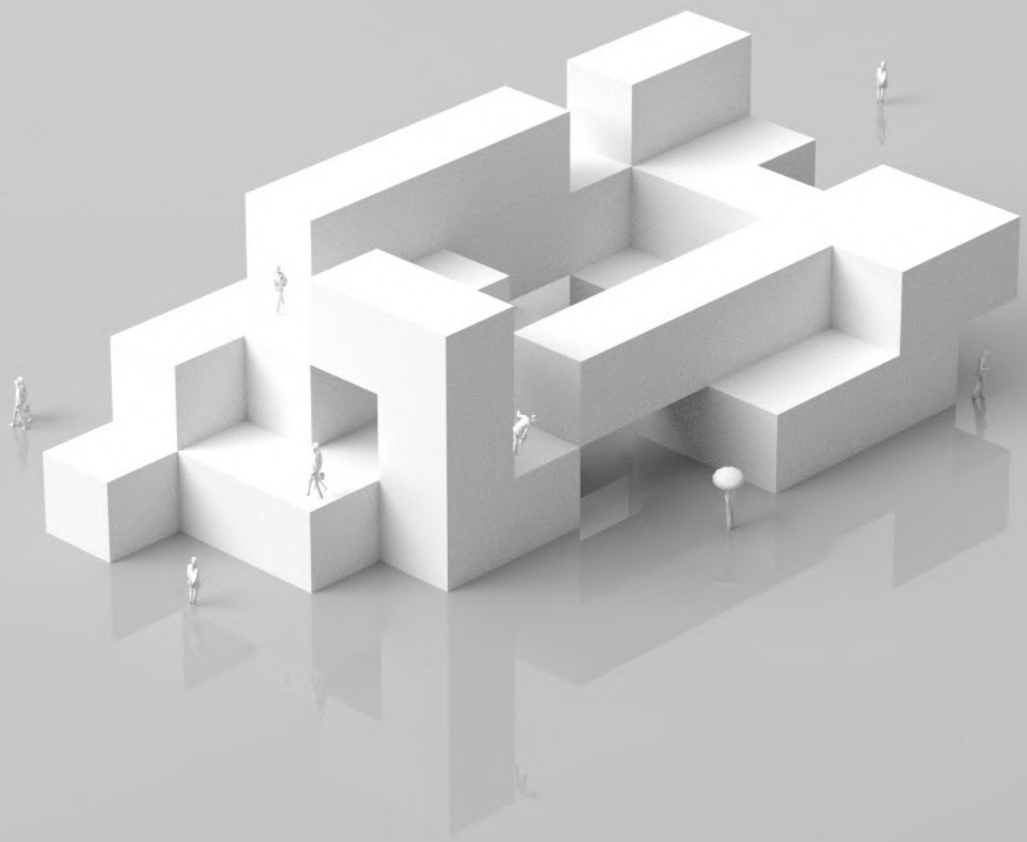


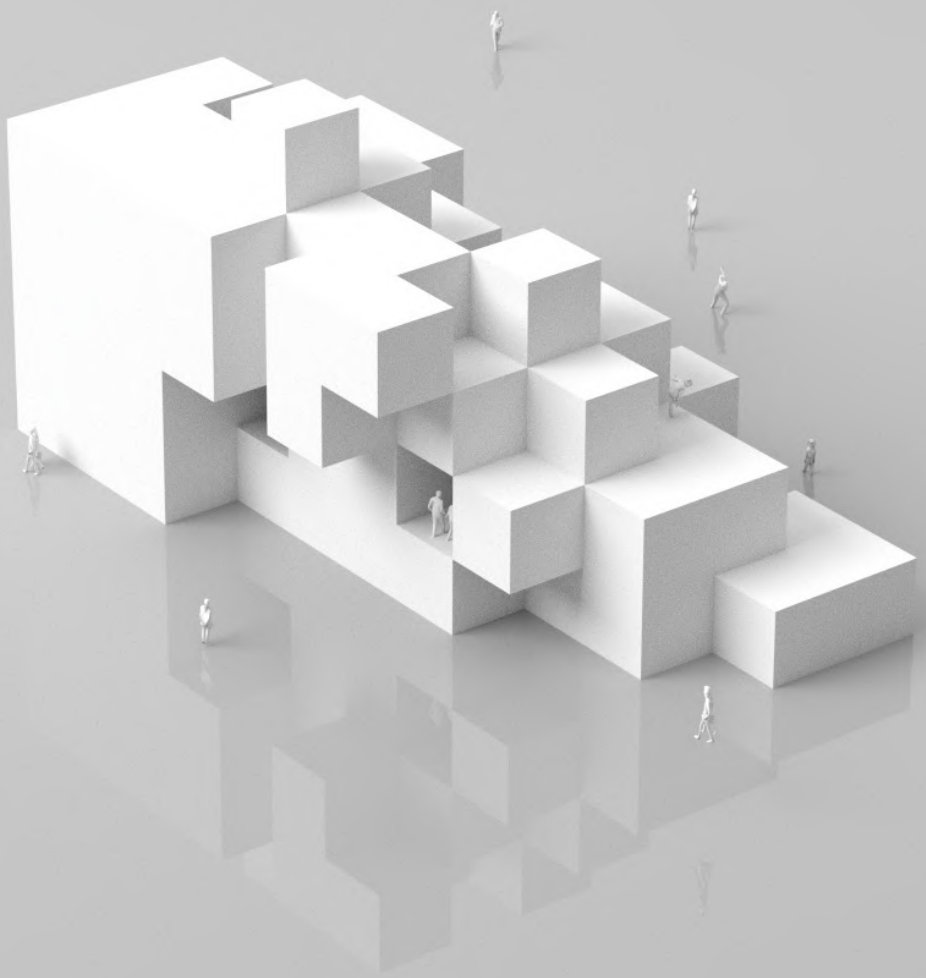
Total reward **1610**
Rew by sunh. **1380**

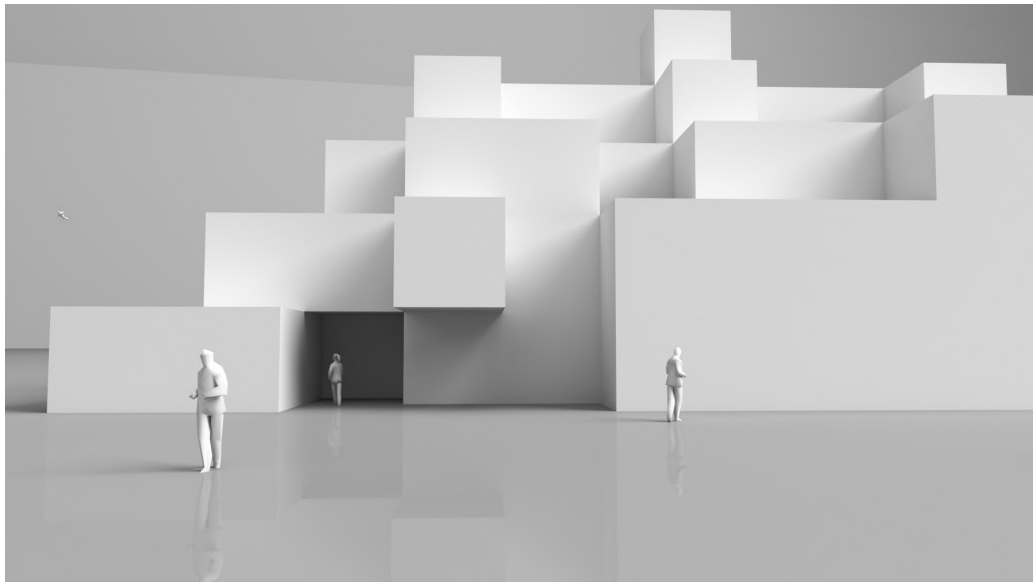
Molto diverso è l'esito dei test condotti in assenza di *game over condition*. Come ben visibile sia nel grafico *model score*, sia negli assemblaggi spaziali restituiti il tasso di successo dell'agente risulta essere molto più elevato (intorno al 40%) e i sistemi spaziali che se ne creano tendono ad essere molto più validi dal punto di vista della connettività. L'obiettivo adesso è quello di consolidare il modello, addestrandolo per un periodo di formazione più lungo.









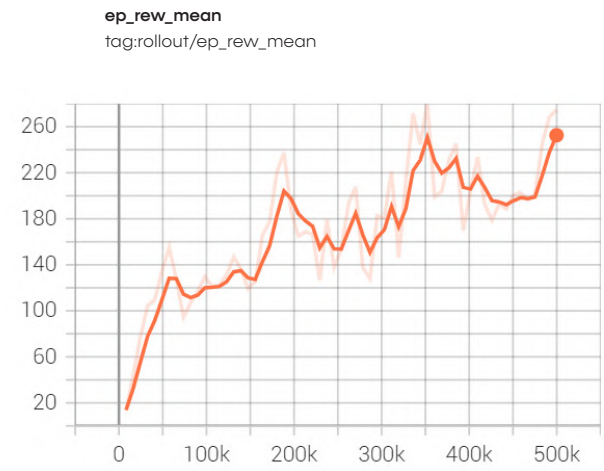


PPO model 0.11

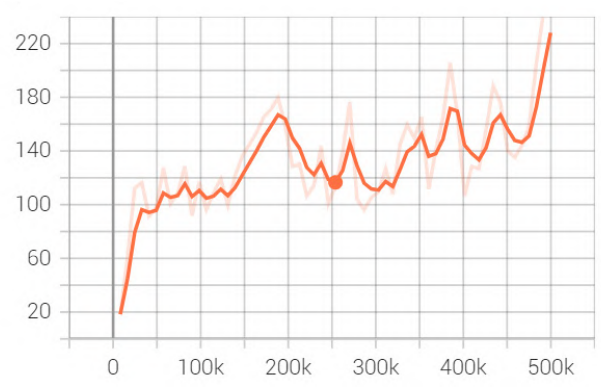
I diversi tentativi condotti sin ora ci hanno condotto verso una direzione compatibile agli obiettivi di questa ricerca.

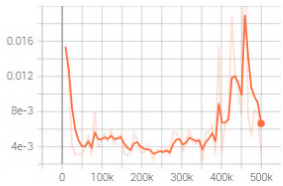
Per consolidare i risultati ottenuti fino a questo punto si è passati ad eseguire un training più lungo. Di fatto l'estensione della durata del training è uno dei metodi più funzionali all'ottenimento di modelli più performanti.

In questo caso si è deciso di estendere le iterazioni del training a 500.000 e di spostare la fascia di illuminazione target da 7h a 9h. Questa seconda decisone è stata portata avanti con l'intento di validare il processo definito sin qui anche su condizioni target differenti.

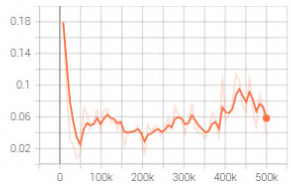


ep_len_mean
tag:rollout/ep_len mean

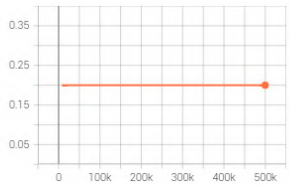




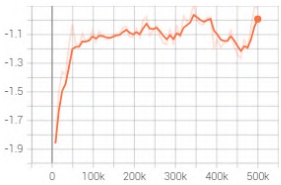
approx_Kl
tag: train/approx_kl



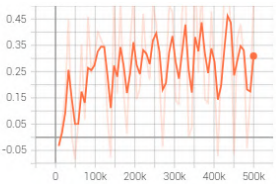
clip_fraction
tag: train/clip_fraction



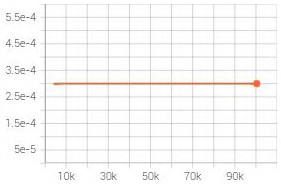
clip_range
tag: train/clip_range



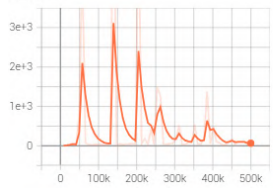
entropy_loss
tag: train/entropy_loss



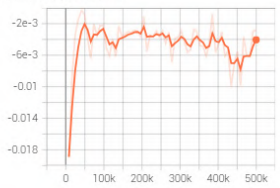
explained_variance
tag: train/explained_variance



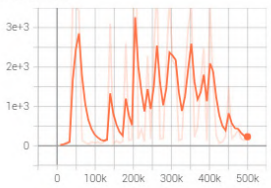
learning_rate
tag: train/learning_rate



loss
tag: train/loss



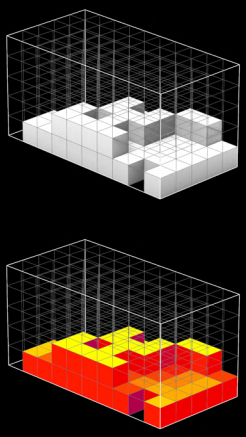
policy_gradient_loss
tag: train/policy_grad_loss



value loss
tag: train/value_loss

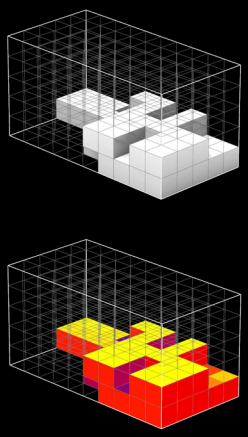
PPO model 0.11

test A



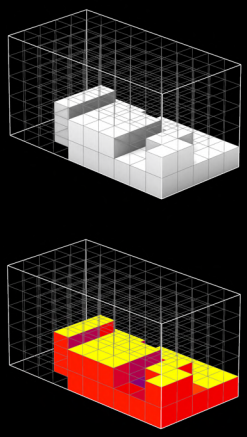
Total reward **1500**
Rew by sunh. **1285**

test B



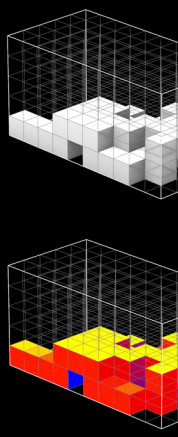
Total reward **1365**
Rew by sunh. **1145**

test C



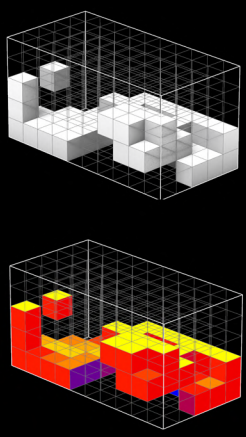
Total reward **1380**
Rew by sunh. **1165**

test D



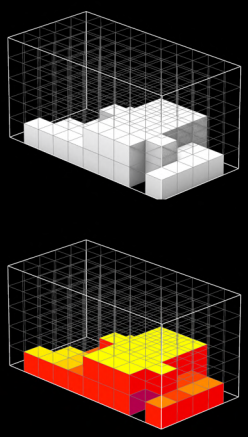
Total reward **1270**
Rew by sunh. **1040**

test F



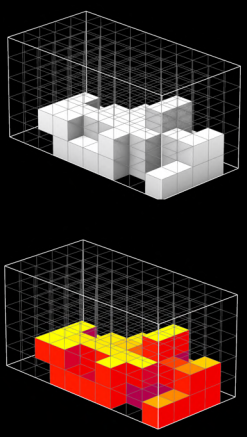
Total reward **215**
Rew by sunh. **0**

test G



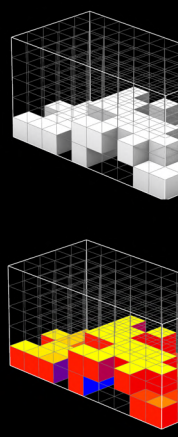
Total reward **1325**
Rew by sunh. **1095**

test H



Total reward **255**
Rew by sunh. **40**

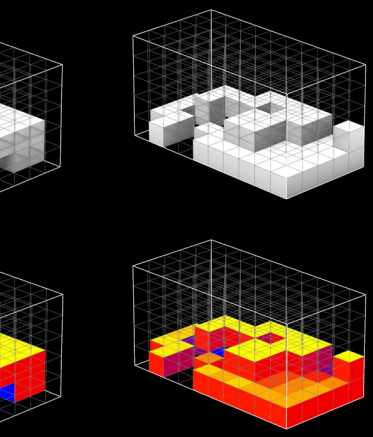
test I



Total reward **240**
Rew by sunh. **225**

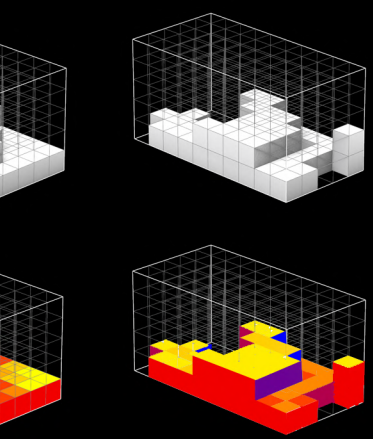
0h 15h

test E



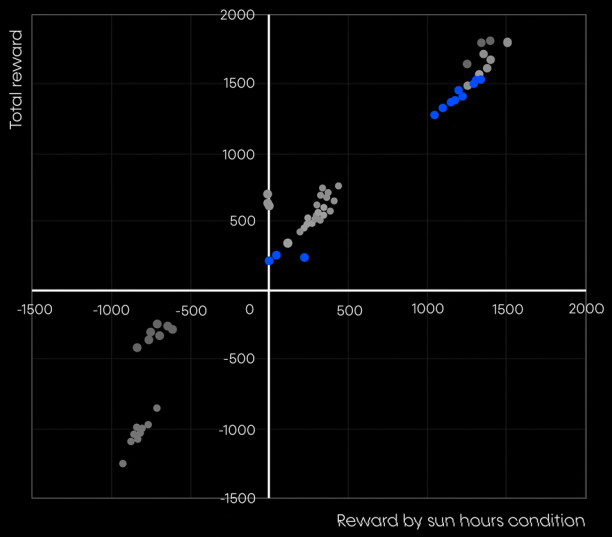
Total reward **45**
Rew by sunh. **265**

test J



Total reward **1425**
Rew by sunh. **1215**

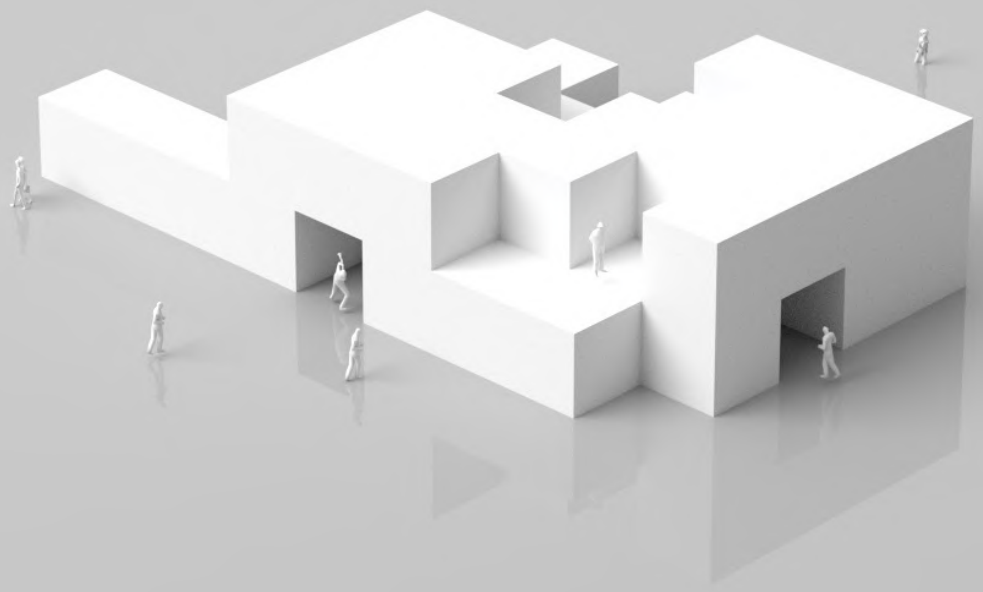
model score

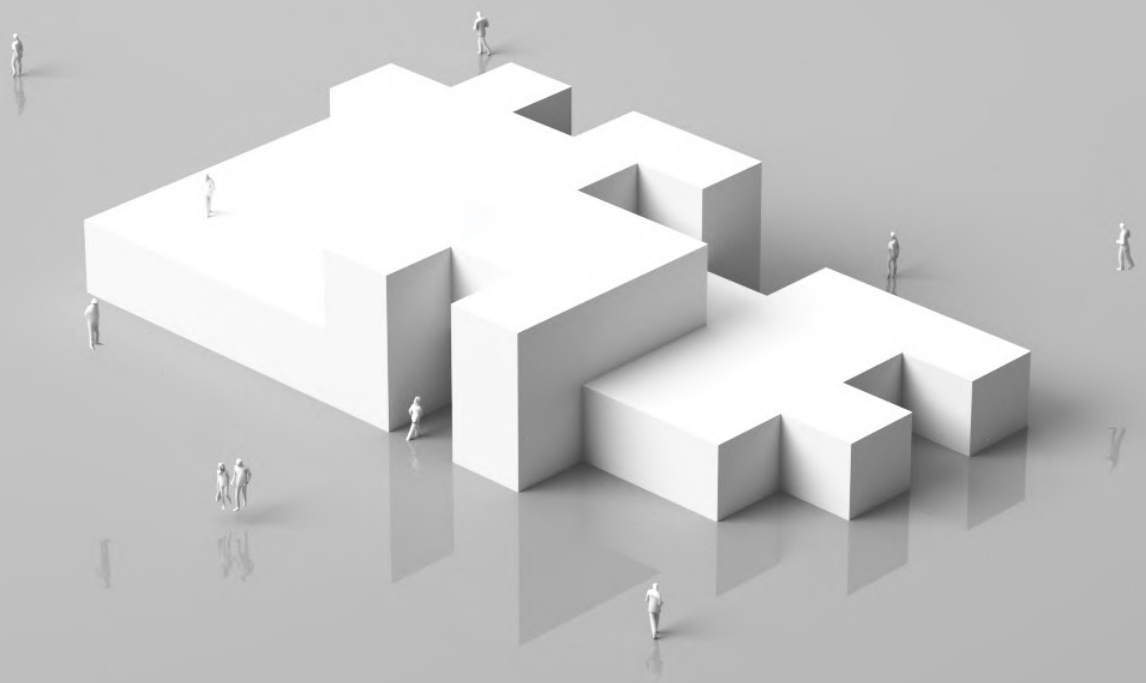


PPO model 0.11 tests

Training timestep = 500.000
Sun hours target = 9h
Assemblage n of part = 50

Come da previsione un periodo di addestramento più duraturo ha generato una significativa crescita delle performance. Dal punto di vista della compatibilità dei sistemi spaziali sviluppati con le condizioni climatiche imposte, il tasso di successo dell'agente si aggira nei pressi dell'80%.





Comparison table & limits

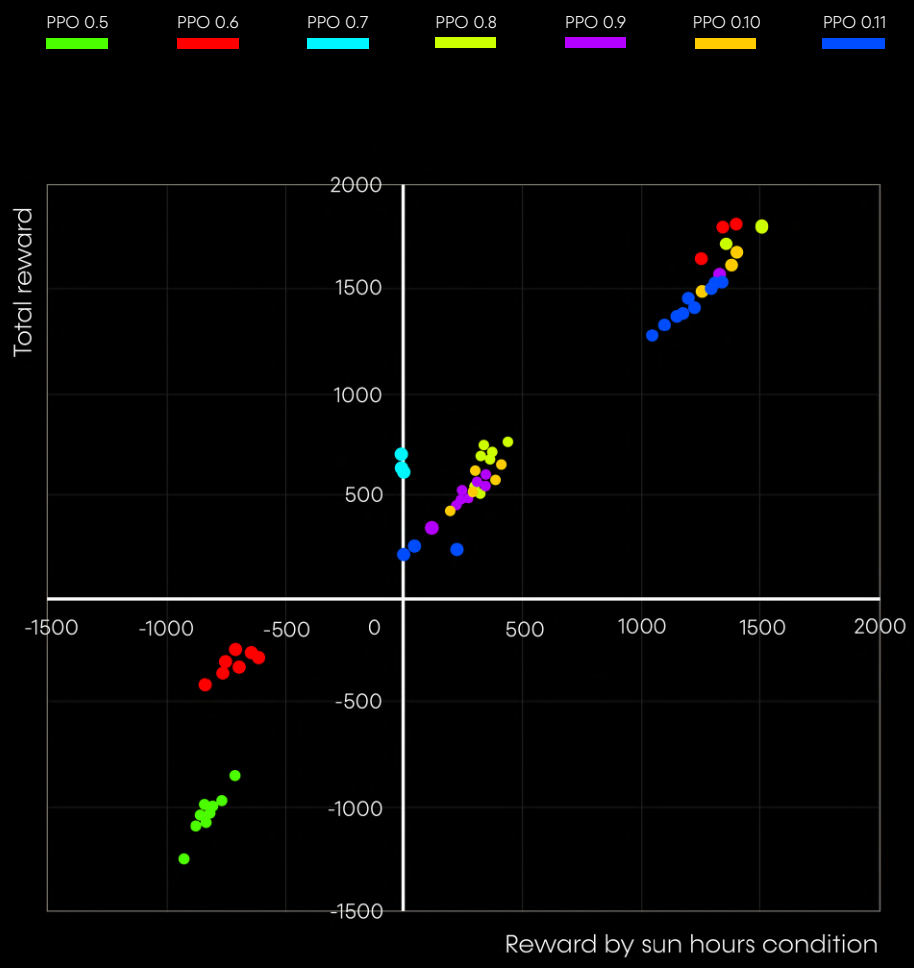
I dati ottenuti in questa serie di test sono racchiusi nella seguente tabella di confronto. Comparando le informazioni sui test effettuati è possibile individuare nei modelli 0.10 e 0.11 quelli più performanti e compatibili agli obiettivi di questa ricerca.

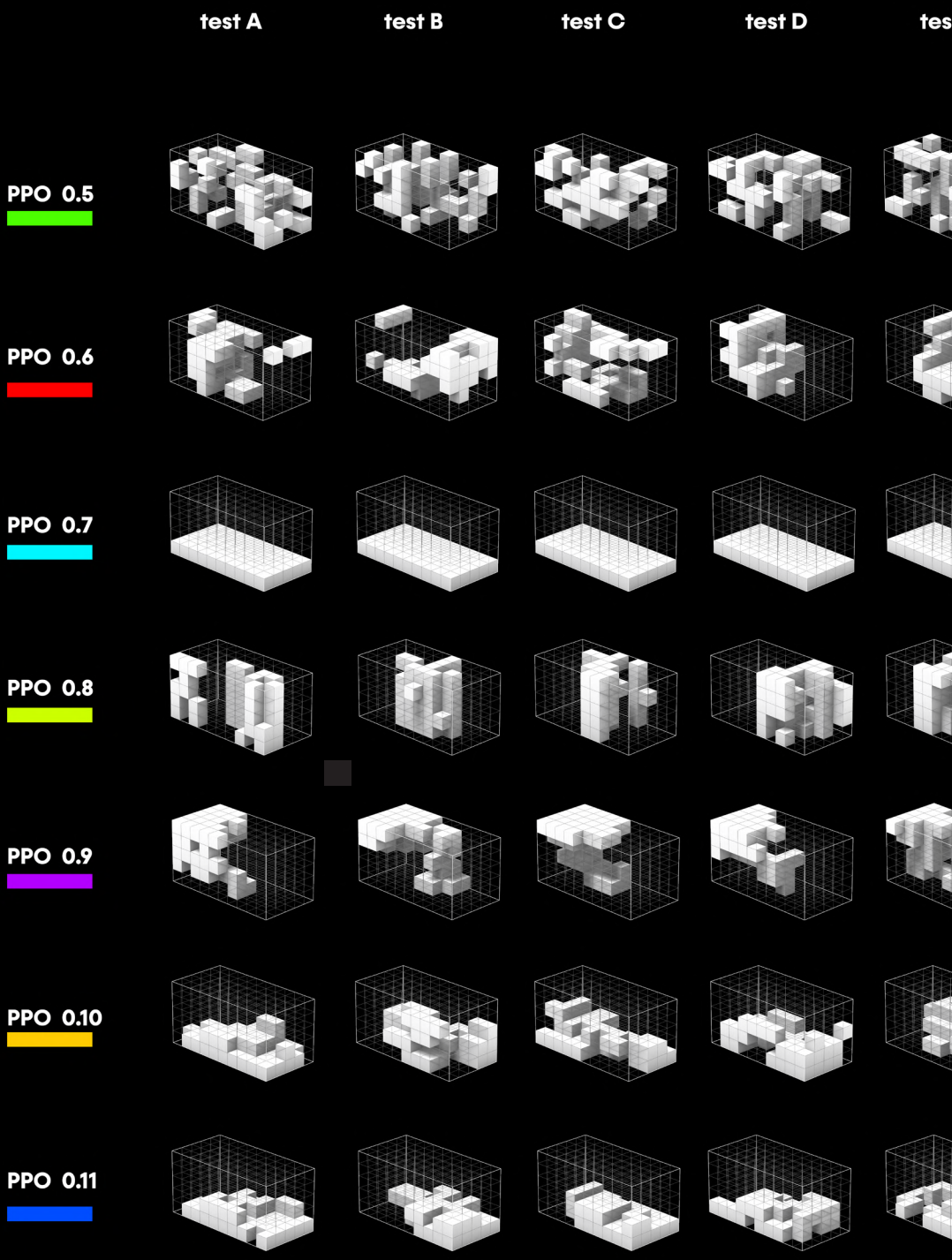
Le performance raggiunte segnano un primo ottimo punto di arrivo in quanto si è riusciti a definire partendo da zero una corretta integrazione tra gli strumenti di Grasshopper, Python e LadyBug per creare un modello in grado di restituire assemblaggi spaziali sviluppati inglobando nella maniera più efficiente possibile le condizioni climatiche di contesto. Inoltre, anche se in maniera pressoché semplificata, il modello tiene conto anche della topologia degli assemblaggi, riuscendo a garantire in diversi casi una totale connettività spaziale agli stessi.

Nonostante ciò resta ancora aperta e non percorribile la questione dell'estendibilità dell'environment. Infatti il tipo di observation space utilizzato in questi modelli ha il limite di non essere estendibile ad ambienti di dimensione differente. Operativamente ciò significa che se volessimo testare il nostro modello su un'ambiente più grande o più piccolo rispetto a quella utilizzata sin ora, dovremmo condurre nuovamente un training.

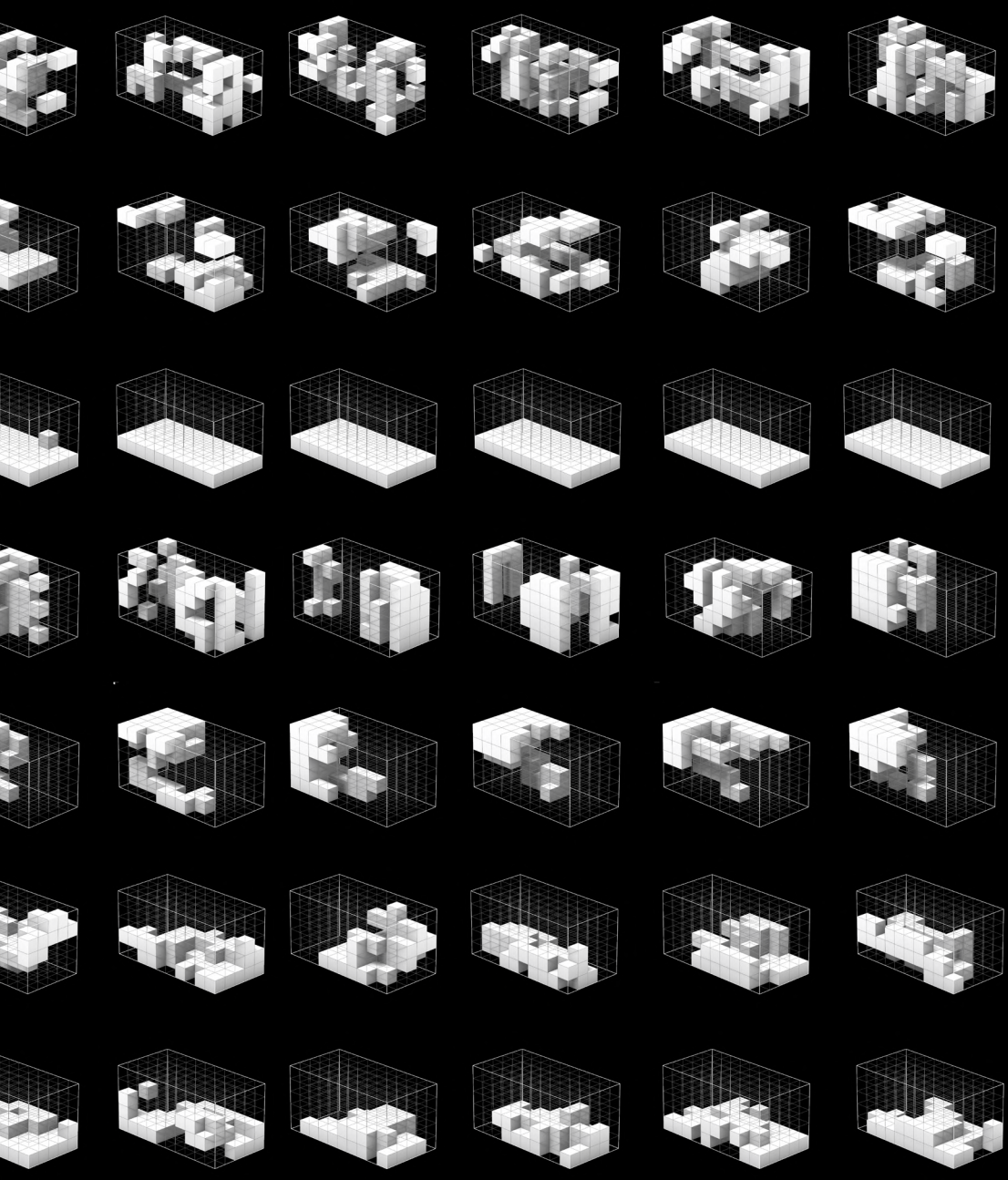
_f40
Scatterplot di confronto tra i modelli PPO sviluppati

Questo pone un grosso limite al concetto di scalabilità del sistema, che è essenziale in questa ricerca.


PPO 0.5
PPO 0.6
PPO 0.7
PPO 0.8
PPO 0.9
PPO 0.10
PPO 0.11
Total reward
2000
1500
1000
500
-500
-1000
-1500
-1500
-1000
-500
0
500
1000
1500
2000
Reward by sun hours condition


test A
test B
test C
test D
tes
PPO 0.5
PPO 0.6
PPO 0.7
PPO 0.8
PPO 0.9
PPO 0.10
PPO 0.11

t E test F test G test H test I test J



3.5 Sun hours based scalable generative system

L'obiettivo di questa seconda parte della ricerca è quello di rendere il modello indipendente da forma e dimensioni dell'environment. In altre parole si mira ad ottenere un modello finale scalabile..

Il concetto di scalabilità del sistema cova al suo interno tre diverse tematiche:

- L'applicabilità del modello in *environment* di estensione diversa e potenzialmente illimitata.

- La possibilità di costruire sistemi costituiti da un numero di parti potenzialmente infinito.

- Adattabilità del modello alla variazione di contesto.

PPO model 2.0

Per tradurre la caratteristiche precitate in termini di computazionali è stato avviato un processo di semplificazione dell'*observation space*: PPO 0.11 e i suoi predecessori ricevevano dall'environment contemporaneamente feedback topologici di carattere locale (*agent +neighbours position*) e globale (*cells state*) (**_f41**).

La lista *cells state* rappresenta una mappatura dello stato di ogni cella che fa parte dell'*environment*. Il suo utilizzo concede da un lato il vantaggio di avere un più alto numero di informazioni da sfruttare per centrare gli obiettivi di training, e dall'altro ha lo svantaggio di rendere il sistema non scalabile (oltre che computazionalmente più dispendioso).

Quello che succede è che se l'*agent* viene addestrato su un environment ricevendo un vettore di *observations* di estensione pari a *num_cells* (numero di celle), esso sarà vincolato ad essere utilizzabile solo in spazi di uguale estensione.

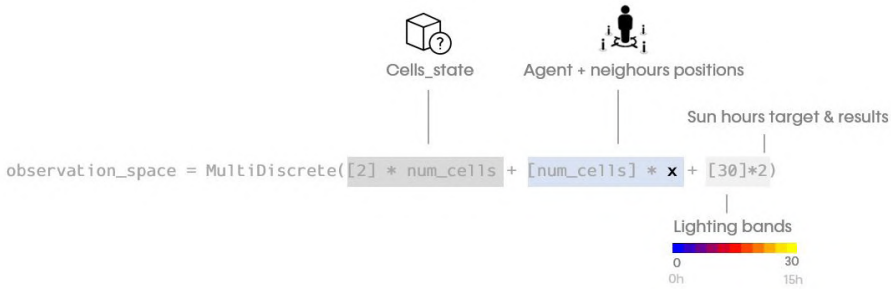
Per questo motivo in questa versione sono stati rimossi i feedback riguardanti lo stato globale della topologia del sistema.
Di conseguenza, il nuovo *observation space* implementato (**_f42**) si basa su due soli contributi:

La lista *neighbours_state* racchiude la mappatura dello stato di ogni cella adiacente a quella in cui si trova l'agent in un vettore composto da 6 elementi compresi tra 0 e 2 dove:
- 0 = cella vuota
- 1 = cella piena
- 2 = vicino non presente (per le celle che si trovano in posizioni di frontiera).

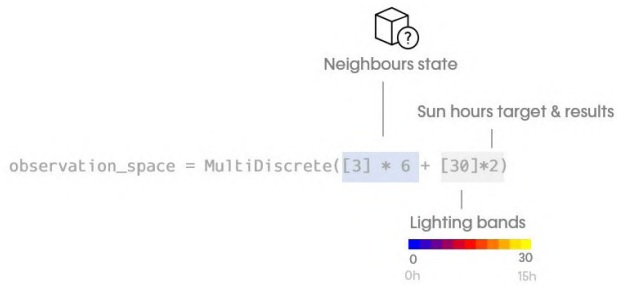
La lista bidimensionale che contiene il target di sun hours globale preimpostato e il risultato dell'analisi ad ogni iterazione.

La *reward function* è rimasta inalterata rispetto a quella utilizzata nel modello 0.11.
Il modello è stato addestrato all'interno di una matrice dimensionale di estensione 5 x 10 x 5 con un target di sun hours pari a 7 ore e con un numero di 50 parti da posizionare.

PPO 0.11 - observation space (_f41)

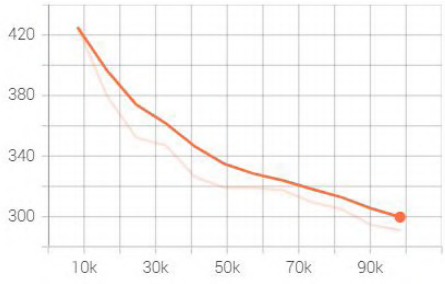


PPO 2.0 - observation space (_f42)

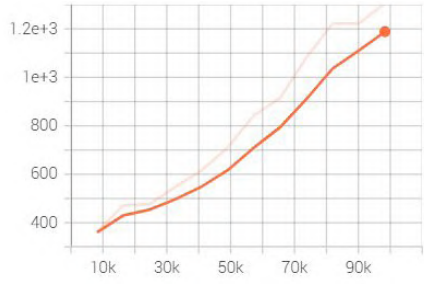


L'andamento del training evidenzia un processo di apprendimento stabile, accompagnato dal raggiungimento di performance elevate. Il grafico *ep_rew_mean* arriva ad un massimo storico che sfiora i 1200 punti di reward a testimonianza di un modello che nella maggior parte dei casi è in grado di permutare e aggregare le unità in gioco per ottenere assemblaggi che misurano un numero di ore di luce diurna medio nei limiti del target considerato.
Testando il modello su assemblaggi di dimensione differente, ottenuti

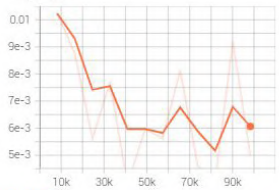
ep_len_mean
tag:rollout/ep_len mean



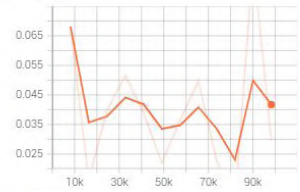
ep_rew_mean
tag:rollout/ep_rew_mean



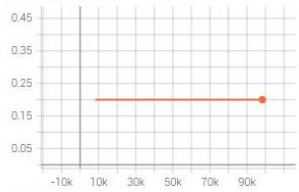
approx_kl
tag: train/approx_kl



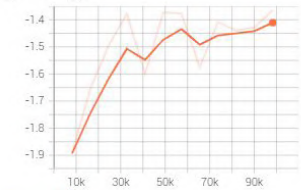
clip_fraction
tag: train/clip_fraction



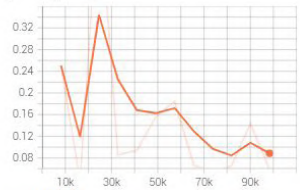
clip_range
tag: train/clip_range



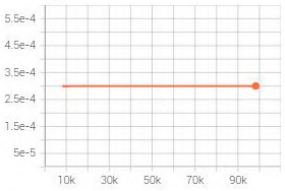
entropy_loss
tag: train/entropy_loss



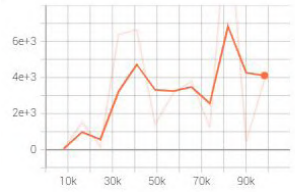
explained_variance
tag: train/explained_variance



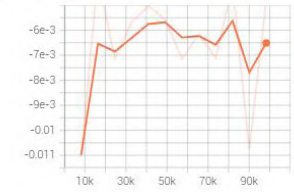
learning_rate
tag: train/learning_rate



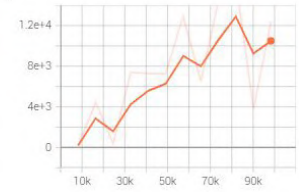
loss
tag: train/loss



policy_gradient_loss
tag: train/policy_grad_loss



value loss
tag: train/value_loss



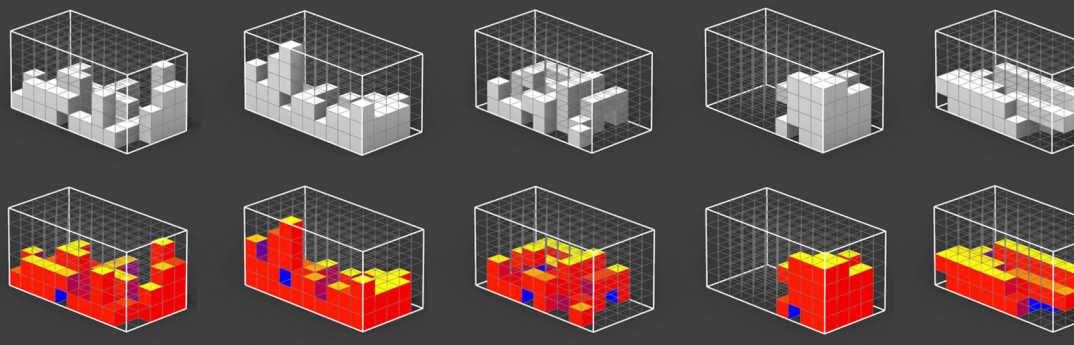
anche variando l'estensione della matrice tridimensionale che definisce lo spazio voxelizzato, si nota come le prestazioni positive in fase di training vengano tradotte in sistemi spaziali in linea con le caratteristiche ipotizzate. Nel 73% dei casi gli assemblaggi rientrano nella fascia di illuminazione target.

Il grado di successo generale dei test è considerabile sufficiente per porre fine al processo di validazione del modello e avviare una fase più approfondita di studio delle qualità architettoniche degli assemblaggi da esso derivati al variare delle condizioni di contesto.

Seppur la rimozione dei feedback topologici sullo stato globale dell'environment, ha consentito di definire un sistema estendibile ad ogni spazio, una precisazione va fatta: anche se concettualmente la procedura appena indicata consentirebbe di generare sistemi spaziali ad estensioni illimitate, gli sforzi computazionali relativi agli strumenti utilizzati pongono comunque dei limiti (quantomeno di natura temporale) alla creazione di sistemi spaziali formati da un numero di unità che superi il 1000.

Questo rappresenta un collo di bottiglia dovuto alle ingenti risorse di calcolo richieste ad ogni iterazione principalmente da Grasshopper e in particolare dagli strumenti di analisi di Ladybug Tools.

Environment **250 cells** (5x10x5)



Total reward **1710**
Sun hours **6.6h**
Target **7h**

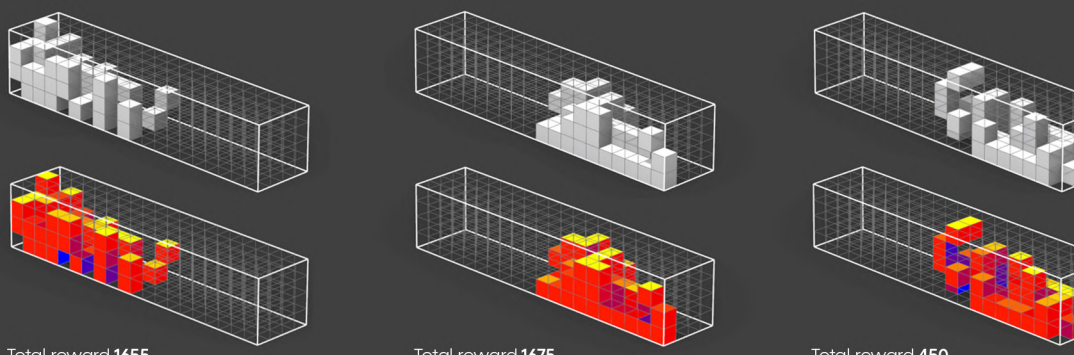
Total reward **1680**
Sun hours **6.7h**
Target **7h**

Total reward **1580**
Sun hours **6.7h**
Target **7h**

Total reward **1660**
Sun hours **6.8h**
Target **7h**

Total reward **1695**
Sun hours **6.5h**
Target **7h**

Environment **320 cells** (4x16x4)

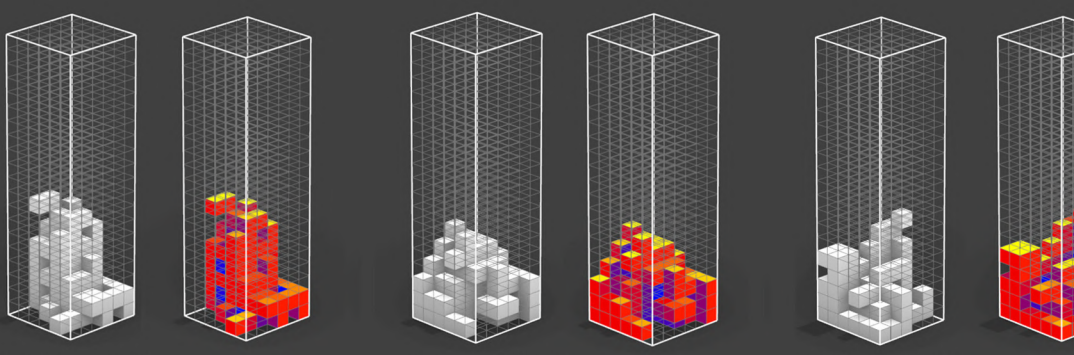


Total reward **1655**
Sun hours **6.5h**
Target **7h**

Total reward **1675**
Sun hours **6.6h**
Target **7h**

Total reward **450**
Sun hours **6.0h**
Target **7h**

Environment **720 cells** (6x6x20)

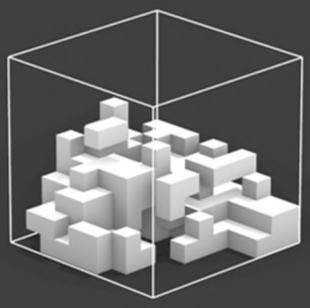


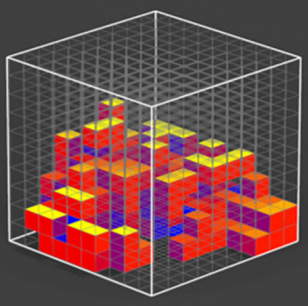
Total reward **2230**
Sun hours **6.6h**
Target **7h**

Total reward **2145**
Sun hours **6.9h**
Target **7h**

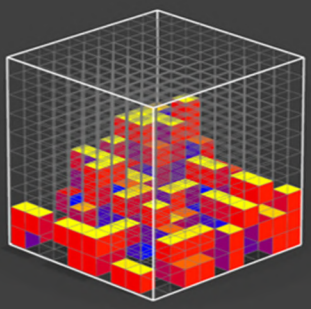
Total reward **1185**
Sun hours **6.1h**
Target **7h**

Environment **1000 cells** (10x10x10)

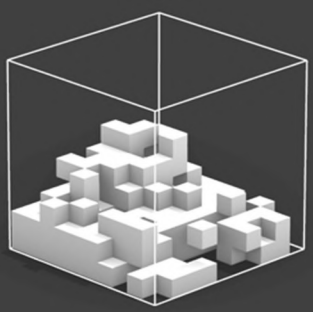


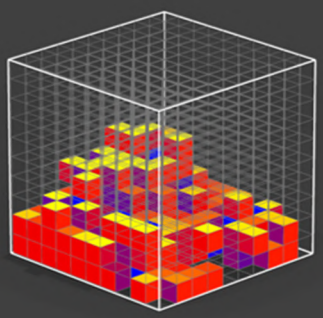


Total reward **2655**
Sun hours **5.9h**
Target **7h**

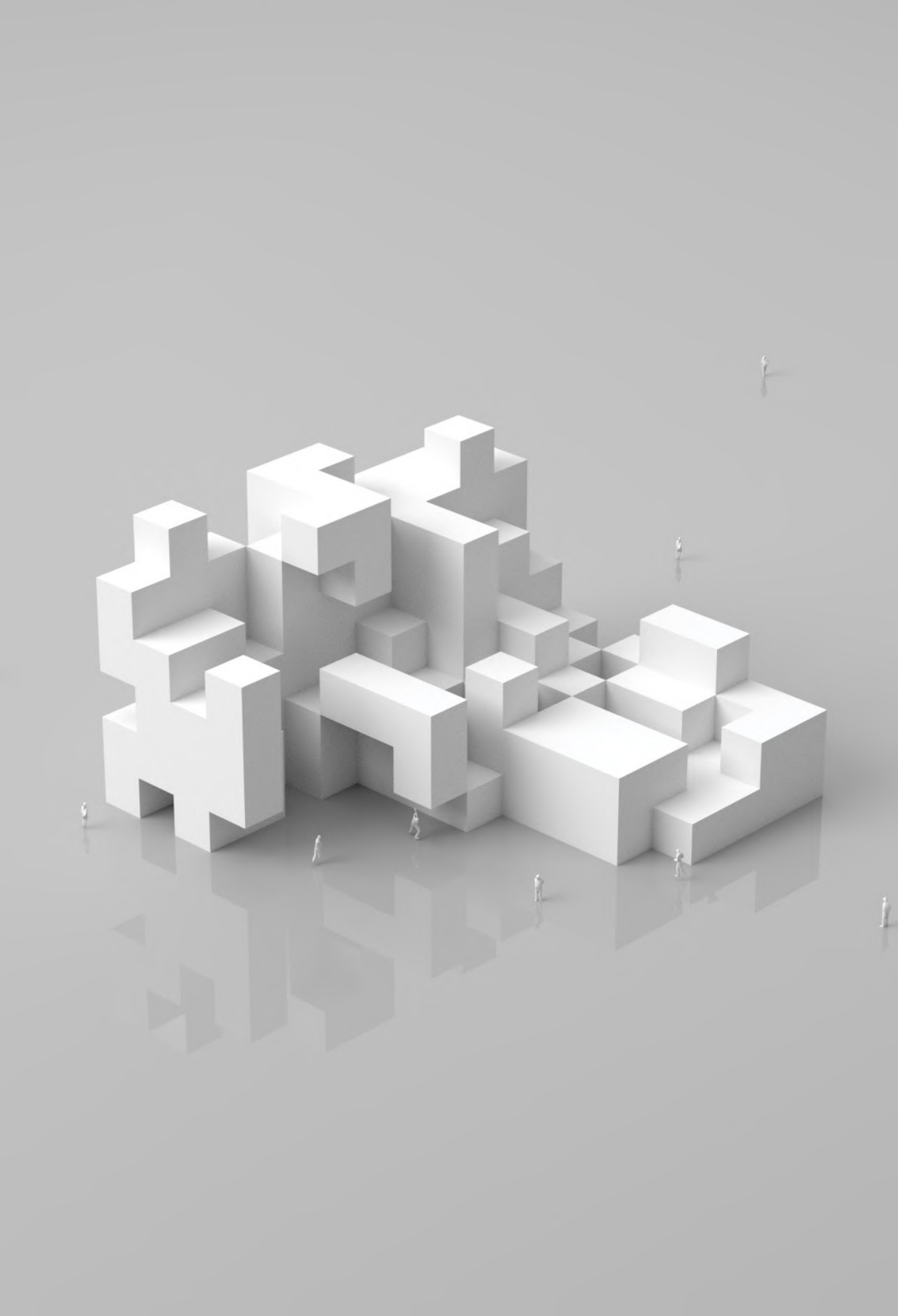


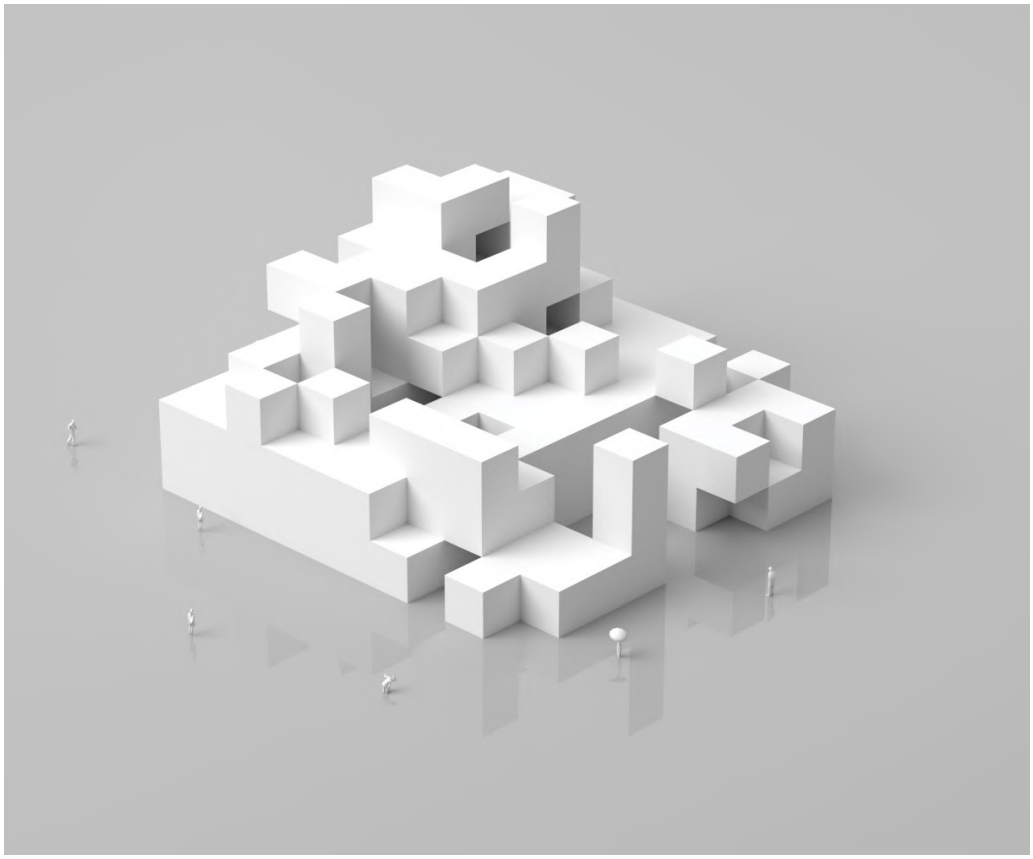
Total reward **3385**
Sun hours **6.5h**
Target **7h**

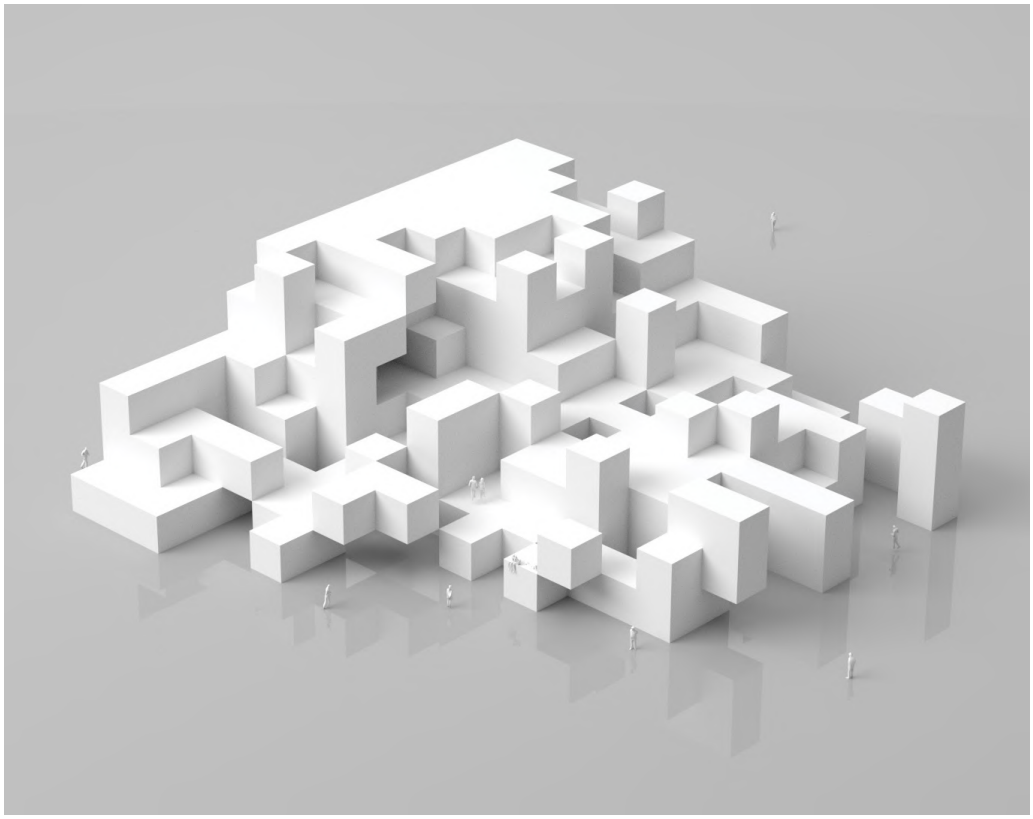


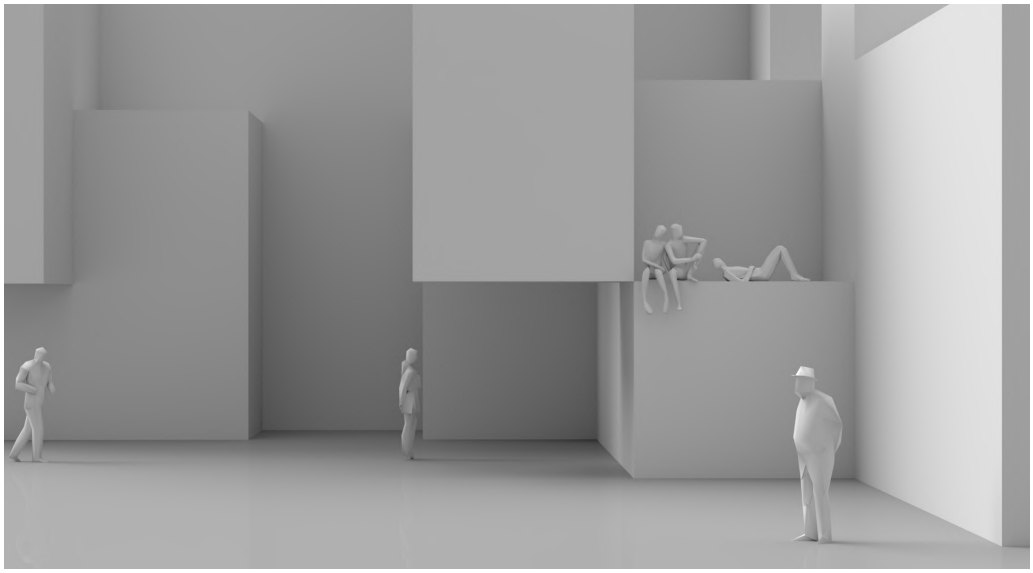


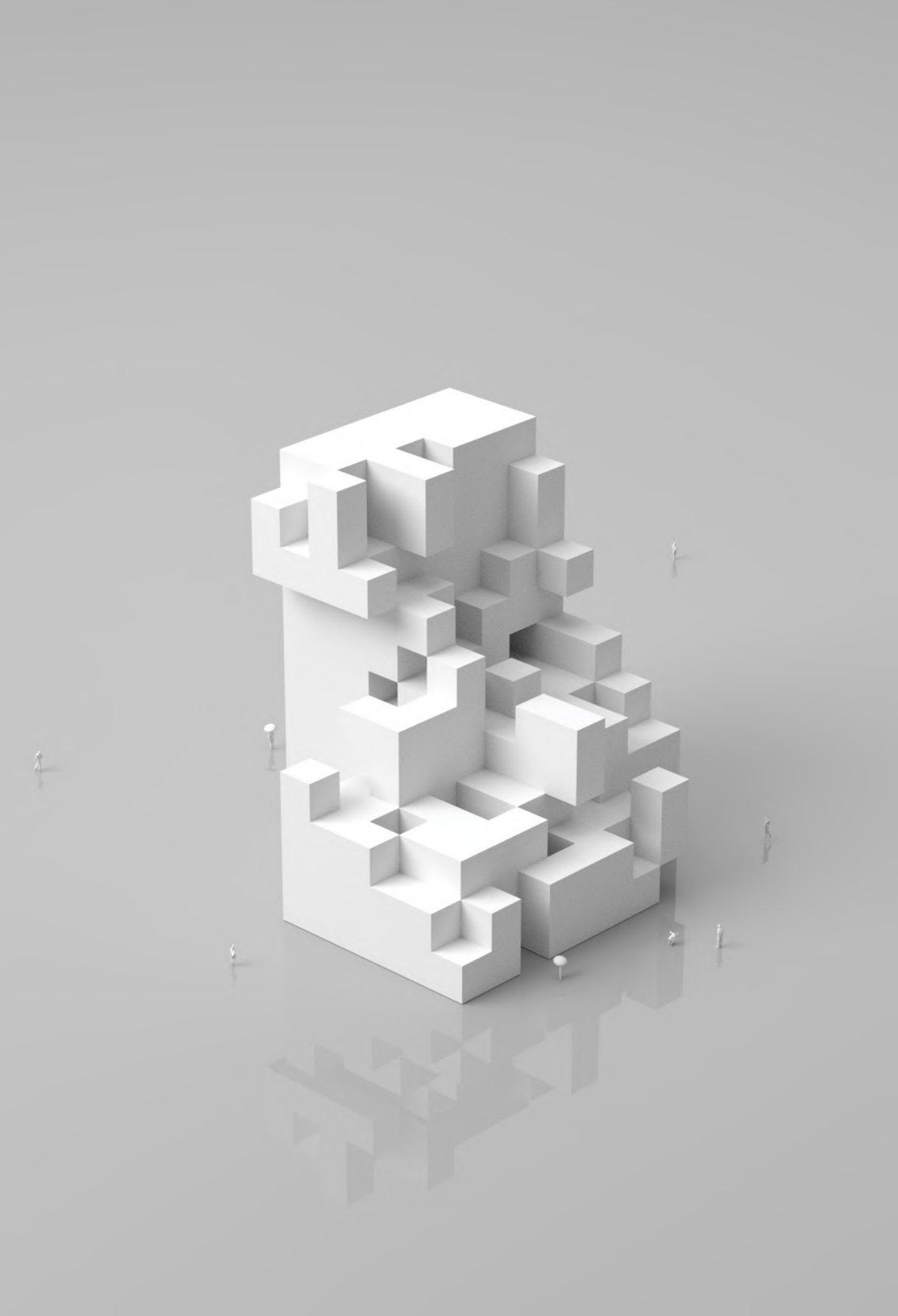
Total reward **2510**
Sun hours **6.0h**
Target **7h**











04 Risultati

4.1 Key Performance Indicators

Concluso il processo di validazione del modello, lo step successivo da compiere è quello di testarne la stabilità in contesti diversificati. A tale proposito verranno condotti dei set di test, identificati da variazioni di contesto riguardanti:

- Lo sviluppo del voxel space.
- Il posizionamento di ostacoli fisici e zone inaccessibili.
- Il target di ore di luce diurna media.
- La grandezza degli assemblaggi (in termini di numero di unità da posizionare).

I test verranno condotti con il fine di estrapolare da essi una serie di dati da analizzare per elaborare considerazioni sui comportamenti del modello, evidenziandone ricorrenze, limiti e punti di forza.

I diversi assemblaggi costruiti in questa fase verranno analizzati definendo un set di Key Performance Indicators (KPI) capace di estrapolare dati su specifiche caratteristiche riguardanti le qualità climatiche, di forma e di connettività di ogni assemblaggio.
I KPI forniscono un quadro di metriche misurabili che consentono di valutare il progresso verso gli obiettivi prefissati e l'efficacia delle azioni intraprese dal modello. Il loro utilizzo è determinante per identificare le aree di forza, di debolezza e le ricorrenze che le variazioni di contesto producono su qualità architettoniche oggettive e quantificabili negli assemblaggi.
I KPI definiti in questa ricerca si suddividono in 3 categorie: *Climate KPI, Connectivity KPI, Shape KPI.*

Climate KPI

I KPI climatici sono racchiusi in tre indicatori, che, coerentemente allo sviluppo del resto della ricerca, sono attinenti alla distribuzione dei dati sul numero di ore di esposizione alla luce solare dell'assemblaggio.

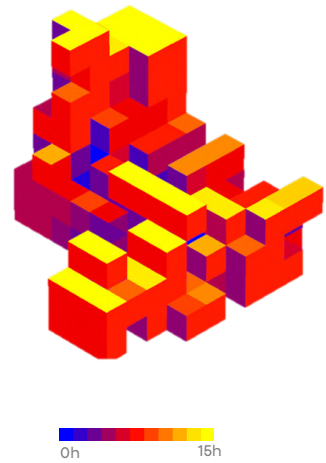
I tre indicatori *Sunlght hours, Standard Deviation e Goal Gap* sono progettati per monitorare l'adattabilità di ogni assemblaggio alle condizioni climatiche di contesto. Da questo punto di vista il modello ha garantito un buon tasso di successo con la fascia di illuminazione target fissa a 7h: l'obiettivo a questo punto è quello di osservare la sua risposta alla variazione della stessa.

A Sunlight hours

Le *Sunlight Hours* (SLH) indicano il numero di ore di luce diurna che illumina l'assemblaggio all'interno di una giornata.

Questo valore è ottenuto come media dei singoli valori di ore di illuminazione computati sulle superfici di involucro esterno di ogni unità che costituisce l'assemblaggio.

Il valore delle SLH può oscillare in un range che varia tra le 0 h e le 15 h.



$$SLH = \frac{\sum x_i}{N}$$

x_i = numero di ore di illuminazione diretta computate sulla singola superficie dell'assemblaggio

N = numero totale di superfici componenti l'assemblaggio

B Standard deviation

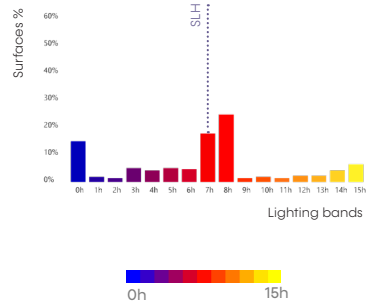
Preso un campione di dati la deviazione standard (σ) indica la variabilità degli stessi attorno al valor medio.

Nel nostro caso è fondamentale per capire il grado di dispersione dei dati acquisiti sull'illuminazione di ogni superficie che compone l'assemblaggio attorno alle SLH.

Valori di σ ridotti indicano una maggior percentuale di superfici illuminate per un numero di ore prossimo a quello target.

$$\sigma = \sqrt{\frac{\sum_{i=1}^{N} (x_i - SLH)^2}{N}}$$

Sun hours access distribution



x_i = numero di ore di illuminazione diretta computate sulla singola superficie dell'assemblaggio

N = numero totale di superfici componenti l'assemblaggio

C Goal Gap

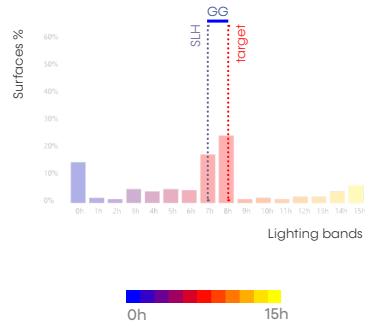
Il *Goal Gap* (GG) indica lo scarto tra il valore target di ore di accesso all'esposizione solare e le SLH.

Ha la funzione di mappare lo scostamento tra obiettivi e risultati in termini climatici al fine di descrivere in maniera dettagliata quanto il modello si sia avvicinato/allontanato dal target impostato.

Il GG è un indicatore di performance in quanto in relazione agli obiettivi di ricerca, suoi valori sotto la soglia delle 0.8h sono associati a test in cui il modello porta con succesoo a compimento il suo compito.

GG = SLH - target

Sun hours access distribution



target = numero di ore di illuminazione diretta medio di progetto.

Connectivity KPI

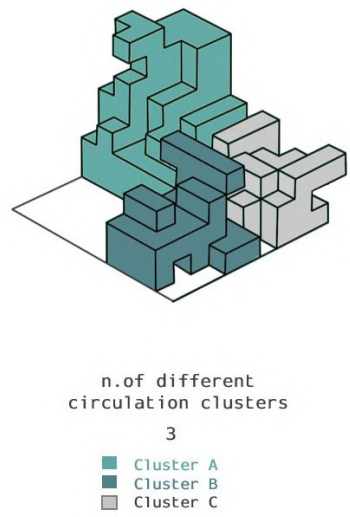
Per quanto riguarda l'analisi della connettività dei sistemi spaziali, vengono utilizzati due KPI in grado di riassumere sinteticamente informazioni legate alla circolazione orizzontale e verticale all'interno dell'assemblaggio.

A Circulation Clusters

Circulation Clusters (CC) indica il numero di cluster isolati e reciprocamente inaccessibili che si creano all'interno di un assemblaggio.

Ai fini della ricerca viene interpretato positivamente un numero di cluster di circolazione più basso possibile. In particolare un numero di CC pari ad 1 indica un assemblaggio totalmente connesso internamente. Nell'analisi dei successivi test un numero di CC minore o uguale a 3 sarà associato ad assemblaggi un grado di connessione accettabile per considerare centrato l'obiettivo legato alla connettività.

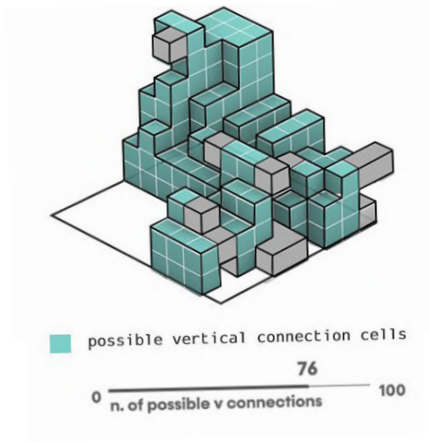
Il numero di CC viene computato utilizzando il componente "Circulation path " in Grasshopper.



B Number of possible vertical connections

Il numero di connessioni verticali (NPVC) possibili indica quante unità sono topologicamente compatibili ad ospitare un potenziale collegamento verticale.

I blocchi computati in questo insieme sono quelli che vedono presente un blocco adiacente nella cella a loro sottostante o sovrastante.



Shape KPI

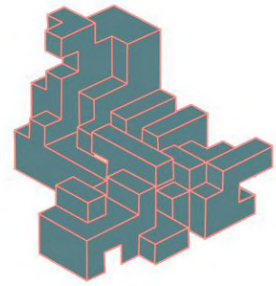
Gli Shape KPI misurarono le caratteristiche architettoniche legate alla forma dei sistemi spaziali ottenuti. Sono dati legati alla densità (*Density index)*, allo sviluppo verticale (*n. of floors*) e all'impronta dell'assemblaggio (*Surrounding index*).

A Density index

Il *Density index (*DI) è un indice legato alla densità dell'assemblaggio.

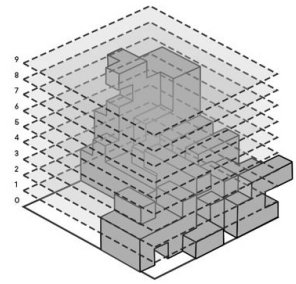
Essendo calcolato come rapporto tra il volume totale e superficie di involucro del sistema, alla suo aumentare corrisponde una diminuzione della densità dell'assemblaggio.

$$DI = \frac{\text{Assembly volume}}{\text{Assembly casing surface}}$$



B N. of floors

N. of floors (NOF) indica il numero di livelli di cui si compone l'assemblaggio. È un dato importante da associare al DI in quanto offre una rappresentazione immediata di quale sia lo sviluppo predominante in termini di direzionalità del sistema.

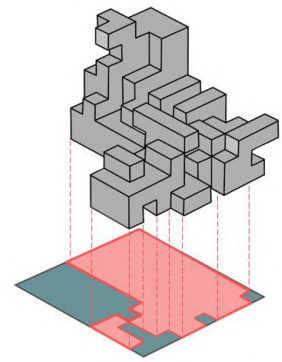


C Surrounding index

Il *Surrounding index* (SI) è un indice che racchiude le proporzioni dimensionali tra l'impronta dell'assemblaggio e il relativo spazio circostante disponibile a livello 0. Viene ottenuto a partire dal rapporto tra impronta e area totale utilizzabile al livello 0.

$$SI = 1 - x$$

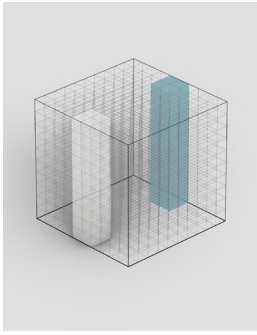
$$x = \frac{\text{footprint}}{\text{Total area}}$$



4.2 Test

Environment settings & obstacles

Definiti i KPI, gli esperimenti condotti al fine di estrapolare informazioni architettoniche riguardanti l'operato dei modelli, sono stati compiuti su environment caratterizzati da impostazioni di progetto diverse tra loro. In primo luogo differenziando i target di ore di illuminazione diurna e dopodiché inserendo impedimenti o complicazioni all'interno della matrice tridimensionale di base.
A tal fine sono stati introdotti due differenti tipi di *obstacles*:



- Il primo si basa sul posizionamento di inaccessible cells (celle inaccessibili), ossia aree dello spazio voxelizzato, assimilabili a zone protette da confini o limiti normativi, in cui l'agent non può "costruire" l'assemblaggio.

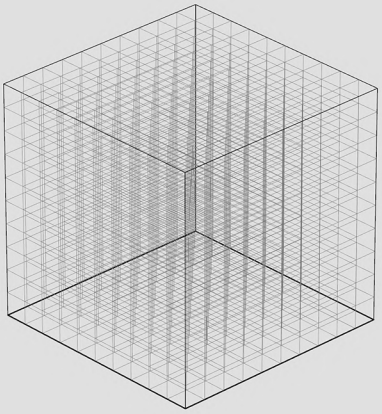
- La seconda tipologia di ostacolo è di natura fisica, dunque oltre a non consentire il posizionamento di blocchi nell'area di collocamento, hanno la funzione di ostruire il passaggio della radiazione solare.

Per i vari set di test verranno estrapolati i dati relativi ai KPI precedentemente descritti per essere poi confrontati globalmente. Per un confronto ottimale degli assemblaggi ottenuti nelle diverse condizioni, si è deciso di mantenere costanti i fattori riguardanti le dimensioni del voxel space (fissato a 1000 celle (10 x 10 x 10)) e il numero di parti da posizionare a 100 unità.

Test set 0

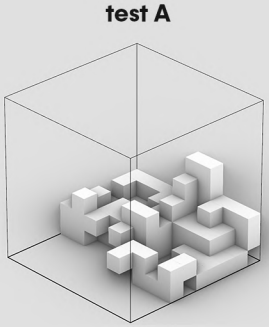
Environment settings

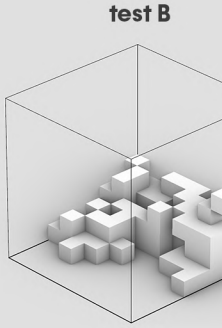
0 Voxel space & obstacles
1000 cells (10x10x10)

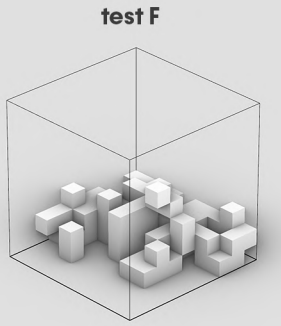


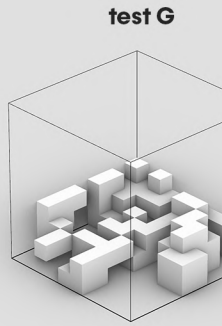
1 Sun hours target
7h

1 Assembly n. of blocks
100



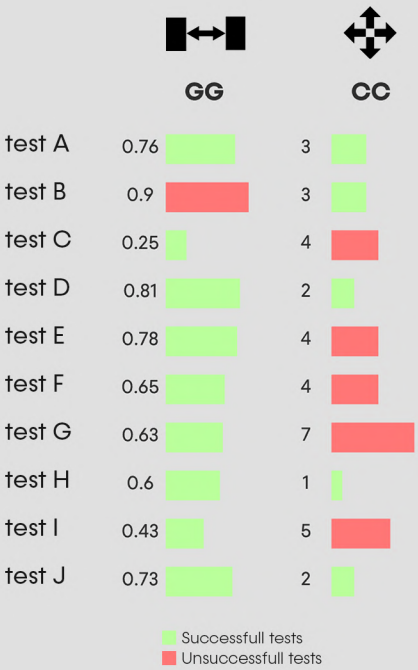






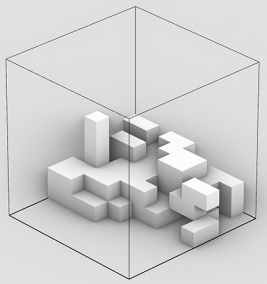
Questo primo set di test è stato condotto su un environment privo di ostacoli con il fine di studiare il comportamento del modello in condizioni di assenza di vincoli.

Model performance KPI

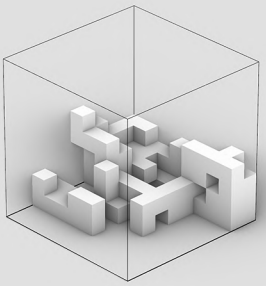


Tested assemblies

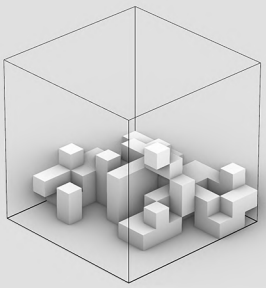
test C



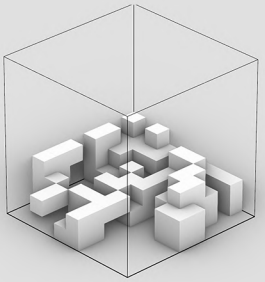
test D



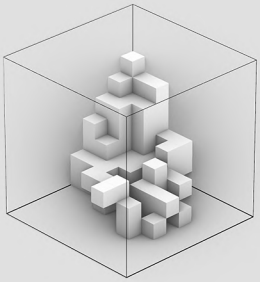
test E



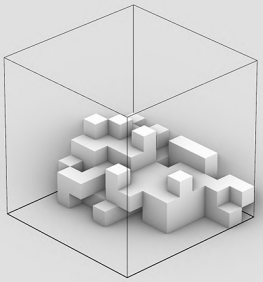
test H



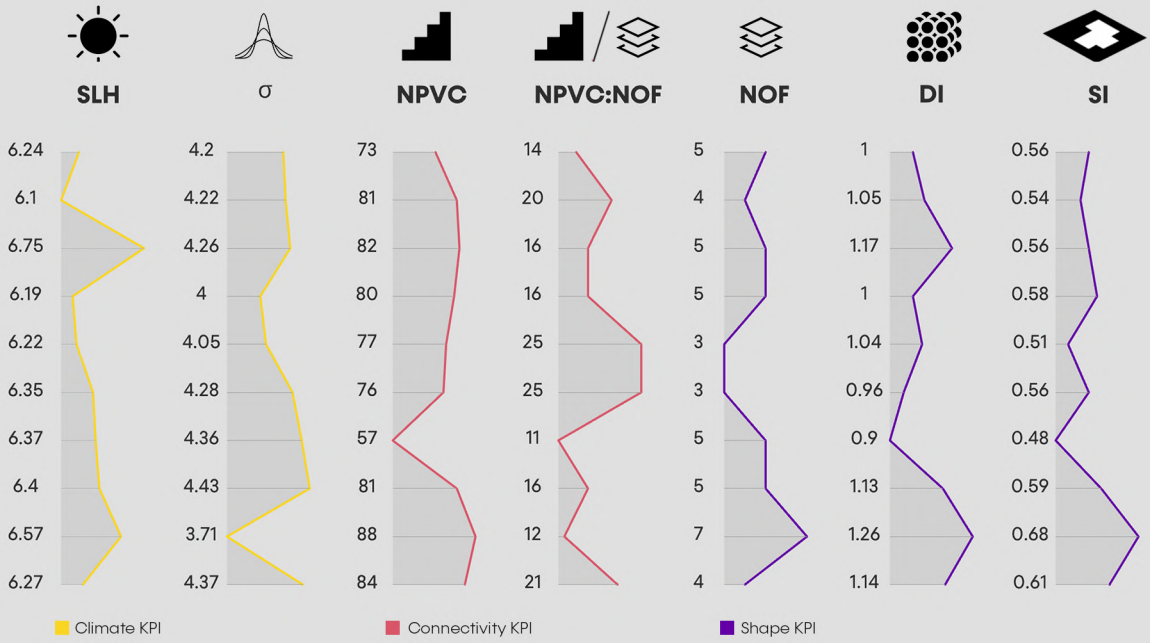
test I



test J



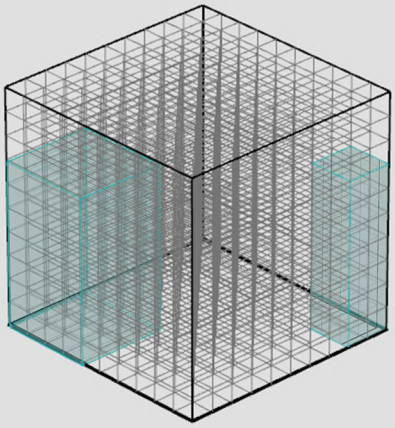
Other KPI



Test set 1

Environment settings

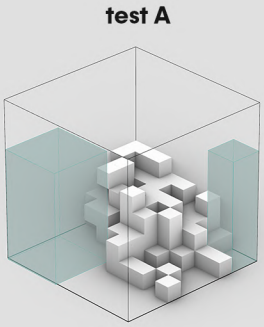
0 Voxel space & obstacles
1000 cells (10x10x10)

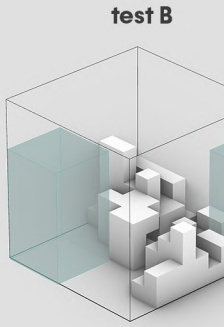


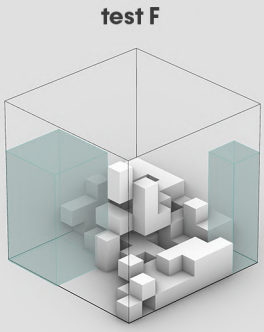
1 Sun hours target
7h

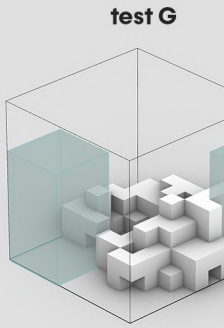
1 Assembly n. of blocks
100

Unaccesible cells



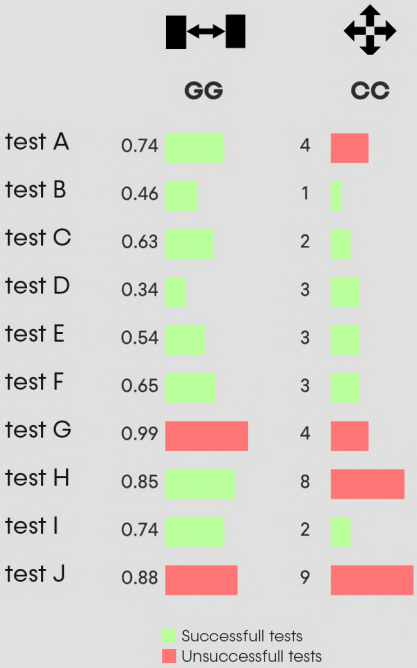






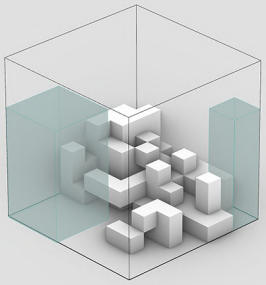
Il test set 1 presenta due gruppi di celle inaccessibili posizionati in maniera da ridurre del 15% lo spazio utilizzabile per generare gli assemblaggi.

Model performance KPI

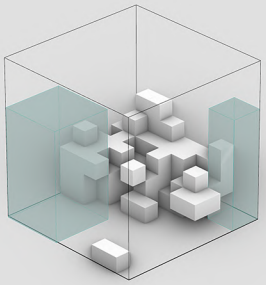


Tested assemblies

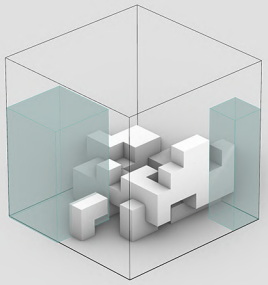
test C



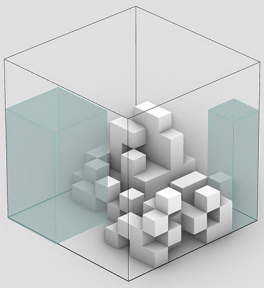
test D



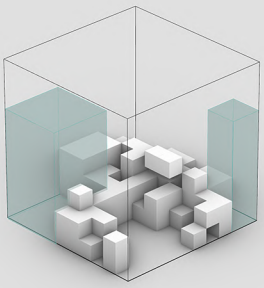
test E



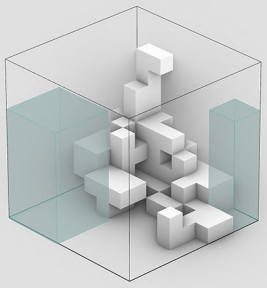
test H



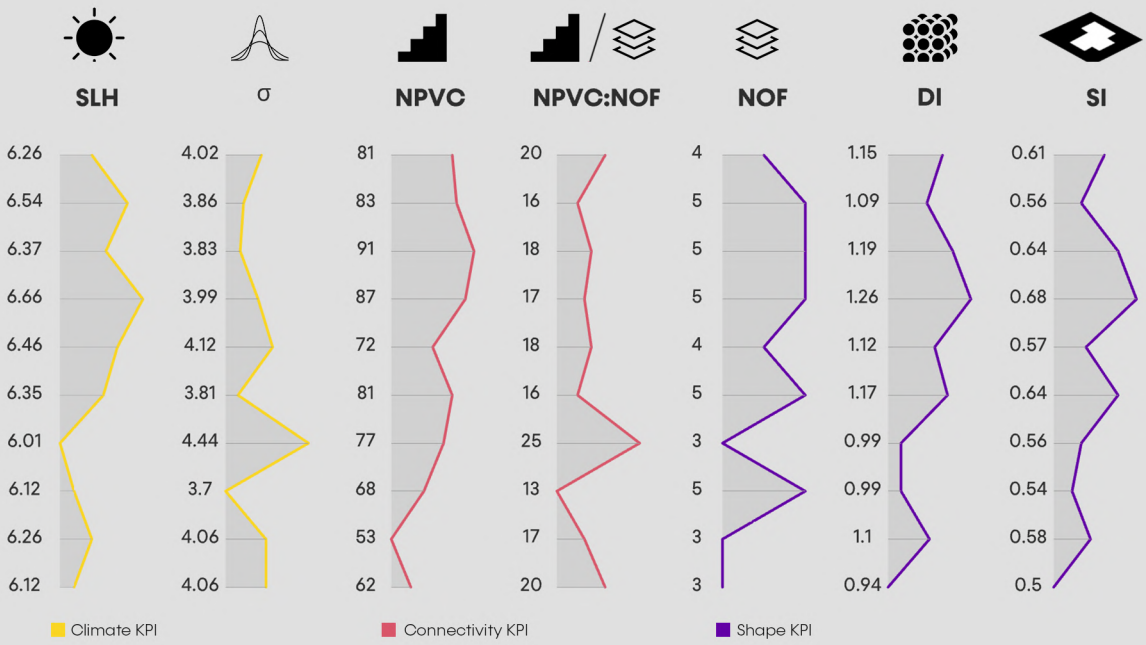
test I



test J



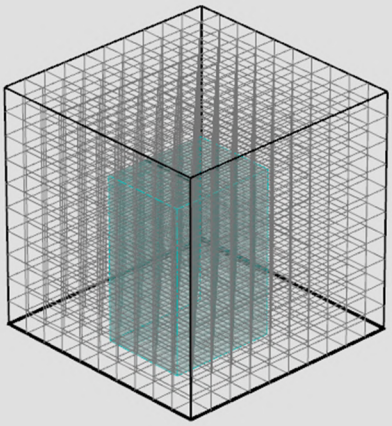
Other KPI



Test set 2

Environment settings

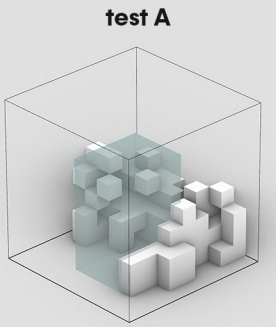
0 Voxel space & obstacles
1000 cells (10x10x10)

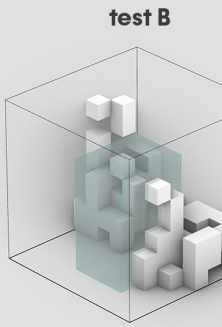


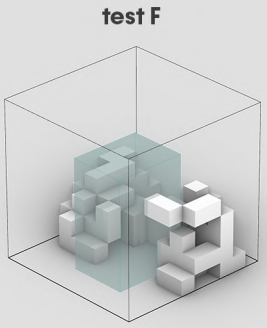
1 Sun hours target
7h

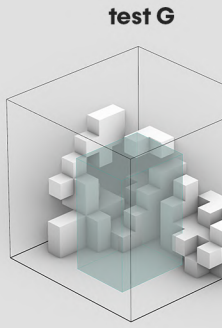
1 Assembly n. of blocks
100

Unaccesible cells



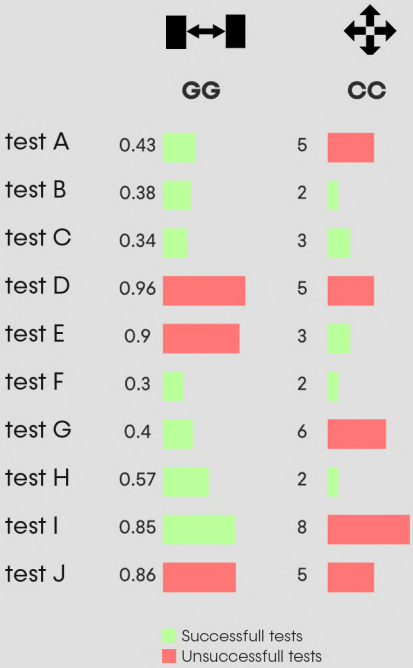






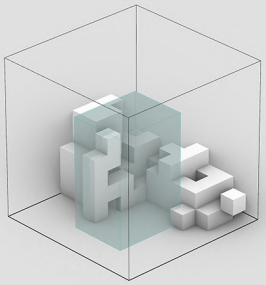
Questo test set vede posizionato un volume centrale di celle inaccessibili. L'idea è quella di studiare il posizionamento dei blocchi in termini di connettività quando viene imposto un vincolo forte è impattante sullo spazio circostante.

Model performance KPI

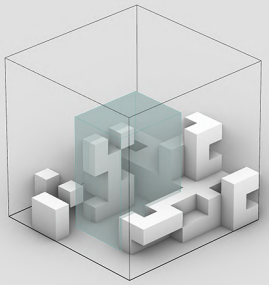


Tested assemblies

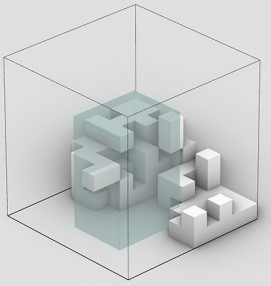
test C



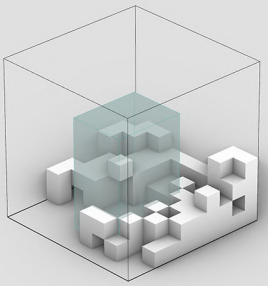
test D



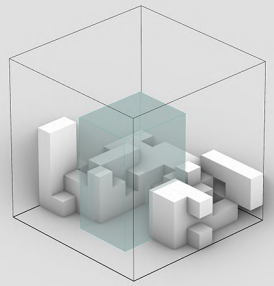
test E



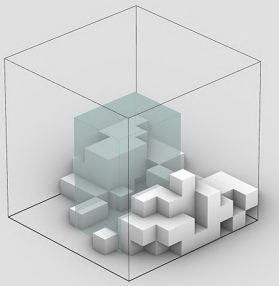
test H



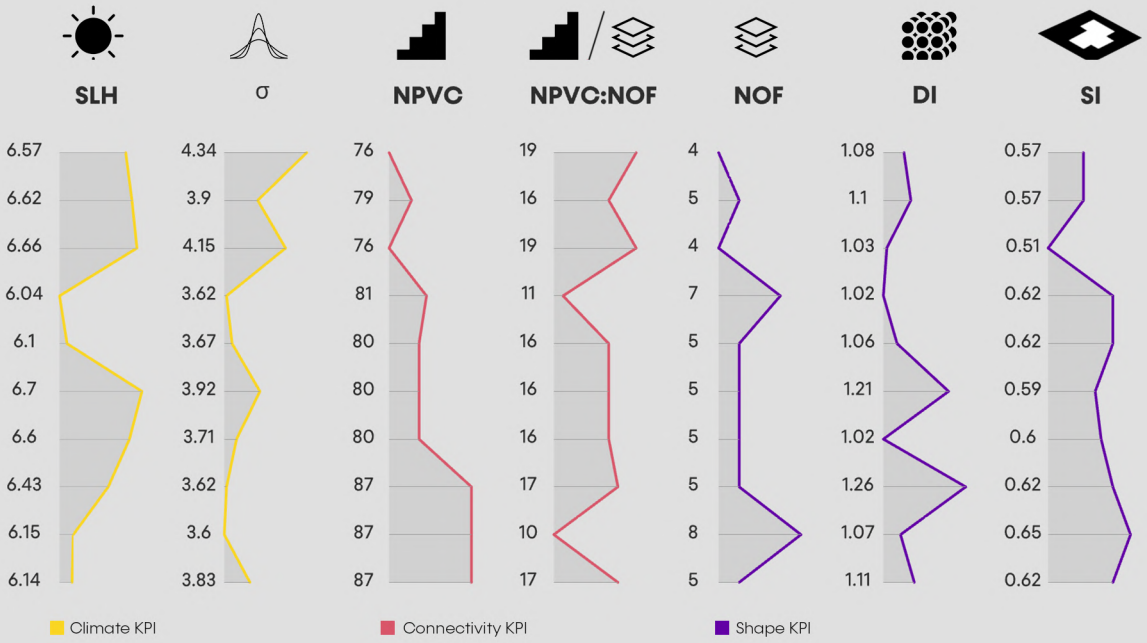
test I



test J



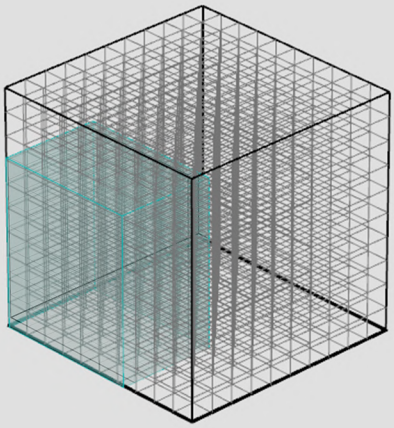
Other KPI



Test set 3

Environment settings

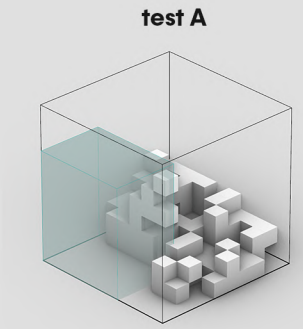
0 Voxel space & obstacles
1000 cells (10x10x10)

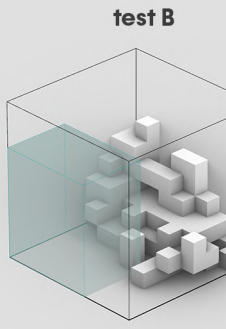


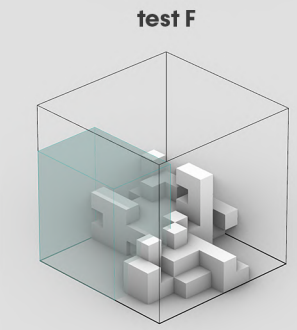
1 Sun hours target
7h

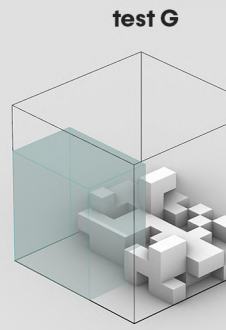
1 Assembly n. of blocks
100

Unaccesible cells



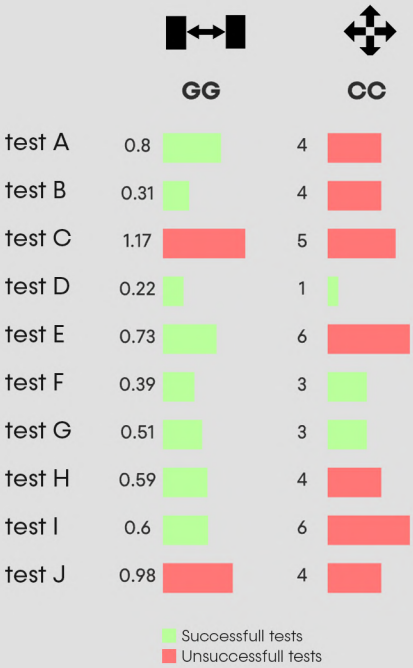






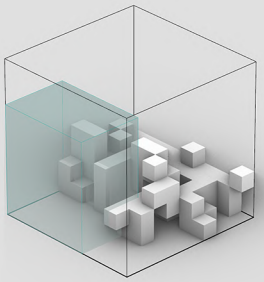
Il test set 3 presenta un ostacolo di tipo "normativo" che riduce lo spazio utilizzabile del 20% conferendoli una particolare forma ad "L".

Model performance KPI

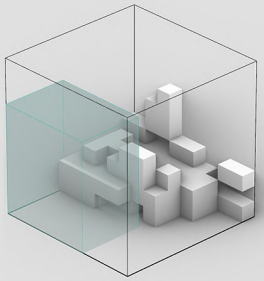


Tested assemblies

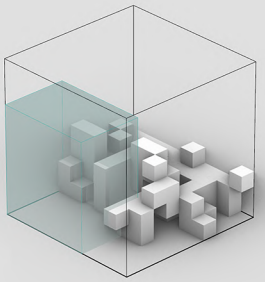
test C



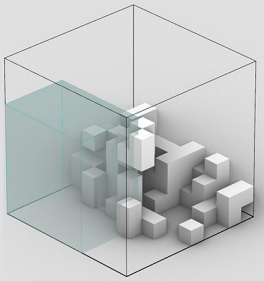
test D



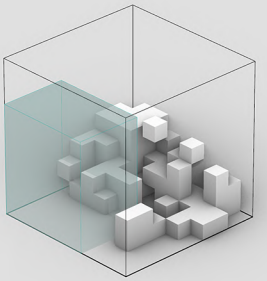
test E



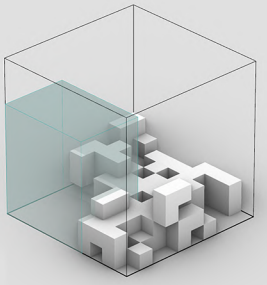
test H



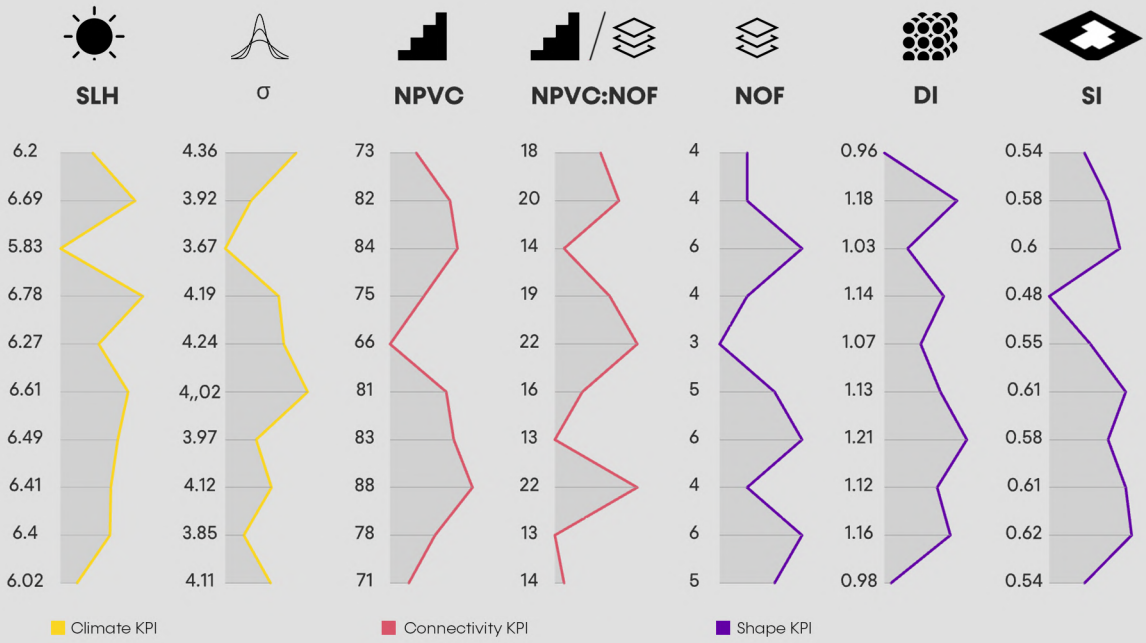
test I



test J



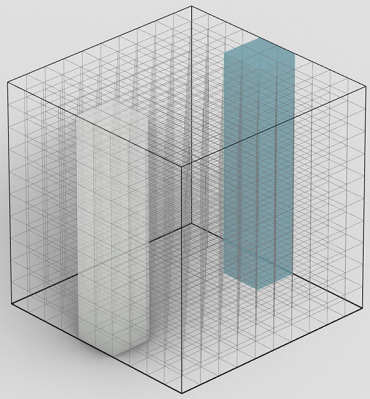
Other KPI



Test set 4

Environment settings

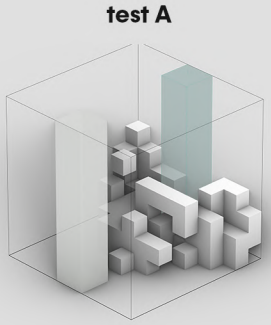
0 Voxel space & obstacles
1000 cells (10x10x10)

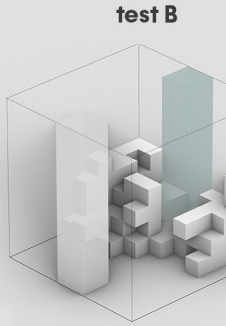


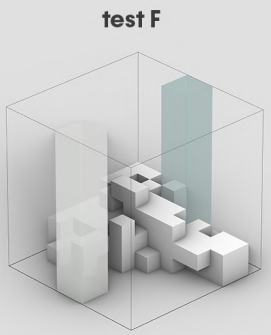
1 Sun hours target
5h

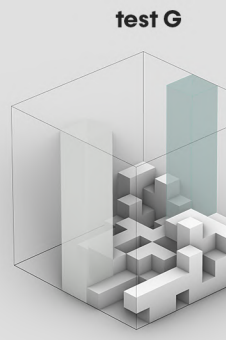
1 Assembly n. of blocks
100

Obstacles
Unaccesible cells



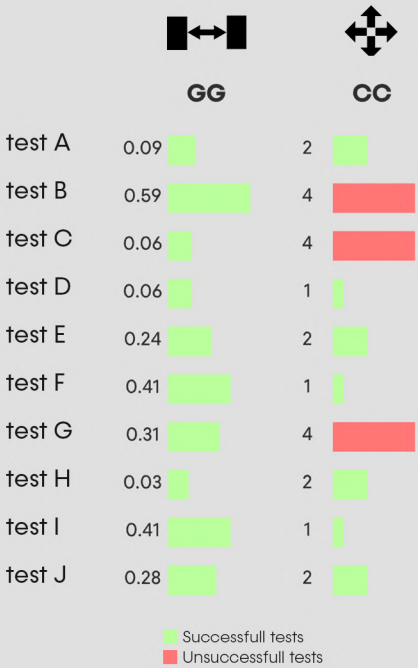




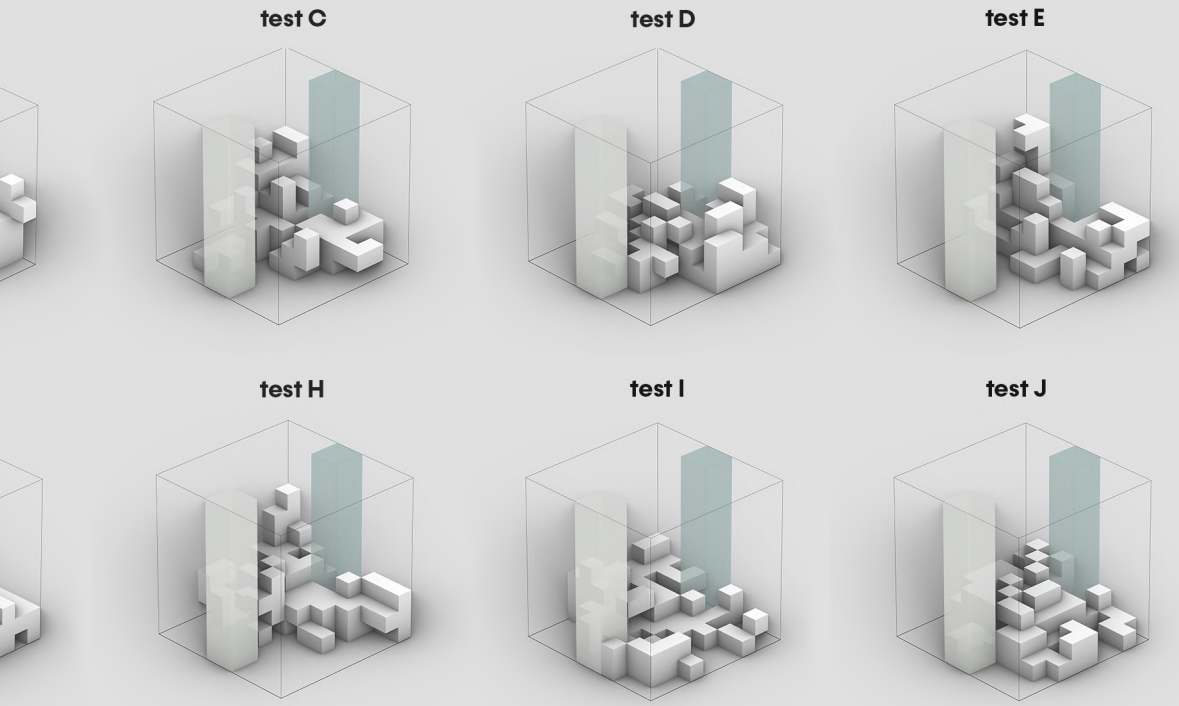


In questo test set è stato introdotto un ostacolo fisico che ha il compito di creare zone d'ombra ostruendo la luce solare. L'obiettivo qui è quello di testare la stabilità del modello con target di illuminazione differenti, in questo caso 5h.

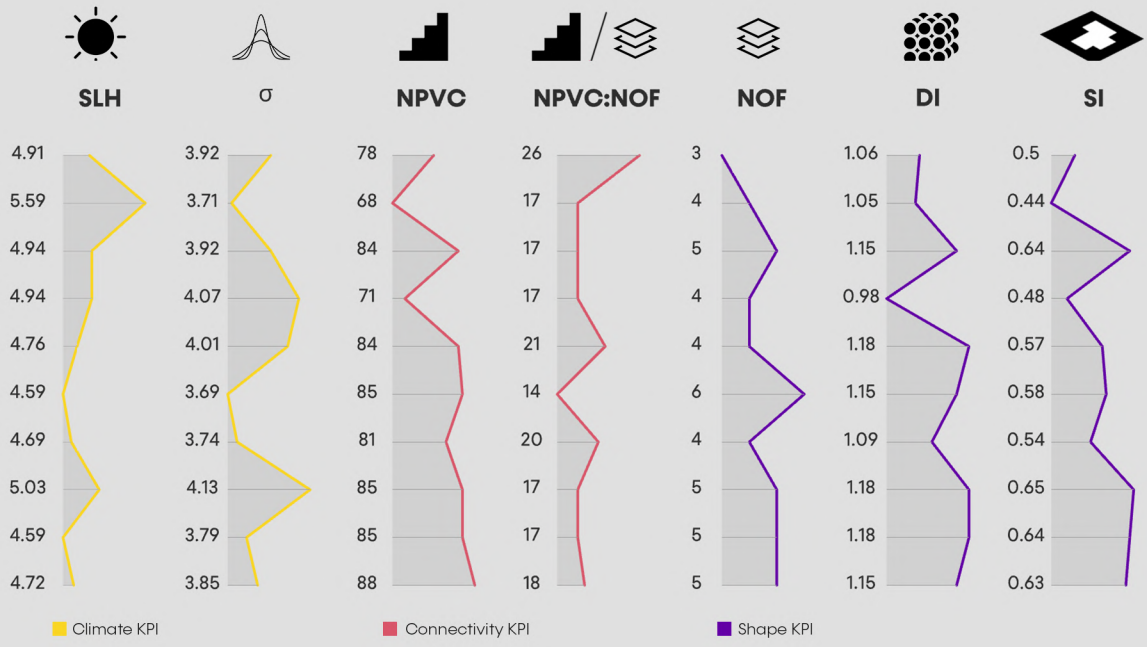
Model performance KPI



Tested assemblies



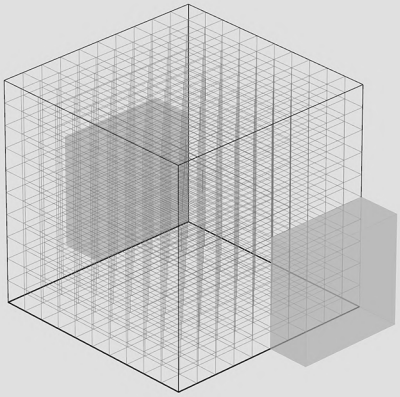
Other KPI



Test set 5

Environment settings

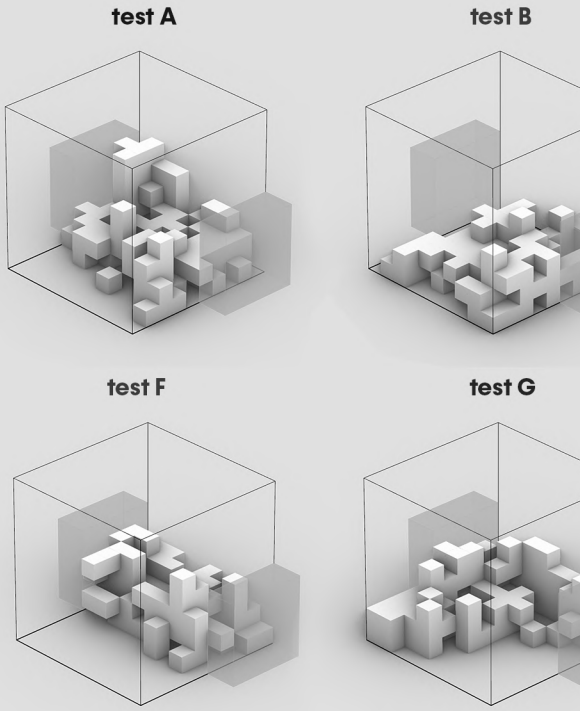
0 Voxel space & obstacles
1000 cells (10x10x10)



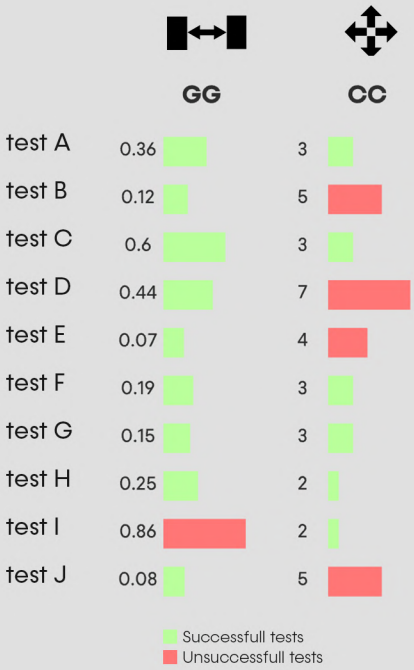
1 Sun hours target
4h

1 Assembly n. of blocks
100

Obstacles

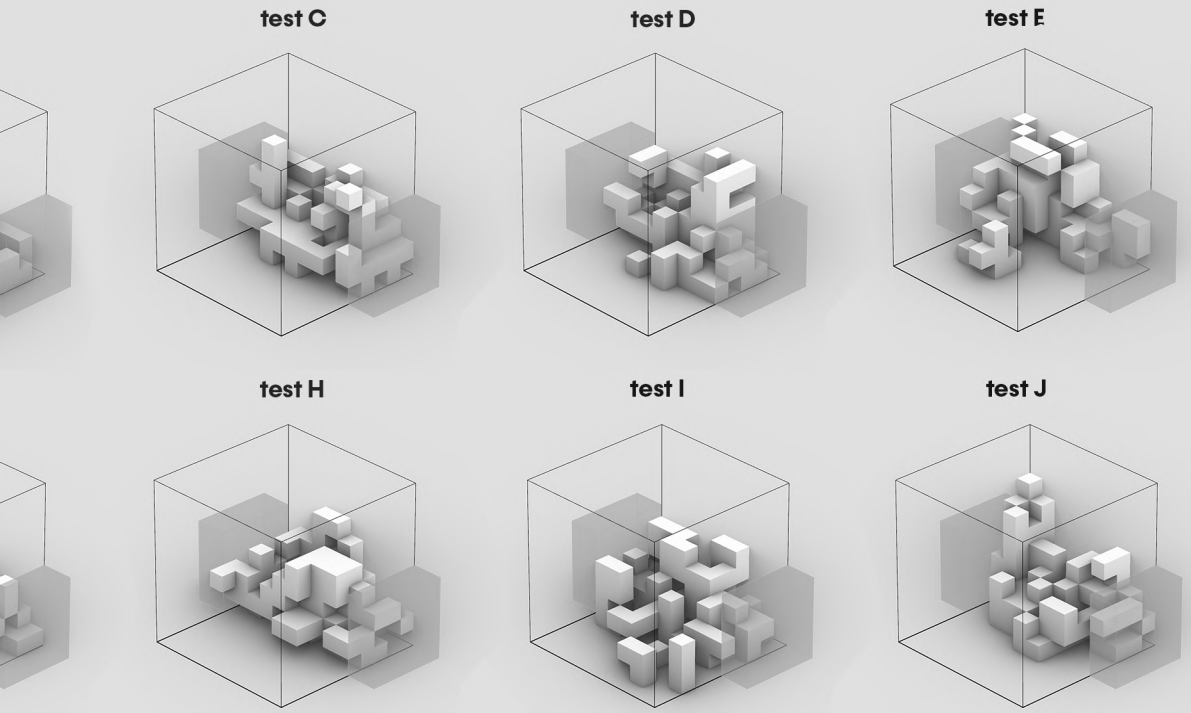


Model performance KPI

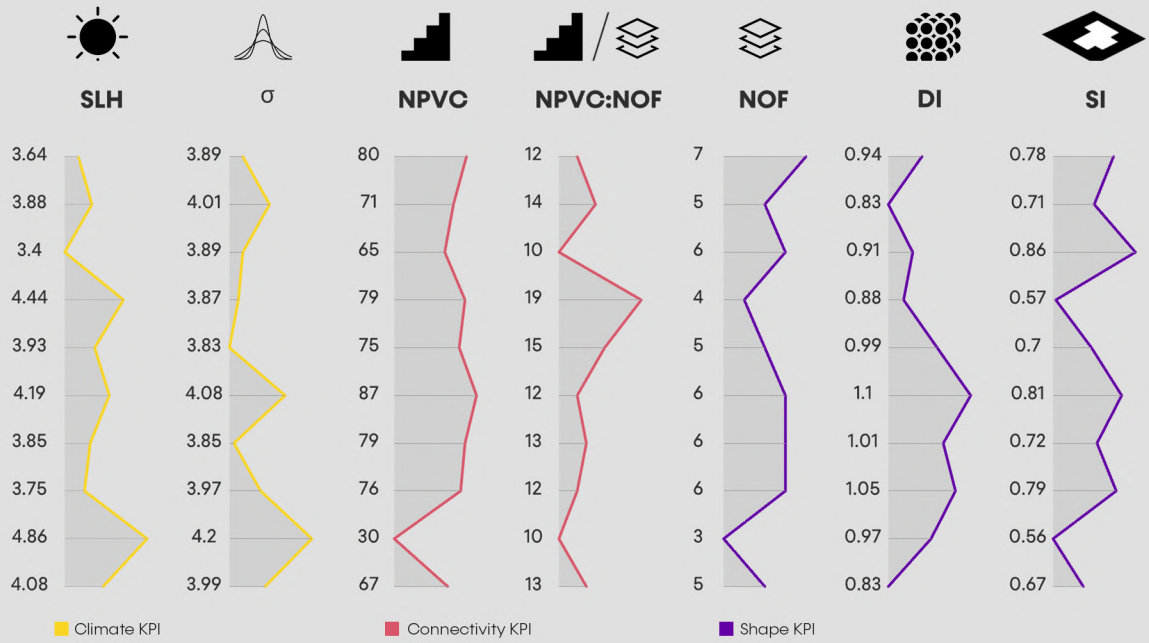


In questo caso sono stati utilizzati due ostacoli fisici posti all'esterno della matrice tridimensionale. Sono caratterizzati da una lunghezza pari alla metà dell'estensione dell'environment. L'idea è quella di capire se il modello riesce ad utilizzare lo spazio d'ombra definito tra i due ostacoli per sottostare alle condizioni esterne di illuminazione impostate a 4h.

Tested assemblies



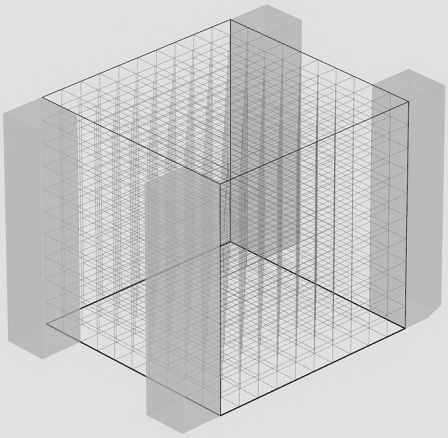
Other KPI



Test set 6

Environment settings

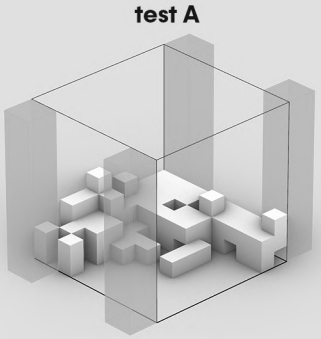
0 Voxel space & obstacles
1000 cells (10x10x10)

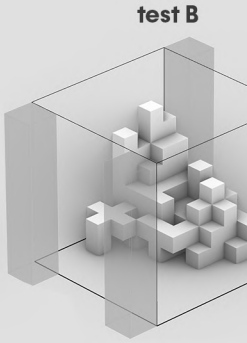


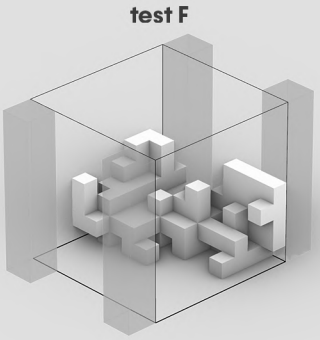
1 Sun hours target
5h

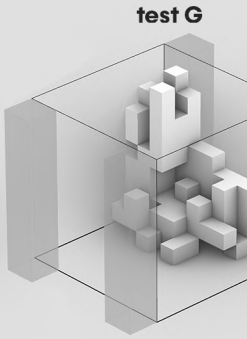
1 Assembly n. of blocks
100

Obstacles



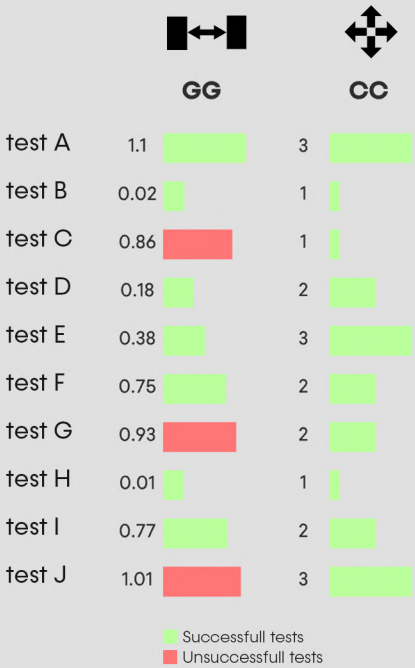




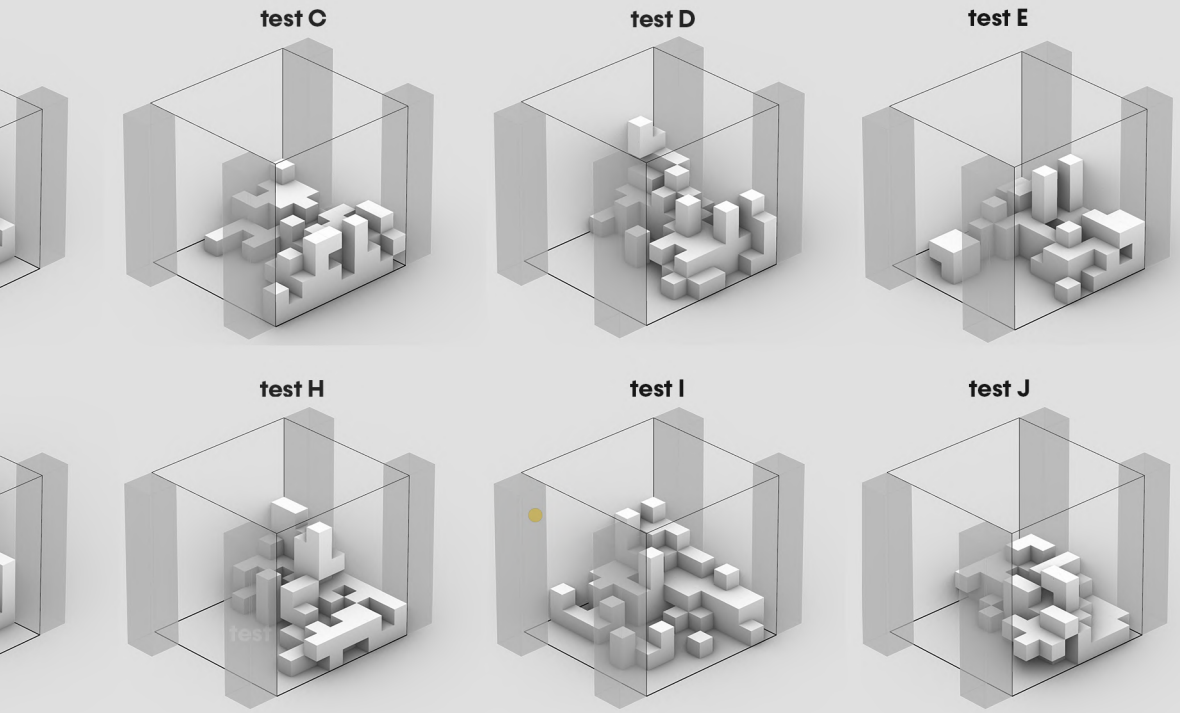


Quest'ultimo test set è una variante del precedente in cui sono stati posti 4 ostacoli simmetrici e che comportano un grosso vincolo in termini di ombreggiamento dell'assemblaggio.

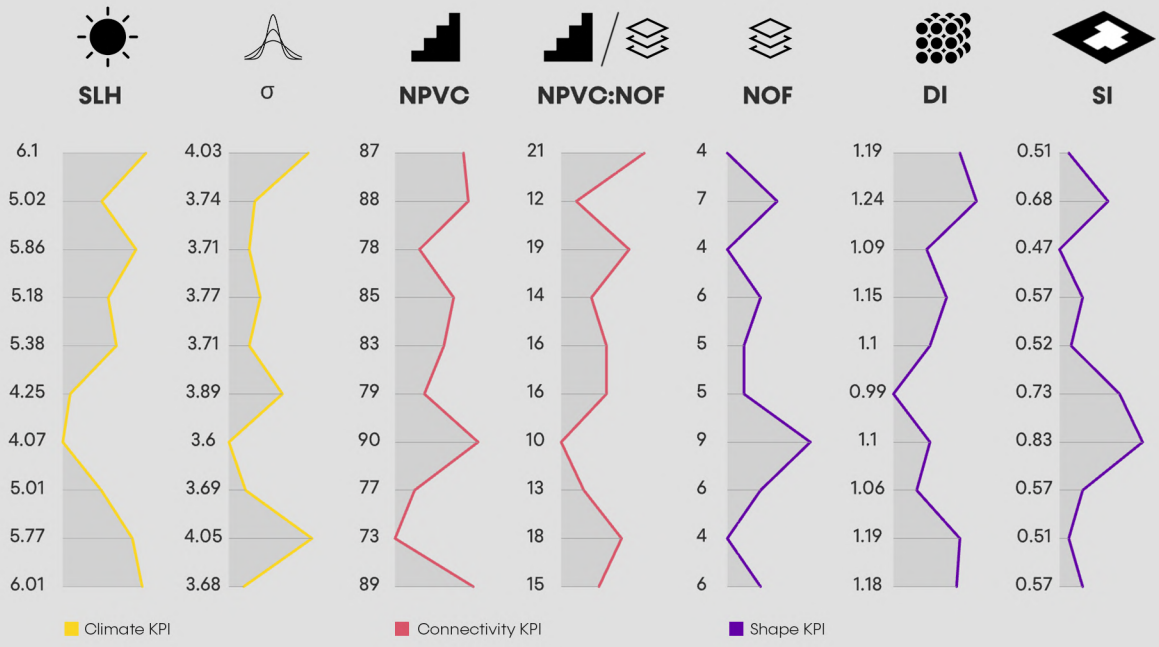
Model performance KPI

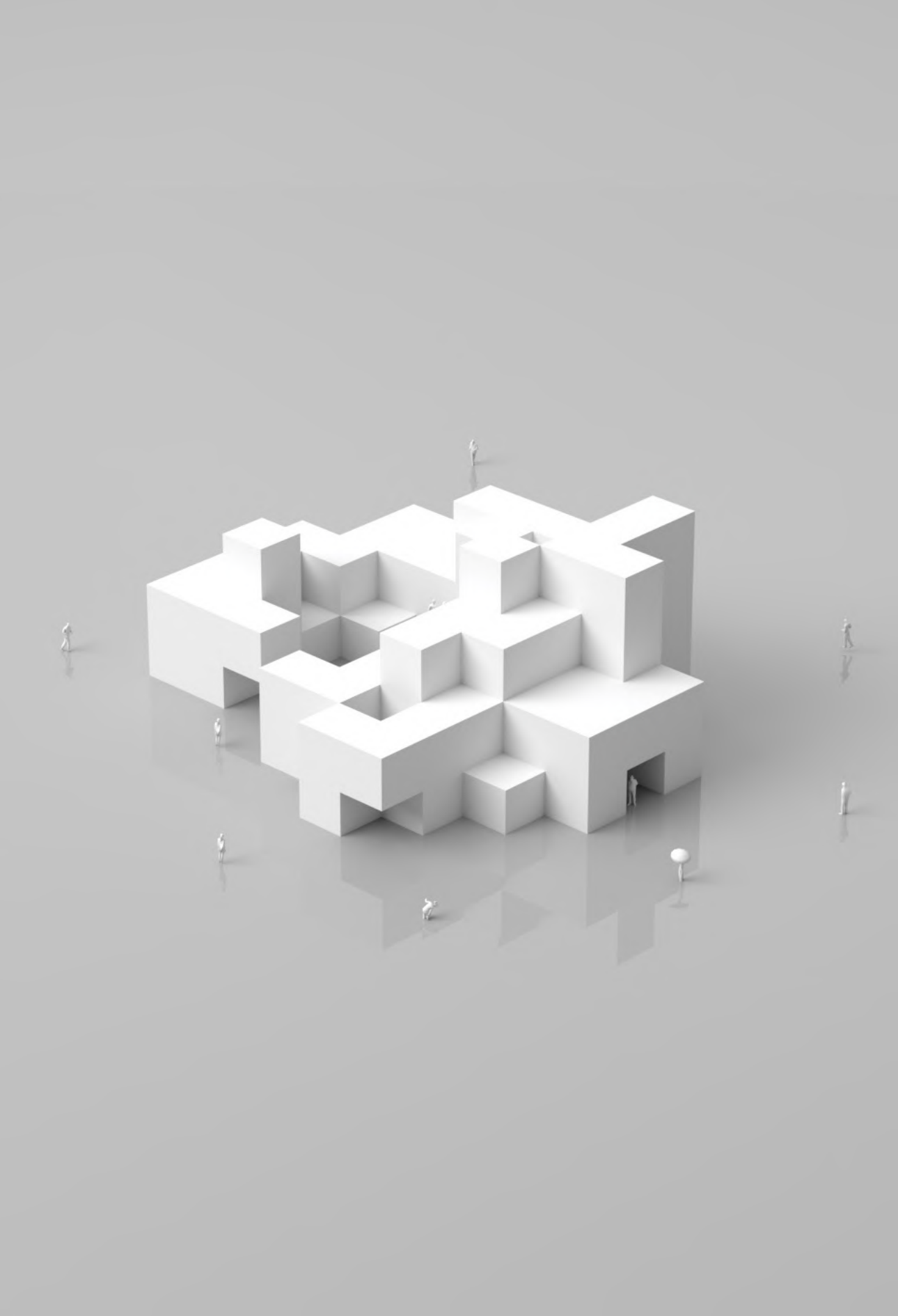


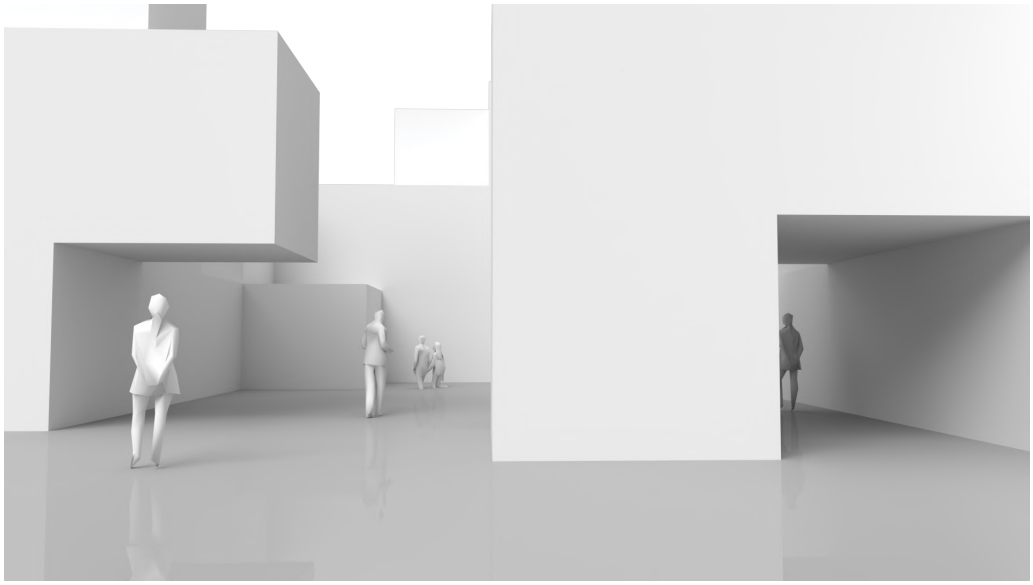
Tested assemblies

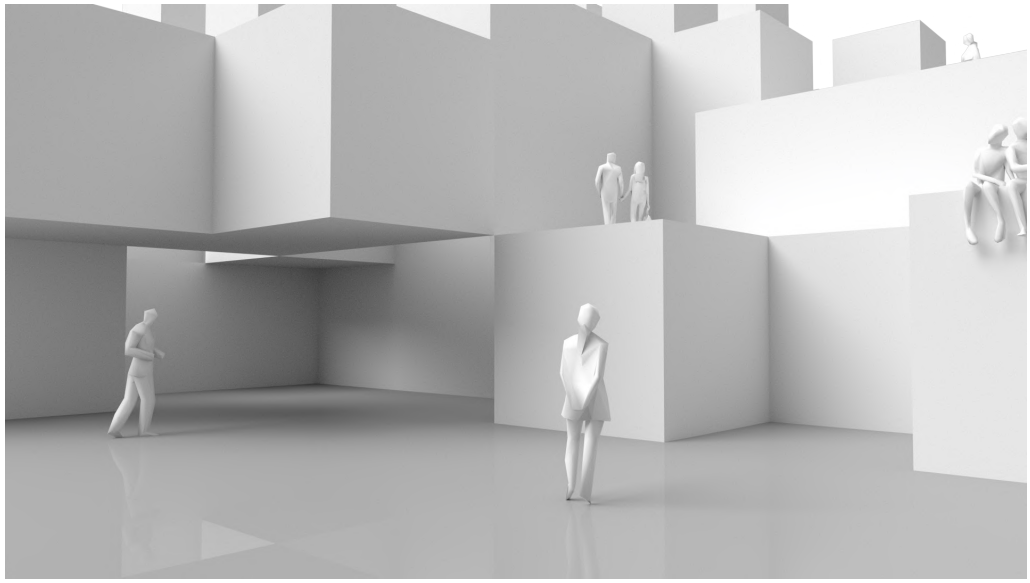


Other KPI









Data analysis

Ottenuti i diversi assemblaggi ed estratti i relativi KPI nei diversi scenari ipotizzati, il prossimo step è quello di confrontarli in maniera da poter mettere in risalto quali sono stati i limiti del modello, i suoi punti di forza, e quali sono le caratteristiche ricorrenti degli assemblaggi dal punto di vista architettonico.

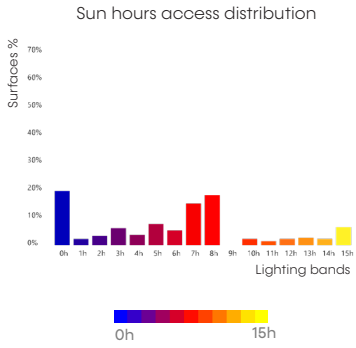
I valori dei KPI verranno rappresentati a tal fine all'interno di heatmap, utili a mostrare l'intensità e la distribuzione degli stessi su una scala colori.

Climate KPI

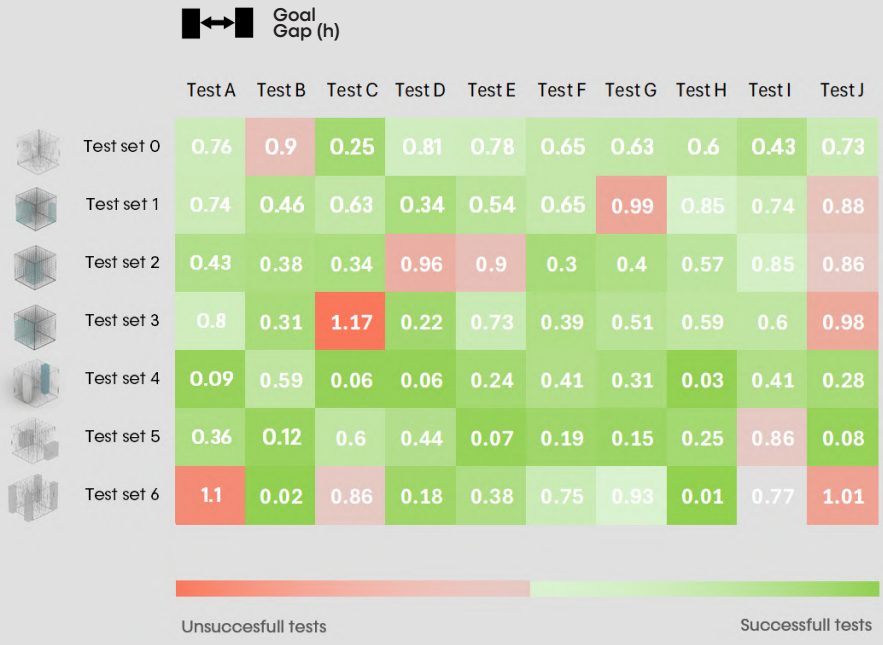
L'andamento sui *Climate KPI* evidenza un importante grado di successo da parte del modello nell'aderire alle condizioni climatiche imposte.

Nell'82,4% dei casi infatti gli assemblaggi risultano esposti alla luce per un numero di ore compatibile con quello target (in un range del 10%).
Analizzando contemporaneamente i grafici sulle Sunlight hours e sul Goal Gap (**_f44**), si nota come le criticità evidenti nei test set 2, 3 e 6, siano in realtà accompagante da uno scarto dal target il cui massimo supera di poco l'ora di luce, a testimonianza di un modello molto performante.
Punto di forza del modello è sicuramente l'adattabilità a condizone climatiche differenti: il test set 4 e 5, aventi target di illuminazione diversi da quello di training, presentano un tasso di successo rispettivamente del 100% e del 90%.
Per quanto riguarda l'analisi della distribuzione dei dati sull'illuminazione delle singole superfici facenti parte dell'assemblaggio, si è deciso di accompagnare allo studio della deviazione standard (**_f45**), l'analisi visiva del grafico sun hour access distribution.

Nel caso massimo (Test set 1 – test G **_f43**) si osserva una deviazione standard di 4,44 con una slh distribution che, salvo un picco di circa il 20% di superfici non illuminate, si dispone in maniera sufficentemente non dispersa attorno al valor medio.

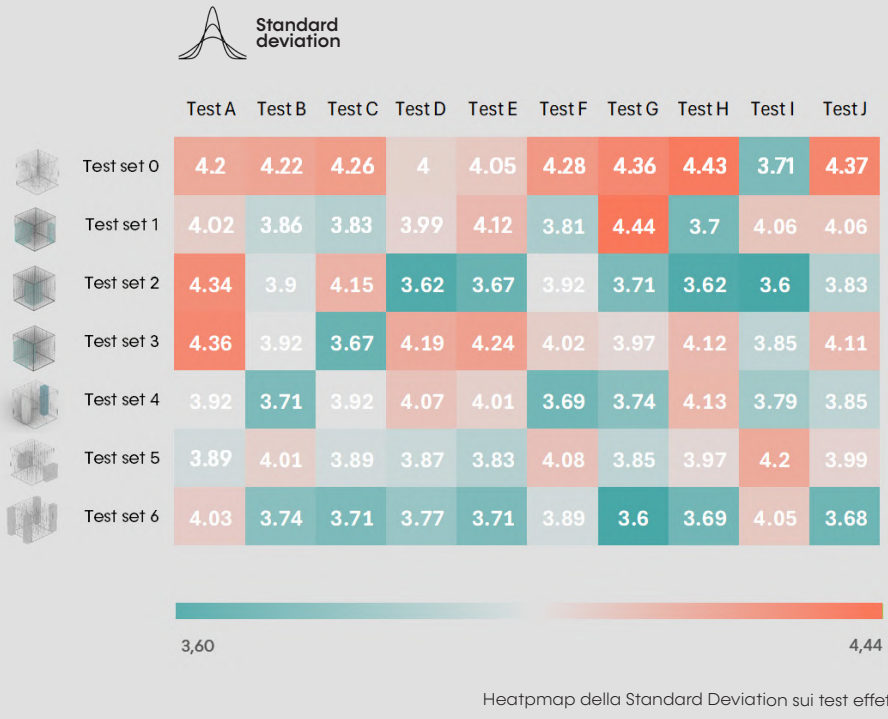


_f43
Distribuzione dell'accesso alla luce solare delle superfici dell'assemblaggio relativo al test G - Test set 1



_f44

Heatpmap del Goal Gap sui test effettuati



_f45

Heatpmap della Standard Deviation sui test effettuati

Connectivity KPI

I valori dei *Connectivity KPI* mostrano la difficoltà del modello a creare assemblaggi che siano sempre totalmente connessi internamente.

Se dal punto di vista della circolazione verticale il valore minimo di NPVC (**_f47**) è pari a 53, con la conseguenza che almeno il 53% di celle sono compatibili ad ospitare un collegamento verticale, dal punto di vista dei *Circulation Clusters* la situazione è leggermente diversa. Il 61% degli assemblaggi ha un grado di connessione sufficiente (CC inferiore a 3) mentre nel resto dei casi le condizioni di contesto si sono dimostrate limitative per il modello.

Non a caso gli assemblaggi più critici da questo punto di vista sono quelli costruiti in environment caratterizzati dalla presenza di celle inaccessibili (test set 1,2,3) che si sono dimostrate problematiche per la creazione di sistemi spaziali totalmente connessi.

Ciò evidenzia un leggero punto di debolezza del modello che può essere affrontato in sviluppi futuri introducendo gli ostacoli anche nella fase di training.

Circultation Clusters

	Test A	Test B	Test C	Test D	Test E	Test F	Test G	Test H	Test I	Test J
Test set 0	3	3	4	2	4	4	2	1	5	2
Test set 1	4	1	2	3	3	3	4	8	2	9
Test set 2	5	2	3	5	3	2	6	2	8	5
Test set 3	4	4	5	1	6	3	3	4	6	4
Test set 4	2	4	4	1	2	1	4	2	1	2
Test set 5	3	5	3	7	4	3	3	2	2	5
Test set 6	3	1	1	2	3	2	2	1	2	3

1 — 9

_f46
Heatpmap del numero di Cluster di circolazione sui test effettuati

N. of possible vertical connections

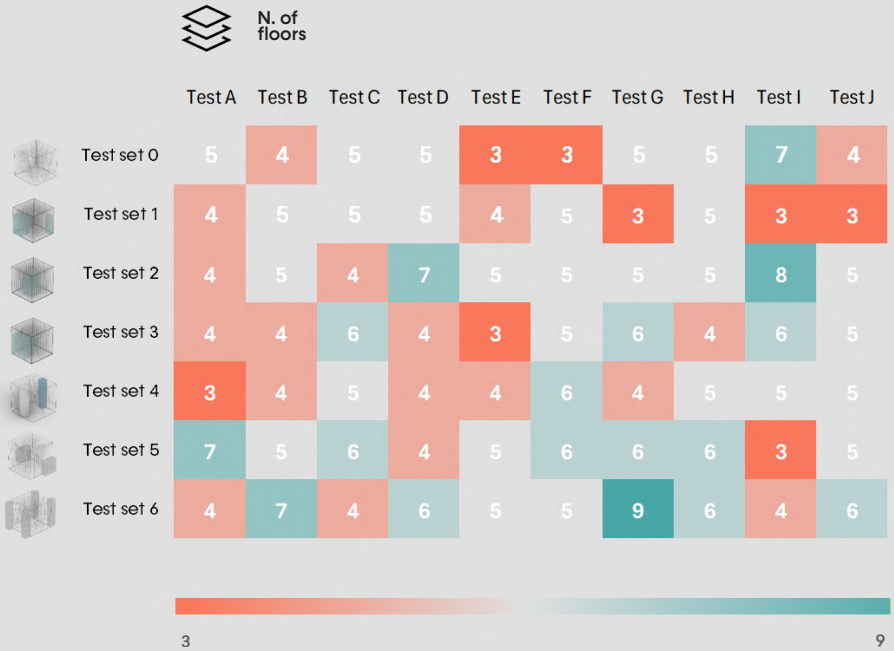
	Test A	Test B	Test C	Test D	Test E	Test F	Test G	Test H	Test I	Test J
Test set 0	73	81	82	80	77	76	57	81	88	84
Test set 1	81	83	91	87	72	81	77	68	53	62
Test set 2	76	79	76	81	80	80	80	87	87	87
Test set 3	73	82	84	75	66	81	83	88	78	71
Test set 4	78	68	84	71	84	85	81	85	85	88
Test set 5	80	71	65	79	75	87	79	76	75	67
Test set 6	87	88	78	85	83	79	90	77	73	89

53 — 91

_f47
Heatpmap del numero di potenziali connessioni verticali all'interno degli assemblaggi ottenuti

Shape KPI

I dati relativi agli *Shape KPI* evidenziano le ripercussioni che i diversi ostacoli hanno provocato sull'assetto spaziale degli assemblaggi.
L'andamento del *Density index* (_f49) mostra come situazioni con maggiore densità si verificano in corrispondenza del test set 5.
L'aumento di densità in questa circostanza è conseguenza della ricerca del modello di "costruire" all'interno di una zona d'ombra per sottostare al target di illuminazione, in questo caso di 4h.
Densità minore si ha paradossalmente nel test set 1, in cui seppur lo spazio utilizzabile è stato ridotto del 15% dalla presenza degli ostacoli, gli assemblaggi sono stati sviluppati in maniera meno densa.
Per chiarire meglio il significato del DI di ogni sistema spaziale, si può prestare attenzione ai casi di massimo (Test set 1 – test D) e minimo (Test set 5 – test J). Alla differenza tra i valori di DI rispettivamente di 1,26 e 0,83 non corrisponde una variazione di densità visivamente eclatante.
Il DI ha però un legame con il *Surrounding index* (_f50)e con *Number of floors* (_f48). Analizzando questi dati si nota come il numero di livelli cresca con l'aumentare della densità (diminuzione del DI). Assemblaggi meno densi sono invece riconducibili ad un maggior consumo di suolo a livello zero. Il *Surrounding Index* massimo a 0,86 corrisponde infatti ad una situazione di densità vicina al minimo (DI = 0,91).



_f48
Heatpmap del numero di livellii su cui si sviluppano gli assemblaggi ottenuti

Density Index

	Test A	Test B	Test C	Test D	Test E	Test F	Test G	Test H	Test I	Test J
Test set 0	1	1.05	1.17	1	1.04	0.96	0.9	1.13	1.26	1.14
Test set 1	1.15	1.09	1.19	1.26	1.12	1.17	0.99	0.99	1.1	0.94
Test set 2	1.08	1.1	1.03	1.02	1.06	1.21	1.02	1.26	1.07	1.11
Test set 3	0.96	1.18	1.03	1.14	1.07	1.13	1.21	1.12	1.16	0.98
Test set 4	1.06	1.05	1.15	0.98	1.18	1.15	1.09	1.18	1.18	1.15
Test set 5	0.94	0.83	0.91	0.88	0.99	1.1	1.01	1.05	0.97	0.83
Test set 6	1.19	1.24	1.09	1.15	1.1	0.99	1.1	1.06	1.19	1.18

0,83 → Decreasing density → 1,26

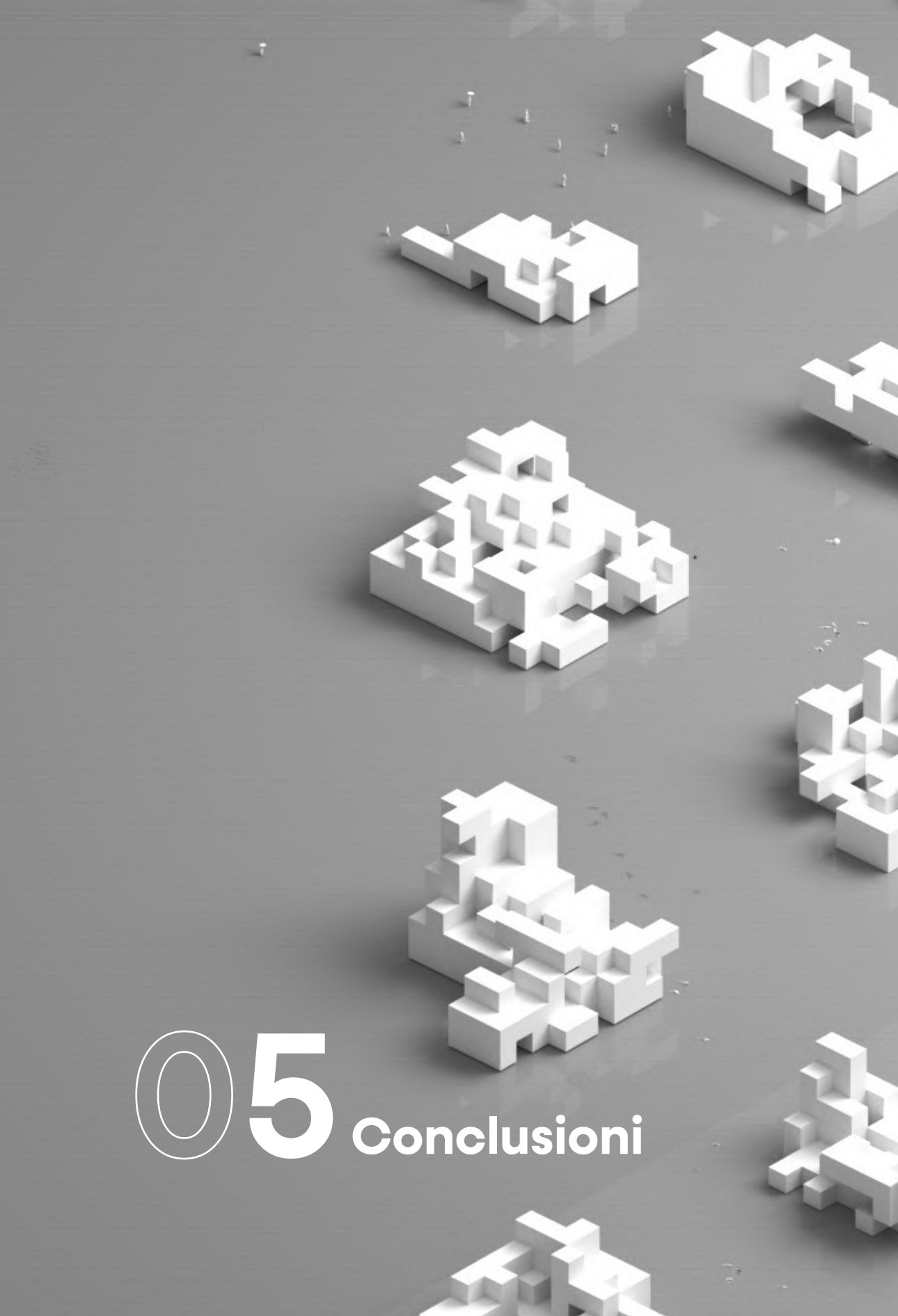
_f49
Heatpmap dell'indice di densità relativo ai test effettuati

Surrounding Index

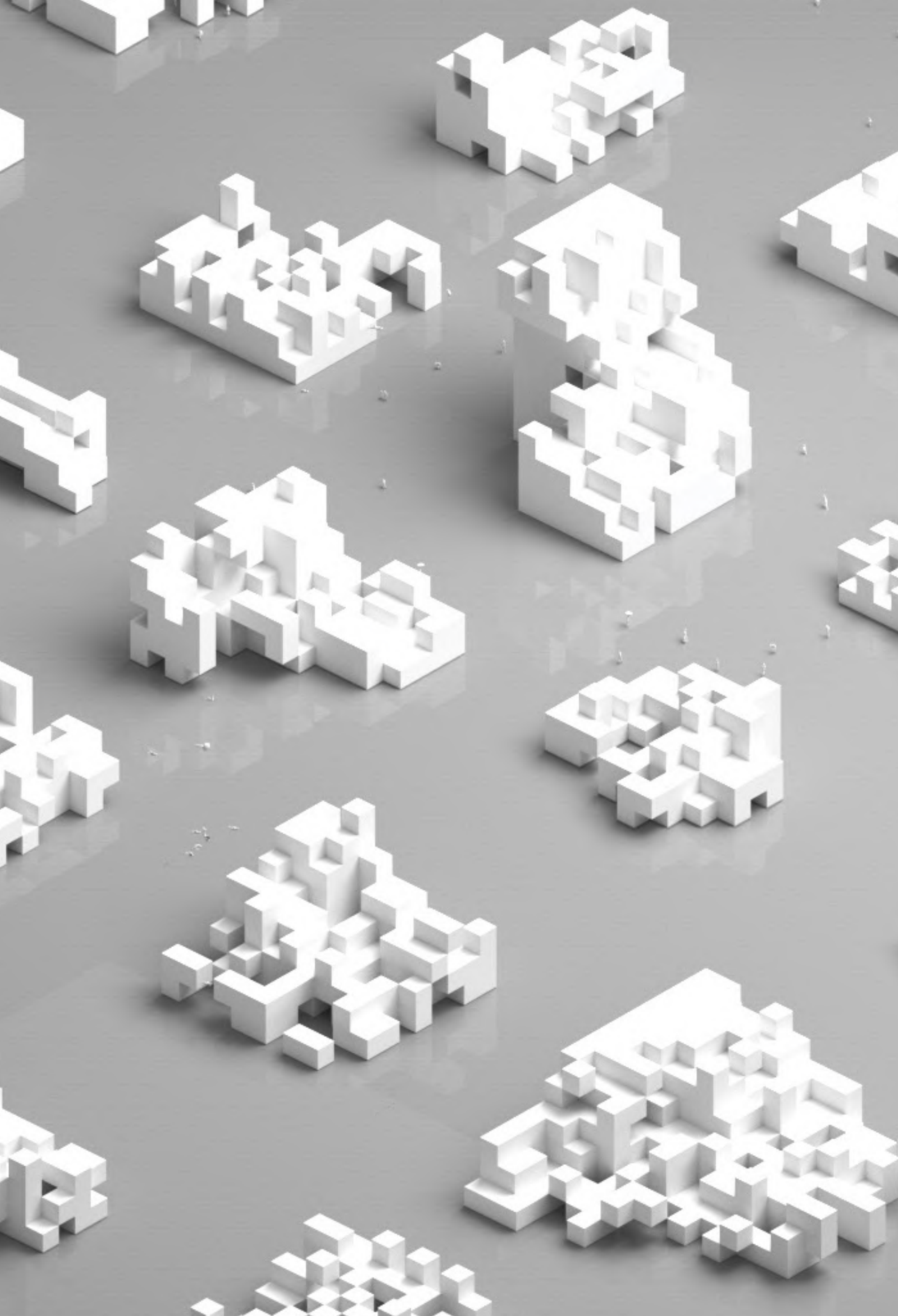
	Test A	Test B	Test C	Test D	Test E	Test F	Test G	Test H	Test I	Test J
Test set 0	0.56	0.54	0.56	0.58	0.51	0.56	0.48	0.59	0.68	0.61
Test set 1	0.61	0.56	0.64	0.68	0.57	0.64	0.56	0.54	0.58	0.5
Test set 2	0.57	0.57	0.51	0.62	0.62	0.59	0.6	0.62	0.65	0.62
Test set 3	0.54	0.58	0.6	0.48	0.55	0.61	0.58	0.61	0.62	0.54
Test set 4	0.5	0.44	0.64	0.48	0.57	0.58	0.54	0.65	0.64	0.63
Test set 5	0.78	0.71	0.86	0.57	0.7	0.81	0.72	0.79	0.56	0.67
Test set 6	0.51	0.68	0.47	0.57	0.52	0.73	0.83	0.57	0.51	0.57

0,44 → Decreasing footprint → 0,83

_f50
Heatpmap del Surrounding index legato all'impronta dei test effettuatii



05 Conclusioni



Conclusioni

ReLighting spaces offre un metodo per l'integrazione di feedback climatici all'interno del processo di assemblaggio di sistemi spaziali-architettonici di tipo combinatorio, mediante l'utilizzo del Reinforcement Learning.
La ricerca di algoritmi a cui affidare le logiche decisionali (selezione delle operazioni da svolgere), definito in un contesto di progettazione combinatoria è, parallelamente al design delle parti costituenti, fondamentale per favorire l'attualizzarsi delle capacità relazionali di quest'ultime permettere l'emergere di qualità architettoniche d'insieme.
Inglobare dati sul clima e l'ambiente come feedback all'interno del processo di generazione di un'architettura, significa creare sistemi adattivi, in grado di apprendere dal loro auto-generarsi informazioni che possono essere catalizzate per ottimizzare le qualità architettoniche, che ad essi sono associabili. Il processo combinatorio diventa così il risultato di una simbiosi tra spazio e clima che rievoca la nozione di architettura intelligente[19] (Steenson, 2017) e che soprattutto è aperto alle esigenze di progettazione specifiche di ogni contesto.
In questa ricerca questo tema è stato affrontato sfruttando le potenzialità del RL come framework per l'apprendimento di strategie decisionali, affidando ad agents che ricevono feedback ambientali e topologici da un'environment costruito per questo scopo, l'automazione delle scelte di aggregazione delle parti che definiscono assemblaggi spaziali a diverse scale di configurazione.

I risultati della ricerca fanno emergere considerazioni e questioni di carattere teorico, tecnico e metodologico.
In primo luogo, attraverso un processo progressivo di studio e testing dei modelli, la ricerca propone un approccio metodologico valido a trattare in termini computazionali aspetti quantitativi dello spazio che possono essere utilizzati a questo fine. Fermo restando che il quadro teorico introdotto nella tesi è parziale e limitato[20] rispetto ai temi introducibili in architettura, di carattere aperto e soggettivo, e per questo associati a qualità dello spazio inaffrontabili dal punto di vista computazionale, il numero di ore di esposizione a luce solare diretta e il numero di adiacenze per le nuove parti posizionate nell'assemblaggio sono stati dati funzionali a studiare in termini numerici il problema alla base di questa ricerca e produrre sistemi spaziali adattivi agli obiettivi climatici e di connettività specificati.
Il capitolo 3.3 offre una documentazione meticolosa di come il problema di RL associato agli obiettivi di ricerca sia stato affrontato focaliz-

/19
Si fa riferimento al capitolo 1.2 Architectural Intelligence

/20
Ci si riferisce al capitolo 3.3 in cui si parla della difficoltà nello studio di una funzione di reward associata problemi di tipo architettonico.

zandosi sulla progressiva raffinazione di *reward function* e *observation space* portando allo sviluppo del PPO model 0.11.
Le performance raggiunte da questo modello inquadrano un primo ottimo risultato in quanto si è riusciti a definire una corretta integrazione tra gli strumenti di Grasshopper, Python e LadyBug per creare un modello in grado di restituire assemblaggi spaziali sviluppati inglobando le condizioni climatiche di contesto in maniera positiva rispetto agli obiettivi della tesi.

In secondo luogo, il capitolo 3.4 mostra come si è riusciti ad estendere il problema di RL a contesti diversificati. Al termine del processo di analisi dei risultati ottenuti si può dire di essere giunti a concepire un modello in grado di utilizzare correttamente i feedback climatici dell'environment per poter costruire step dopo step, sistemi spaziali compatibili con le condizioni di illuminazione imposte. Ogni assemblaggio emerge coerentemente da un processo basato su una commistione tra informazioni locali di tipo topologico e informazioni globali riguardanti l'esposizione alla luce diurna. Recependo queste, il modello ingloba parte del clima all'interno di un processo di composizione scalabile ad environments di qualsiasi dimensione e connotati da condizioni di contesto di genere differente. Questo risultato pone un ulteriore punto di arrivo per questa ricerca.

Ciononostante, il metodo sviluppato presenta diversi limiti a cui si legano diverse possibilità di sviluppo futuro.
Innanzitutto, dal punto di vista tecnologico, avendo a disposizione hardware più performanti e affinando la pipeline utilizzata è possibile migliorare i tempi di training e testing dei modelli. In merito ai training una considerazione va fatta sulla loro flessibilità: i diversi test condotti dimostrano come le performance di ogni modello crescono tanto più quanto esso viene applicato su ambienti con condizioni vicine a quelle di addestramento. Questo accade prevalentemente in riferimento all'inserimento di ostacoli e relativamente al tema della connettività degli assemblaggi.
Un'ulteriore limite di questo studio può essere inquadrato nell'ambito del design delle parti, proprio della progettazione combinatoria. Affrontando il problema di ricerca in una maniera semplificata, ci si è dovuti fermare ad un processo di analisi architettonica limitato ad una scala molto generale, relativa al solo massing degli assemblaggi (quella inquadrata dai KPI utilizzati).

Risvolti interessanti potrebbe avere l'estensione della ricerca in questo senso: Il processo potrebbe essere implementato andando a dettagliare le unità in gioco affidandosi a criteri di progettazione che includono riflessioni sul clima, come ad esempio quelli suggeriti dalla bioclimatica. si potrebbe definire un framework RL che premi all'interno di ogni singola unità, il posizionamento di elementi di natura differente (come parti opache, con grado di trasparenza variabile, sistemi di om-

breggiamento, etc..) a seconda delle condizioni climatiche ad esse più idonee. Questo consentirebbe di restituire assemblaggi più completi e analizzabili in maniera più minuziosa sotto diversi punti di vista non presi in considerazione in questa ricerca. È bene però tenere a mente che scendere ad una scala più raffinata delle parti resta un problema da trattare facendo attenzione a limitare la varietà delle stesse, sia per non accrescere eccessivamente la complessità del sistema, sia per privilegiarne la sua scalabilità (rinunciando ad un grande numero di "soluzioni ottimali" specifiche, preferendo una risposta buona abbastanza se commisurata agli obiettivi di progettazione ma aperta a più scenari).

Possibilità future sono quelle riguardanti l'estensione del metodo ad environment di natura differente: In un primo senso si potrebbe ipotizzare di passare da environment di tipo discreto, come quelli utilizzati, ad altri di tipo continuo, consentendo possibilità di aggregazione alle parti che esulino dalla rigidità di una matrice tridimensionale.

Infine, l'intero processo potrebbe essere esteso ad altri fattori climatici oltre che al numero di ore di illuminazione diurna. Ventilazione naturale e radiazione solare incidente sono solo alcune delle informazioni che meritano di essere investigate, per ottenere dei modelli in grado di sfruttare le informazioni provenienti sul clima con maggiore completezza. Questo sviluppo andrebbe condotto non con un'idea di incremento della complessità delle condizioni che il sistema è in grado di gestire ma come tentativo di estendere la cognizione che il modello acquisisce dell'ambiente in cui si genera, a dati maggiormente aderenti alla complessità della realtà.

Bibliografia

Alexander, C. (1968) Systems Generating Systems.

Alexander, C. (2019) *A city is not a tree. 50th anniversary edition*. Edited by M.W. Mehaffy. Portland, Oregon: Sustasis Press Association with Center for Environmental Structure.

Bava, A. (2020) 'Computational Tendencies'. Available at: https://www.e-flux.com/architecture/intelligence/310405/computational-tendencies/.

Bratton, B. (2019) 'The Terraforming. Strelka Press - Automation as Ecology chapter.', in.

Carpo, M. (2011) *The alphabet and the algorithm*. Cambridge, Mass.: MIT Press.

Colchester, J. (2016) *Systems + Complexity - An Overview*. (Complexity Theory Book).

DeLanda, M. (2010) 'Emergence.', Lebbeus Wood. Available at: https://lebbeuswoods.wordpress.com/2010/07/27/manuel-delanda-emergence/.

DeLanda, M. (no date) 'Materialism for Architects'.

Goodfellow, I., Bengio, J. and Courville, A. (2016) DeepLearning book.

Google (2018a) 'Machine Learning Crash Course'. Available at: https://developers.google.com/machine-learning/crash-course?hl=it.

Google (2018b) 'Machine Learning Glossary'. Available at: https://developers.google.com/machine-learning/glossary?hl=it.

Graaf, R. de (2017) *Four walls and a roof: the complex nature of a simple profession*. Cambridge, Massachusetts London, England: Harvard University Press.

Hugging Face (no date) 'Deep RL Course', Hugging Face. Available at: https://huggingface.co/learn/deep-rl-course/unit0/introduction.

IBM (2021) 'Cos'è l'intelligenza artificiale (AI)?' Available at: https://www.ibm.com/it-it/topics/artificial-intelligence.

Jacobs, K. (2015) 'Moshe Safdie and the Revival of Habitat 67'. Available at: https://www.architectmagazine.com/awards/moshe-safdie-and-the-revival-of-habitat-67_o#:~:text=(The%20U.S.%20pavilion%2C%20for%20instance,than%20most%20architects%20ever%20achieve.

Johnson, S. (2002) *Emergence: the connected lives of ants, brains, cities and software*. London: Penguin Books.

Koehler, D. (2021a) 'One Minute Architecture'. Available at: https://lab-eds.org/One-Minute-Architecture.

Koehler, D. (2021b) 'Stairs and Slabs'. Available at: https://lab-eds.org/Stairs-and-Slabs-1.

Koehler, D. (2022) 'ThreeNine'. Available at: https://lab-eds.org/ThreeNine.

Murphy, D. (2016) *Last futures: nature, technology and the end of architecture*. London: Verso.

OpenAI (2021) 'Stable-baselines-3'. Available at: (https://stable-baselines.readthedocs.io/en/master/guide/algos.html).

Pasquinelli, M. (2019) 'Three Thousand Years of Algorithmic Rituals: The Emergence of AI from the Computation of Space'. Available at: https://www.e-flux.com/journal/101/273221/three-thousand-years-of-algorithmic-rituals-the-emergence-of-ai-from-the-computation-of-space/.

Pasquinelli, M. (2020) 'The Nooscope — a visual manifesto of the limits of AI'. Available at: https://www.skynettoday.com/editorials/nooscope.

Royal Society (2017) *Machine learning: the power and promise of computers that learn by example*. The Royal Society.

Sanchez, J. (2021) *Architecture for the commons: participatory systems in the age of platforms*. New York London: Routledge, Taylor & Francis Group.

Silver, D. et al. (2016) 'Mastering the game of Go with deep neural networks and tree search', Nature, 529(7587), pp. 484–489. Available at: https://doi.org/10.1038/nature16961.

Steenson, M.W. (2017) *Architectural intelligence: how designers, and architects created the digital landscape*. Cambridge, MA: The MIT Press.

Sutton, R.S. and Barto, A. (2014) *Reinforcement learning: an introduction*. Nachdruck. Cambridge, Massachusetts: The MIT Press (Adaptive computation and machine learning).

Veloso, P. and Krishnamurti, R. (2022) 'An Academy of Spatial Agents. Generating spatial configurations with deep reinforcement learning'.

Walsh, N.P. (2023) 'Habitat 67's original vision recreated in Unreal Engine by Safdie Architects, Epic Games, and Neoscape'. Available at: https://archinect.com/news/article/150349986/habitat-67-s-original-vision-recreated-in-unreal-engine-by-safdie-architects-epic-games-and-neoscape.

Wang, D. and Snooks, R. (2021) 'Artificial Intuitions of Generative Design: An Approach Based on Reinforcement Learning.'

Watanabe, M.S. (1994) 'Induction Design', Architectural Intelligence by Artificial Intelligence. Available at: https://www.makoto-architect.com/aitect.html.

Watanabe, M.S. (1995) 'Sun God city 2', Makoto-architect. Available at: https://www.makoto-architect.com/sun_god_city2.html.

Wietschorke, Liu and Chen (2021) 'Reinforcement Learning for Architectural Combinatorial Optimization'.

Zevi, B. (1995) *Saper vedere l'architettura: saggio sull'interpretazione spaziale dell'architettura.* Torino: Einaudi (Einaudi tascabili Saggi, 152).

Software

McNeel, R., 2022. Rhino (Version 7.0) [software]. Seattle: McNeel & Associates. Available at: https://www.rhino3d.com

McNeel, R., 2022. Grasshopper (Version 1.0) [software]. Seattle: McNeel & Associates. Available at: https://www.grasshopper3d.com.

Ladybug Tools LLC, 2024. Ladybug Tools (Version 1.4.0) [software]. Available at: https://www.ladybug.tools.

TensorFlow Team, 2024. TensorBoard (Version 2.9.0) [software]. Available at: https://www.tensorflow.org/tensorboard.

Hill, A., Raffin, A., Ernestus, M., Gleave, A., Kanervisto, A., and Dormann, N., 2024. Stable-Baselines3 (Version 1.5.1) [software]. Available at: https://github.com/DLR-RM/stable-baselines3.

Farama Foundation, 2024. Gymnasium (Version 0.27.0) [software]. Available at: https://gymnasium.farama.org/index.html

Zwierzycki, M., 2011. Hoopsnake (Version 0.6.5) [software]. Available at: https://www.food4rhino.com/en/app/hoopsnake.

Ringraziamenti

Questa tesi chiude un percorso intenso, ricco di emozioni, positive e negative, che mai probabilmente avrei pensato di vivere e che mi porterò dietro per sempre.

Arrivato a questo punto vorrei innanzitutto ringraziare il prof. Alessio Erioli. Vorrei sottolineare la sua disponibilità, grazie alla quale ho potuto portare avanti questo percorso, e la sua capacità nel creare un ambiente di studio costruttivo. Questo mi ha permesso di trovare un mio indirizzo di indagine architettonica personale e arricchire il mio bagaglio di umano oltre che quello di conoscenze teoriche.

Tutto quello che ho fatto per arrivare sino a questo giorno non sarebbe mai stato possibile senza il supporto della mia famiglia a cui voglio esprimere la mia gratitudine.

Grazie a mia madre, che non smette mai un secondo di credere in me, stando dalla mia parte in maniera incondizionata e contribuendo a lasciar esprimere la mia parte più creativa e istintiva.

Ringrazio mio padre, sempre presente a trasmettermi calma, pazienza e lucidità, anche e soprattutto nei momenti meno semplici e capace di farmi credere in valori per me fondamentali come il sacrificio, la semplicità e il rispetto verso gli altri.

Ringrazio mio fratello Angelo che in questi anni è stato un punto di riferimento razionale e maturo, non solo per i tanti consigli e confronti, ma anche fermandosi spesso ad ascoltare i miei punti di vista. Grazie a Beatrice, la sua presenza a Bologna è stata per certi versi come quella di una sorella maggiore. Il bene che mi avete dimostrato mi ha fatto sentire di essere a casa anche quando ero lontano e di questo vi sono grato.

Ringrazio le mie nonne, Nunziatina e Ninetta, e tutti i miei zii, cugini e famigliari per il supporto che nonostante le difficoltà in un modo o nell'altro sono riusciti a darmi. Un pensiero va ai miei nonni, con un sorriso ad immaginarvi ancora qui.

Un grazie speciale va a Danila, forse l'unica in grado di riuscire a capirmi veramente. I suoi modi, la sua dolcezza, la sua gentilezza e il suo guardare sempre il bene nelle cose mi hanno accompagnato in questa esperienza in una maniera che non dimenticherò mai. Grazie per essere sempre al mio fianco.

ReLighting spaces

Tesi di Laurea in Architettura e Composizione architettonica III

Corso di Ingegneria Edile-Architettura

Dipartimento di Architettura

Alma Mater Studiorum - Università di Bologna

di Giuseppe Massafra

Relatore: prof. Alessio Erioli