Griglia di finestre

Descrizione problema

Si ha una facciata rettangolare di un edificio, di date dimensioni $b_w \times b_h$. Si vogliono disporre finestre quadrate (di dimensioni $w_{size} \times w_{size}$), secondo un semplice schema a griglia, volendo:

- massimizzare l'area coperta dalla griglia;
- massimizzare il numero di finestre, rispetto allo spazio vuoto

La disposizione è soggetta ai seguenti vincoli:

- il layout deve rientrare nelle dimensioni della facciata:
  sia n_w il numero di finestre lungo un piano dell'edificio (cioè il numero di colonne della griglia), e sia s_w lo spazio tra due finestre consecutive. Allora deve valere $n_w \times w_{size} + (n_w - 1) \times s_w \leq b_w - k$, dove k è un valore (fisso o in percentuale su b_w) che permette di avere spazio tra le prime e ultime finestre e i bordi della facciata. Analogo discorso vale verticalmente.
- la distanza tra due finestre non deve essere inferiore a un dato valore minimo.

Modellazione

Il problema si può scomporre in due sottoproblemi differenti solo nei parametri, per la disposizone di righe e colonne, di tipo MIQCLP (Mixed-Integer Quadratically Constrained Linear Program). Descriviamo il problema di individuare numero e posizione per le finestre lungo una riga.
Input:

- b_w: larghezza della facciata dell'edificio
- $s_{w_{min}}$: minima distanza tra due finestre consecutive

Variabili di ottimizzazione:

- n_w: numero di finestre lungo una riga, intero positivo

- s_w: spazio tra due finestre consecutive lungo una stessa riga, reale positivo

Vincoli:

- $n_w * w_{size} + (n_w - 1) * s_w \leq b_w - k$
- $s_w \geq s_{w_{min}}$

Funzione obiettivo (da massimizzare):
$f(n_w, s_w) = n_w * M + s_w$
M è un valore sufficientemente grande per permettere al numero di finestre di avere priorità su s_w (un buon valore è b_w).

Risoluzione - Approccio 1

La modellazione del problema è banale in Gurobi, ma la forma del problema rende difficile ottenere buone soluzioni [1].

Risoluzione - Approccio 2

Considerando che si è deciso di prioritizzare la massimizzazione del numero di finestre lungo una riga/colonna, il problema si può semplificare considerando il numero di finestre come un input del problema, e massimizzare la sola distanza tra le finestre. A questo punto si ricerca il valore massimo di n_w per cui il problema, adesso un LP, sia risolvibile. Per velocizzare la ricerca, si può limitare n_w all'intervallo tra 0 (nessuna finestra) e $\left\lfloor \frac{b_w - k}{w_{size}} \right\rfloor$ e procedere con una ricerca binaria.

Risoluzione - Approccio 3

È possibile ottenere buone soluzioni tramite Particle Swarm Optimization, modificando il problema come segue:

- n_w diventa una variabile a valori reali.
- Si usa una nuova funzione obiettivo:
  $f(n_w, s_w) = n_w * M + s_w - frac(n_w) * M_1 - constraint\_penalty * M_2$,
  dove

[1] `https://groups.google.com/forum/#!topic/gurobi/VPGidwIa37k`

- $frac(x) \coloneqq x - \lfloor x \rfloor$, usato per dare penalità a valori di n_w non interi. In caso l'algoritmo dia un valore di n_w non intero è (generalmente) possibile approssimare e verificare che i vincoli del problema originale siano ancora rispettati.
- $constraint\_penalty \coloneqq max(n_w * w_{size} + (n_w - 1) * s_w - (b_w - k), 0)$. È una penalità sulla violazione del vincolo di estensione massima della riga. Il vincolo sulla distanza minima tra due finestre è espresso implicitamente al momento della definizione del dominio di s_w.
- M_1 e M_2 sono costanti sufficientemente grandi (nelle prove 1e6).

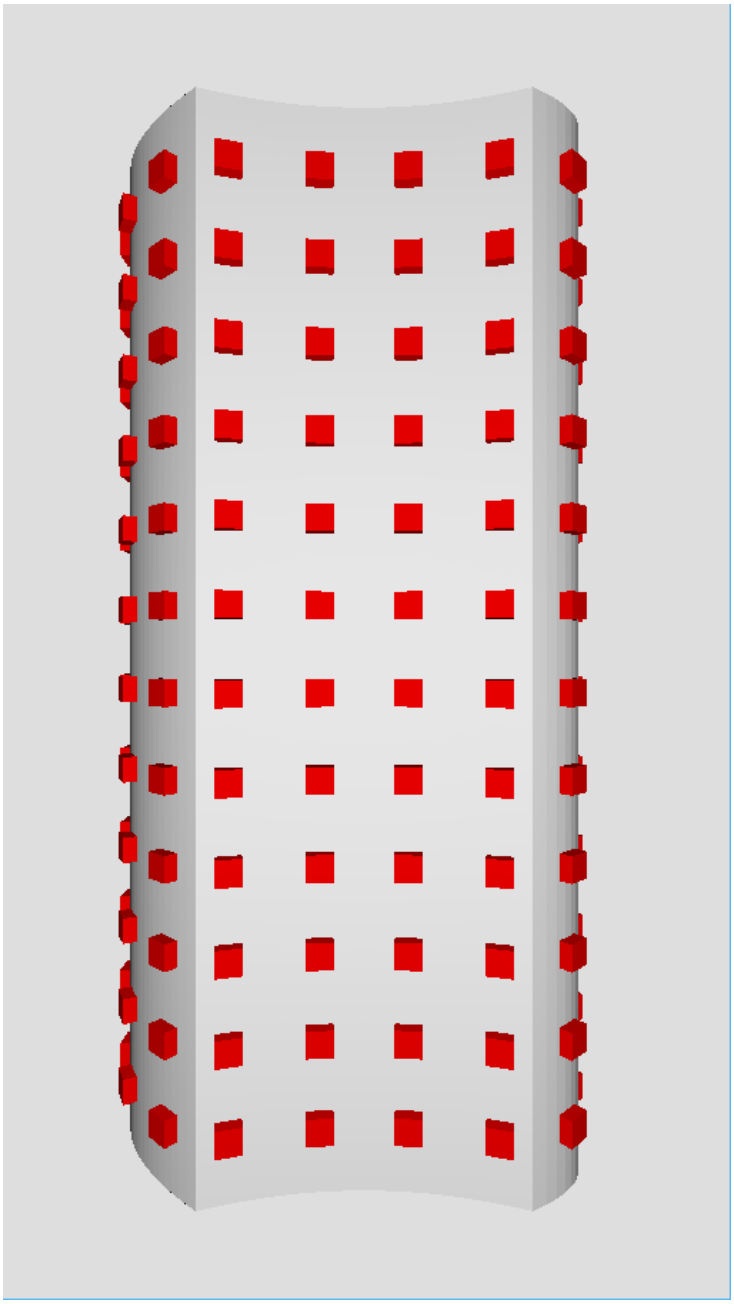
Il principale vantaggio di questa soluzione rispetto alla precedente è la capacità di poter formulare e risolvere problemi in maniera unificata anche rispetto a funzioni obiettivo o di vincolo più complesse o non trattabili dai risolutori matematici classici, anche se formalizzazioni ad-hoc per lo specifico problema di riferimento possono portare a soluzioni migliori, se non ottime.
I risultati di quest'ultimo approccio restano comunque molto buoni e sono mostrati nella sezione *Risultati*.

Disposizione delle finestre

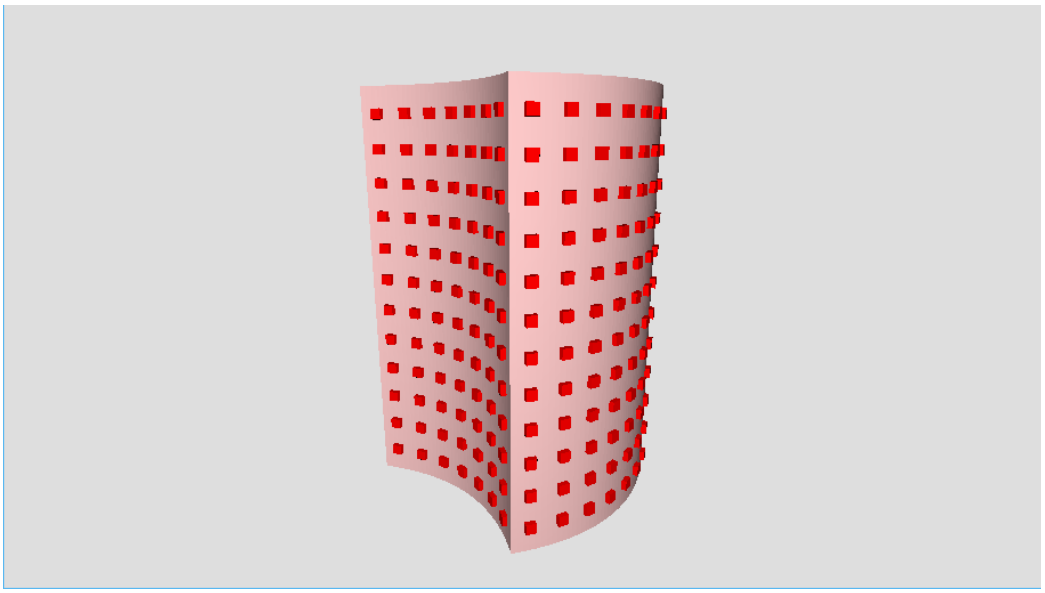
Una volta trovati i valori di n_w e s_w (e degli analoghi per il problema della disposizione lungo le colonne, n_h e s_h), si dispongono le finestre come segue. Si definisce $e_w := \frac{b_w - n_w * w_{size} - (n_w - 1) * s_w}{2}$, a indicare lo spazio tra il margine sinistro della facciata e il margine sinistro della prima finestra. Si definisce analogamente e_h.
Usando un sistema di coordinate dove $(0, 0)$ corrisponde allo spigolo in basso a sinistra della facciata e (b_w, b_h) a quello in alto a destra (facciata vista frontalmente), la x-esima (partendo da sinistra) finestra della y-esima riga (partendo dal basso) avrà lo spigolo in basso a sinistra in coordinate $(e_w + (x - 1) * (w_{size} + s_w), e_h + (y - 1) * (w_{size} + s_h))$.

Risultati

Di seguito alcuni render:



Layout a griglia semplice



Layout a griglia semplice