

# Progetto per il corso *Algoritmi e modelli per l'ottimizzazione discreta*

Sara Malaspina, Silvia Perelli

---

# Problema

---

L'obiettivo del progetto è risolvere un problema di scheduling con due giocatori, ciascuno dei quali possiede un insieme di job da svolgere su una singola macchina.

Ciascun giocatore ha una funzione obiettivo che è influenzata dalla posizione dei job dell'altro giocatore ed entrambi vorrebbero cercare di minimizzare la somma dei tempi di completamento dei propri job.

Tuttavia, il problema è NP hard.

# Modelli

---

Abbiamo modellato il problema di scheduling come un problema di programmazione lineare intera, considerando tre varianti:

- Soluzioni Min-Max
- Soluzioni di Kalai-Smorodinsky
- Soluzioni pesate con parametro  $\alpha$

Input:

- $J_1$ : insieme di job del giocatore 1
- $J_2$ : insieme di job del giocatore 2
- $n$ : numero di job totali
- $p[i]$ : durata del job  $i$

# Soluzioni Min-Max (1)

---

## Variabili temporali (continue)

- $s[i]$ : tempo di inizio del job  $i$   $\forall i \in \{1, \dots, n\}$
- $c[i]$ : tempo di completamento del job  $i$   $\forall i \in \{1, \dots, n\}$

## Variabili di precedenza (binarie)

- $x[i, j] = \begin{cases} 1 & \text{se il job } i \text{ precede il job } j \\ 0 & \text{altrimenti} \end{cases} \quad \forall 1 \leq i < j \leq n$

## Variabile obiettivo

- $z$ : tempo di completamento massimo tra i due giocatori

# Soluzioni Min-Max (2)

---

## Funzione Obiettivo

- $\min z$

## Vincoli sul tempo di completamento

- $c[i] = s[i] + p[i] \quad \forall i \in \{1, \dots, n\}$
- $z \geq \sum_{i \in J_1} c[i]$
- $z \geq \sum_{i \in J_2} c[i]$

## Vincoli di precedenza

- $s[j] \geq c[i] - M * (1 - x[i, j]) \quad \forall 1 \leq i < j \leq n$
- $s[i] \geq c[j] - M * x[i, j] \quad \forall 1 \leq i < j \leq n$

# Soluzioni di Kalai-Smorodinsky (1)

---

## Variabili temporali (continue)

- $s[i]$ : tempo di inizio del job  $i$   $\forall i \in \{1, \dots, n\}$
- $c[i]$ : tempo di completamento del job  $i$   $\forall i \in \{1, \dots, n\}$

## Variabili di precedenza (binarie)

- $x[i, j] = \begin{cases} 1 & \text{se il job } i \text{ precede il job } j \\ 0 & \text{altrimenti} \end{cases} \quad \forall 1 \leq i < j \leq n$

## Variabile obiettivo

- $z$ : tempo di completamento massimo (normalizzato) tra i due giocatori

# Soluzioni di Kalai-Smorodinsky (2)

---

## Funzione Obiettivo

- $\min z$

## Vincoli sul tempo di completamento

- $c[i] = s[i] + p[i] \quad \forall i \in \{1, \dots, n\}$
- $z \geq \frac{\sum_{i \in J_1} c[i]}{worst_1 - best_1}$
- $z \geq \frac{\sum_{i \in J_2} c[i]}{worst_2 - best_2}$

## Vincoli di precedenza

- $s[j] \geq c[i] - M * (1 - x[i, j]) \quad \forall 1 \leq i < j \leq n$
- $s[i] \geq c[j] - M * x[i, j] \quad \forall 1 \leq i < j \leq n$

# Soluzioni di Kalai-Smorodinsky (3)

---

Calcolo  $best_1$  e  $worst_1$  (analogo per il giocatore 2)

- $best_1$ : somma dei tempi di completamento dei job del giocatore 1 disposti prima dei job del giocatore 2 e in ordine Shortest Processing Time
- $worst_1$ : somma dei tempi di completamento dei job del giocatore 1 disposti dopo i job del giocatore 2 e in ordine Shortest Processing Time (anche i job del giocatore 2 sono disposti in ordine Shortest Processing Time)



# Soluzioni di Kalai-Smorodinsky (4)

---

Questo modello normalizza le funzioni obiettivo dei singoli giocatori, tenendo conto dei valori minimi e massimi che ciascuno di essi può ottenere.

Tale strategia è utile nei casi in cui i job di un giocatore durano molto meno dei job dell'altro, poiché utilizzando una soluzione Min-Max andremmo a favorire il giocatore con i job che durano di più.

# Soluzioni pesate (1)

---

## Variabili temporali (continue)

- $s[i]$ : tempo di inizio del job  $i$   $\forall i \in \{1, \dots, n\}$
- $c[i]$ : tempo di completamento del job  $i$   $\forall i \in \{1, \dots, n\}$

## Variabili di precedenza (binarie)

- $x[i, j] = \begin{cases} 1 & \text{se il job } i \text{ precede il job } j \\ 0 & \text{altrimenti} \end{cases}$   $\forall 1 \leq i < j \leq n$

## Funzione obiettivo

- $\min \alpha * \sum_{i \in J_1} c[i] + (1 - \alpha) * \sum_{i \in J_2} c[i]$   $\alpha \in [0, 1]$

# Soluzioni pesate (2)

---

Vincolo sul tempo di completamento

- $c[i] = s[i] + p[i] \quad \forall i \in \{1, \dots, n\}$

Vincoli di precedenza

- $s[j] \geq c[i] - M * (1 - x[i, j]) \quad \forall 1 \leq i < j \leq n$
- $s[i] \geq c[j] - M * x[i, j] \quad \forall 1 \leq i < j \leq n$

## Soluzioni pesate (3)

---

Questo modello restituisce, al variare del parametro  $\alpha$ , soluzioni che non sono dominate tra loro, ossia non riesco a migliorare la soluzione di uno dei due giocatori se non peggioro quella dell'altro.

Tali soluzioni appartengono al fronte di Pareto.

# Implementazione (1)

---

Abbiamo utilizzato il solver Gurobi tramite la libreria di Python gurobipy.

Il main del programma realizzato permette all'utente di scegliere tra 5 gruppi di istanze:

- Istanze simmetriche
- Istanze con pochi job (da 4 a 8) e bassa varianza tra le durate
- Istanze con molti job (da 8 a 12) e bassa varianza tra le durate
- Istanze con molti job (da 4 a 8) e alta varianza tra le durate
- Istanze con molti job (da 8 a 12) e alta varianza tra le durate

Per istanze con più di 12 job il solver non riesce a trovare la soluzione ottima in un tempo massimo di 5 minuti per ciascun modello.

## Implementazione (2)

---

Per ciascuna istanza all'interno del gruppo selezionato viene calcolata la soluzione ottima per i tre modelli proposti.

In particolare, nel caso delle soluzioni pesate, vengono calcolate 11 soluzioni ottime, facendo variare il parametro  $\alpha$  con passo 0.1.

Ogni modello restituisce anche la somma dei tempi di completamento dei job di ciascun giocatore e lo scheduling proposto.

# Output Modelli

```
Istanza n° 1:
Soluzione ottima trovata:
Valore di z: 101.0
Job 0: inizio = 10.0, completamento = 32.0
Job 1: inizio = 32.0, completamento = 65.0
Job 2: inizio = 65.0, completamento = 91.0
Job 3: inizio = 0.0, completamento = 10.0
x_0_1 = 1.0
x_0_2 = 1.0
x_0_3 = 0.0
x_1_2 = 1.0
x_1_3 = 0.0
x_2_3 = 0.0

Scheduling trovato:
Posizione 1: Job 3
Posizione 2: Job 0
Posizione 3: Job 1
Posizione 4: Job 2

Payoff giocatore 1: 97.0
Payoff giocatore 2: 101.0
```

Min-Max

```
Istanza n° 1:
Soluzione ottima trovata:
Valore di z: 1.2
Job 0: inizio = 0.0, completamento = 22.0
Job 1: inizio = 32.0, completamento = 65.0
Job 2: inizio = 65.0, completamento = 91.0
Job 3: inizio = 22.0, completamento = 32.0
x_0_1 = 1.0
x_0_2 = 1.0
x_0_3 = 1.0
x_1_2 = 1.0
x_1_3 = 0.0
x_2_3 = 0.0

Scheduling trovato:
Posizione 1: Job 0
Posizione 2: Job 3
Posizione 3: Job 1
Posizione 4: Job 2

Payoff giocatore 1: 87.0
Payoff giocatore 2: 123.0
```

Kalai-Smorodinsky

```
Istanza n° 1:
Soluzione ottima trovata:
Valore di obj: 95.5 per alpha 0.5
Job 0: inizio = 10.0, completamento = 32.0
Job 1: inizio = 58.0, completamento = 91.0
Job 2: inizio = 32.0, completamento = 58.0
Job 3: inizio = 0.0, completamento = 10.0
x_0_1 = 1.0
x_0_2 = 1.0
x_0_3 = 0.0
x_1_2 = 0.0
x_1_3 = 0.0
x_2_3 = 0.0

Scheduling trovato:
Posizione 1: Job 3
Posizione 2: Job 0
Posizione 3: Job 2
Posizione 4: Job 1

Payoff giocatore 1: 123.0
Payoff giocatore 2: 68.0
```

Pesata per  $\alpha = 0.5$

## Implementazione (3)

---

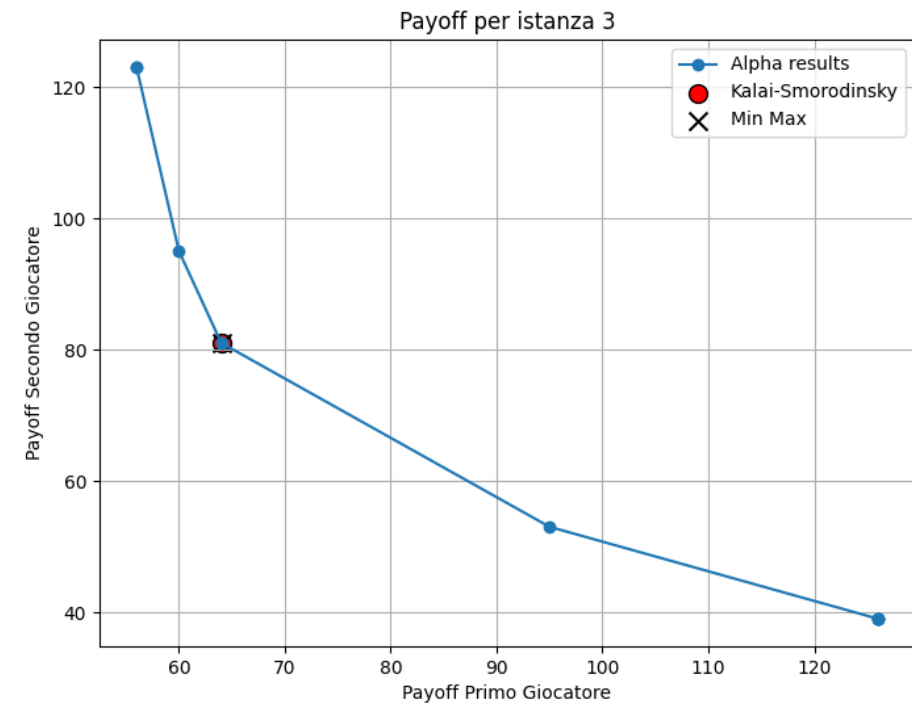
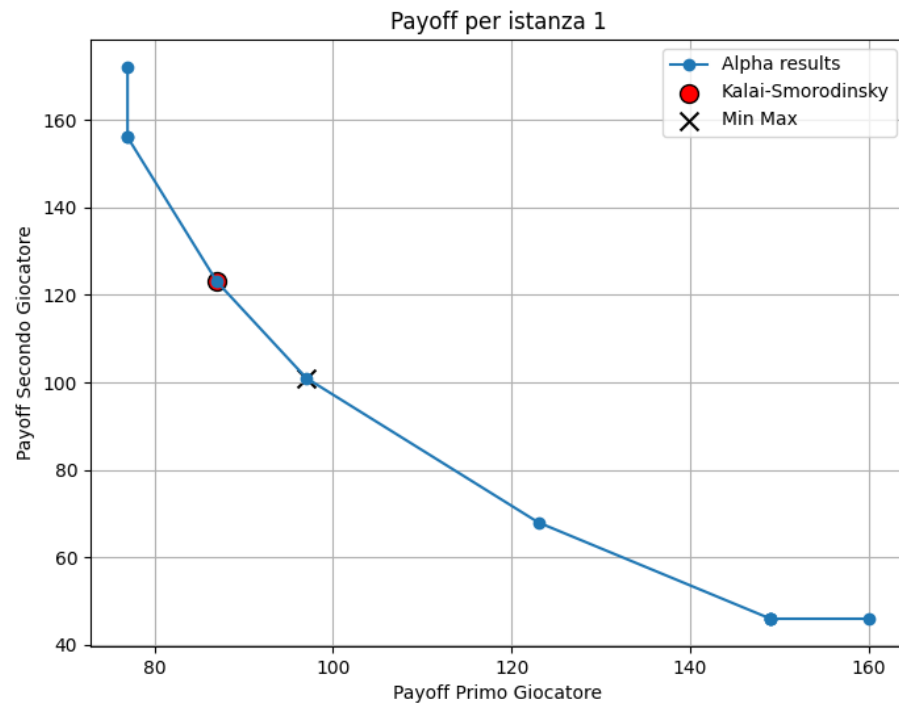
Abbiamo confrontato le soluzioni dei diversi modelli per via grafica riportando, per ogni istanza, la somma dei tempi di completamento del primo giocatore sulle ascisse e la somma dei tempi di completamento del secondo giocatore sulle ordinate.

Sul grafico sono state evidenziate la soluzione di Kalai-Smorodinsky, la soluzione Min-Max e le soluzioni pesate per ogni valore di  $\alpha$ .

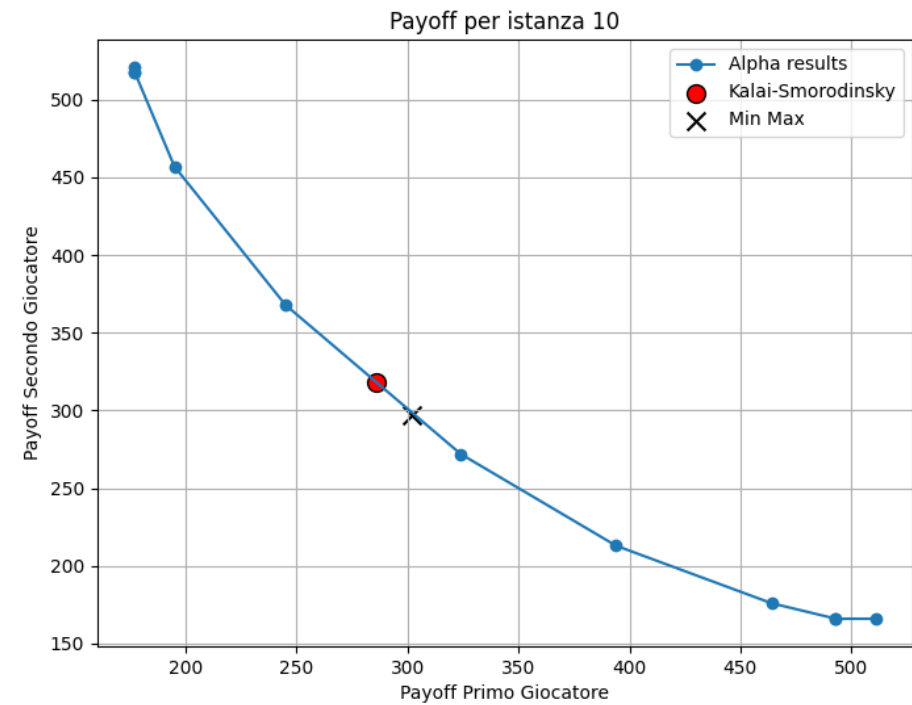
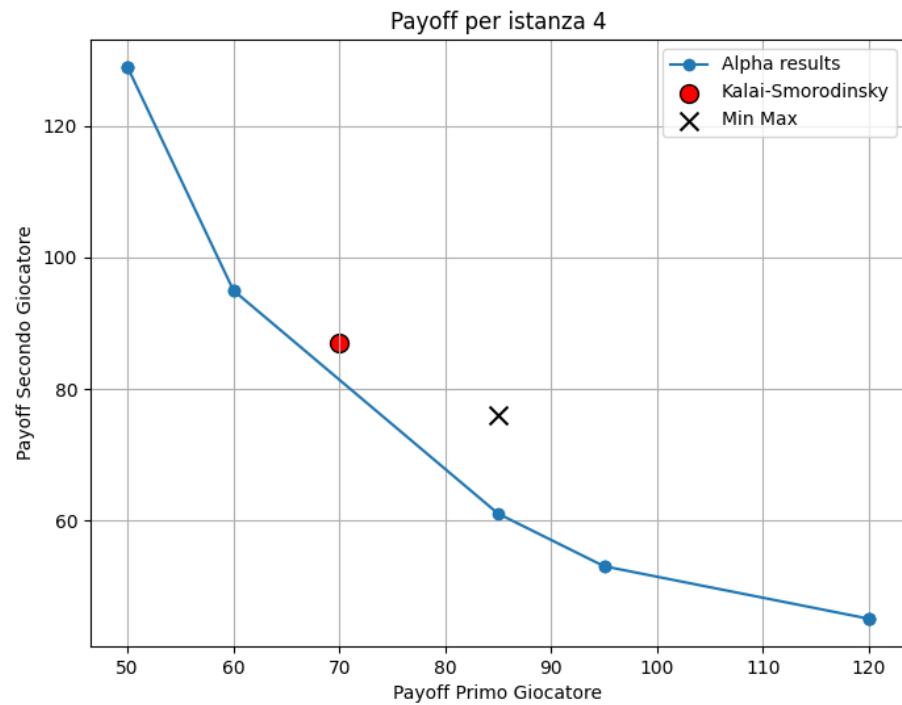
È possibile notare che per alcune istanze, la soluzione Min-Max o di Kalai-Smorodinsky coincidono con una delle soluzioni ottime per uno o più valori di  $\alpha$ .



# Grafici (1)



# Grafici (2)



# Grafici (3)

