Parallel machine scheduling

Gruppo 18

Bertè Sonia
Lochi Lorenza Maria
Sawan Omar
Verdi Federico

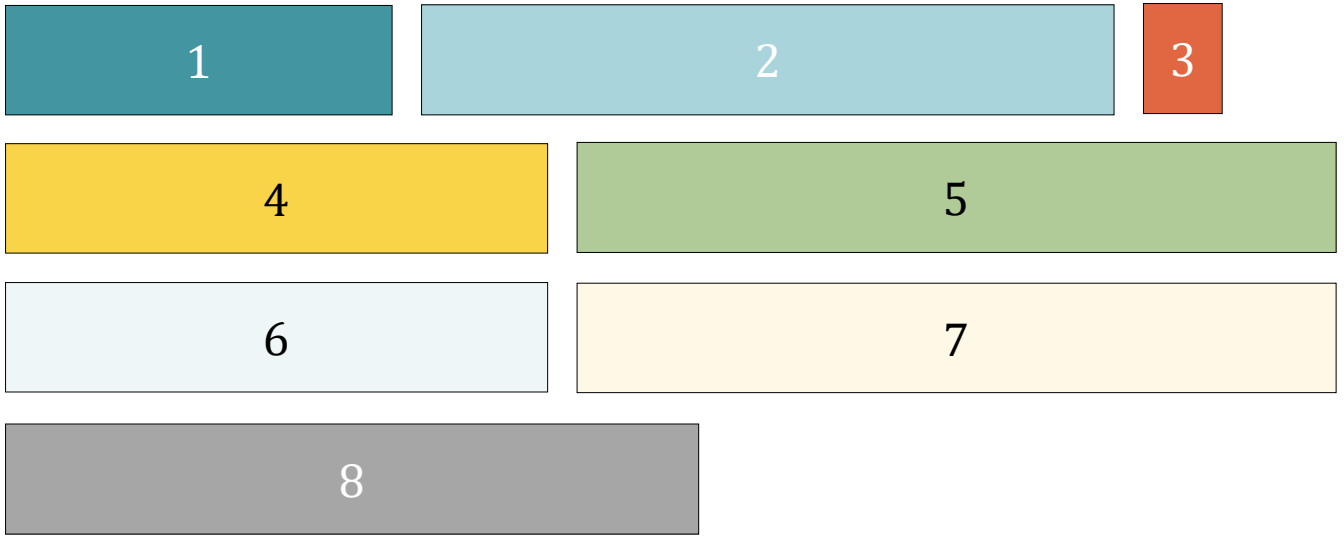
Descrizione del problema

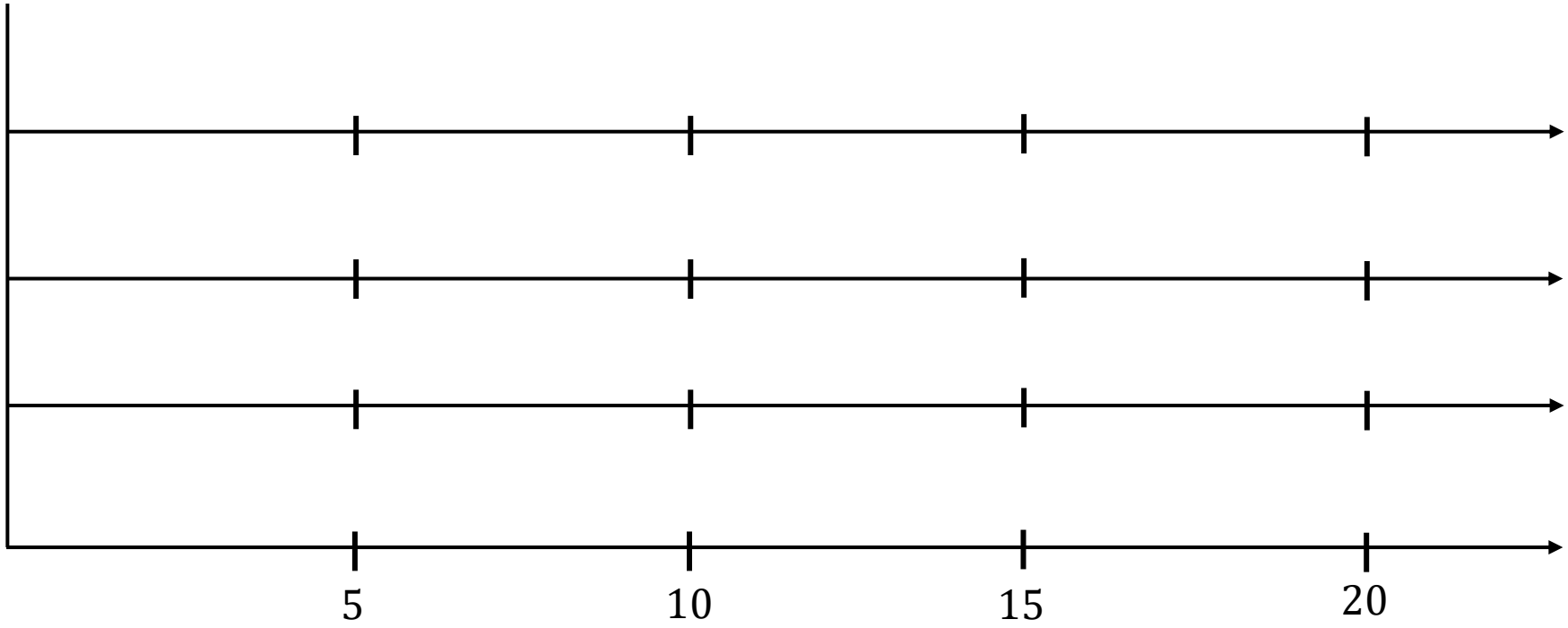
- Si abbiano J job, con $J = \{1, 2, \dots, n\}$, ed M macchine parallele identiche, con $M = \{1, 2, \dots, m\}$. A ciascun job è associato un tempo di processamento p_j ed una penalità w_j.
- Il problema richiede di schedulare tali job nelle macchine in modo da minimizzare la somma pesata dei tempi di completamento $\sum_{j=1}^{n} w_j C_j$, con C_j l'istante di completamento del j-esimo job.
- Tale problema può essere indicato attraverso la notazione a tre campi come: $P||\sum w_j C_j$. Il problema così descritto appartiene alla classe *NP-hard*.

- I job non possono essere interrotti o spostati da una macchina all'altra.
- Non sono presenti vincoli di precedenza tra i job.

Esempio

- $n = 8, m = 4$
- $p_j = \{5, 9, 1, 7, 10, 7, 10, 9\}$
- $w_j = \{15, 4, 39, 39, 2, 30, 28, 29\}$

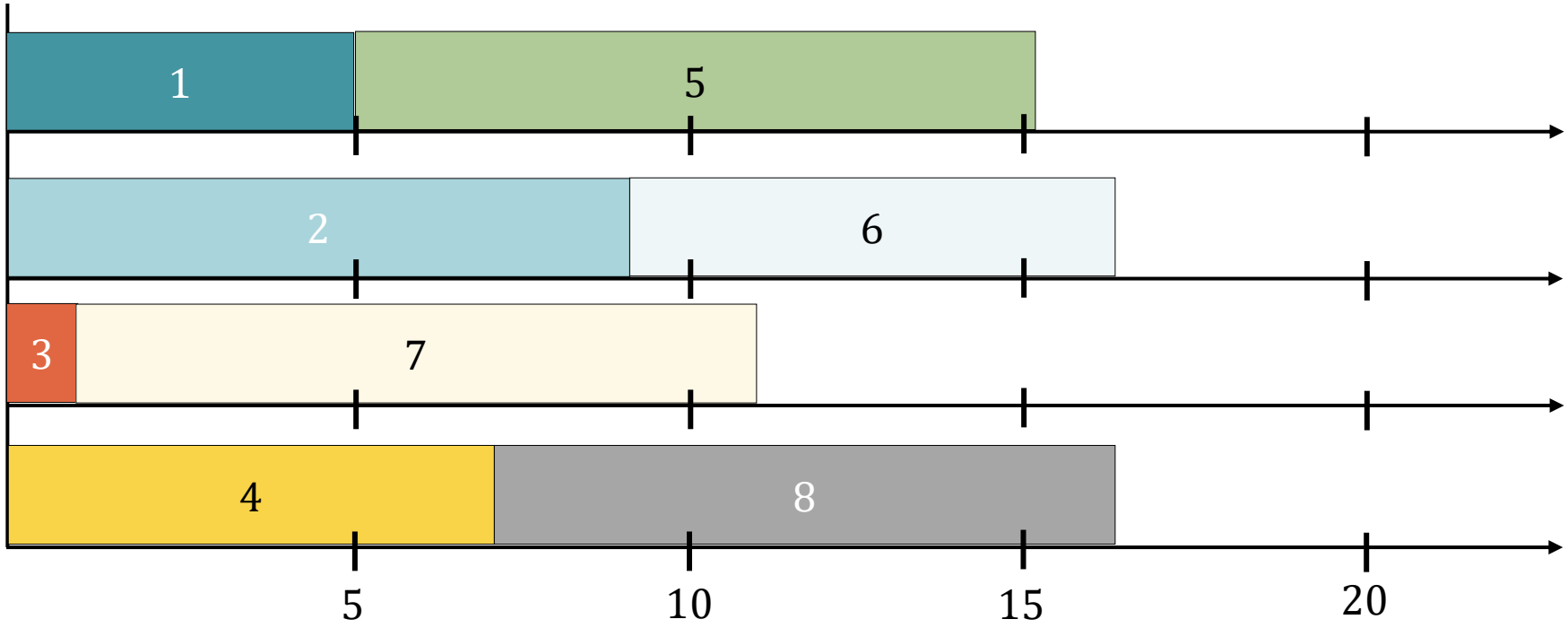




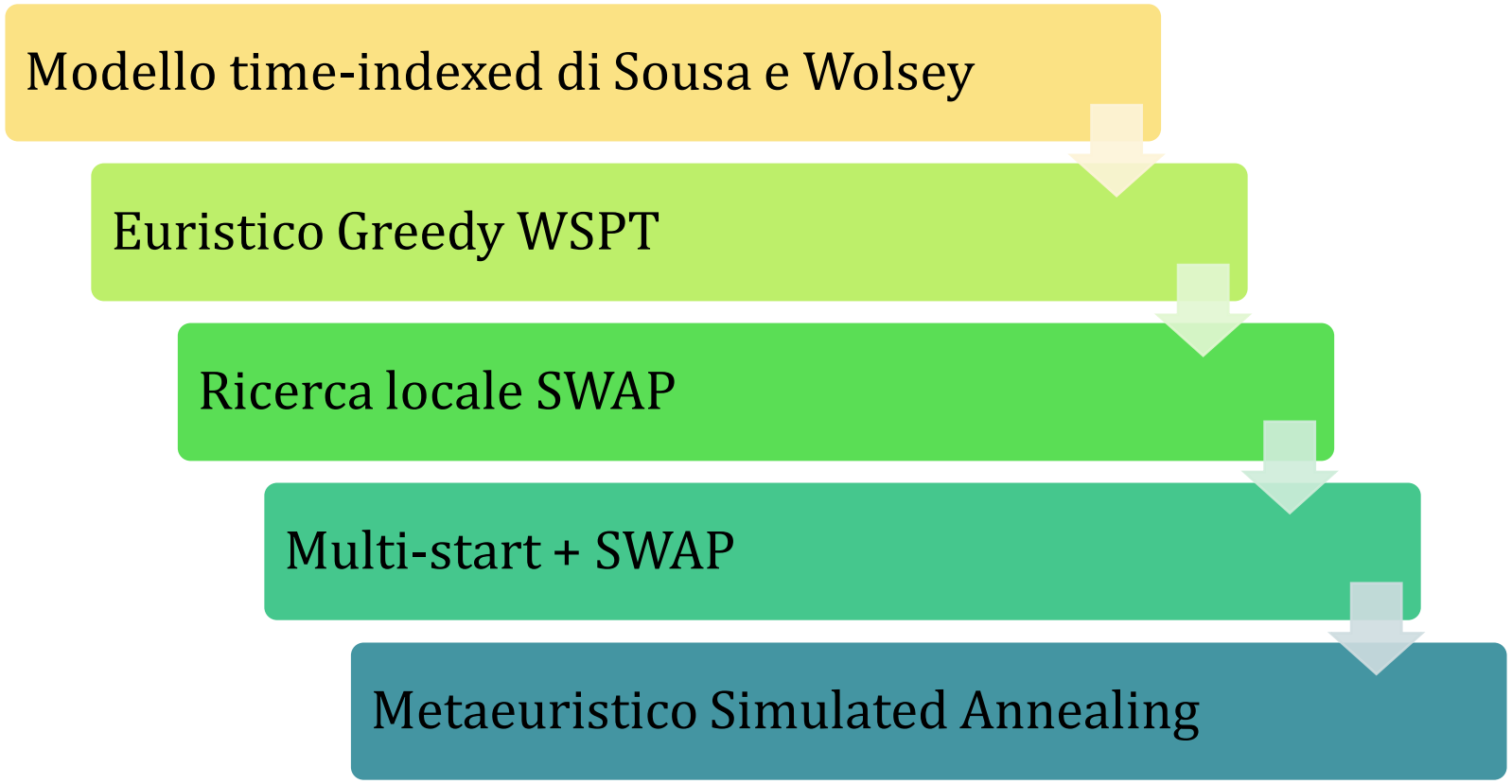
Esempio

- $n = 8, m = 4$
- $p_j = \{5, 9, 1, 7, 10, 7, 10, 9\}$
- $w_j = \{15, 4, 39, 39, 2, 30, 28, 29\}$

$$\boldsymbol{z} = \sum_{j=1}^{8} w_j C_j = 15 \times 5 + 4 \times 9 + 39 \times 1 + 39 \times 7 + 2 \times 15 + 30 \times 16 + 28 \times 11 + 29 \times 16 = \mathbf{1705}$$



Passaggi operativi



Modello time-indexed di Sousa e Wolsey

- $x_{jt} = \begin{cases} 1 & \text{se il j-esimo job inizia il suo processamento all'istante t} \\ 0 & \text{altrimenti} \end{cases}$ $\quad \forall j \in J, \forall t \in T - p_j$
- T: orizzonte temporale, calcolato come $T = \left\lfloor \frac{1}{m} \sum_{j \in J} p_j + \frac{(m-1)}{m} p_{max} \right\rfloor$
- p_{max}: massimo tempo di processamento tra tutti i job

Modello time-indexed di Sousa e Wolsey

$$C_j = \sum_t t x_{jt} + p_j$$

$$\text{(TI) } \min \sum_{j \in J} \sum_{t=0}^{T-p_j} w_j t x_{jt} + \sum_{j \in J} w_j p_j \qquad (1)$$

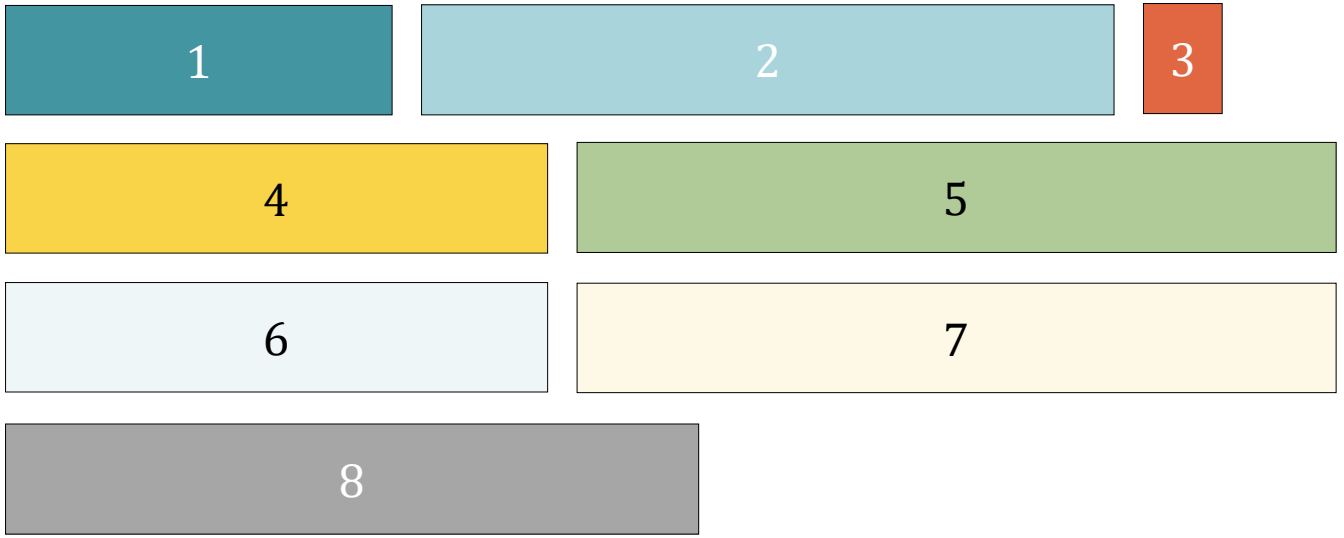
$$\text{st. } \sum_{t=0}^{T-p_j} x_{jt} = 1 \qquad j \in J \qquad (2)$$

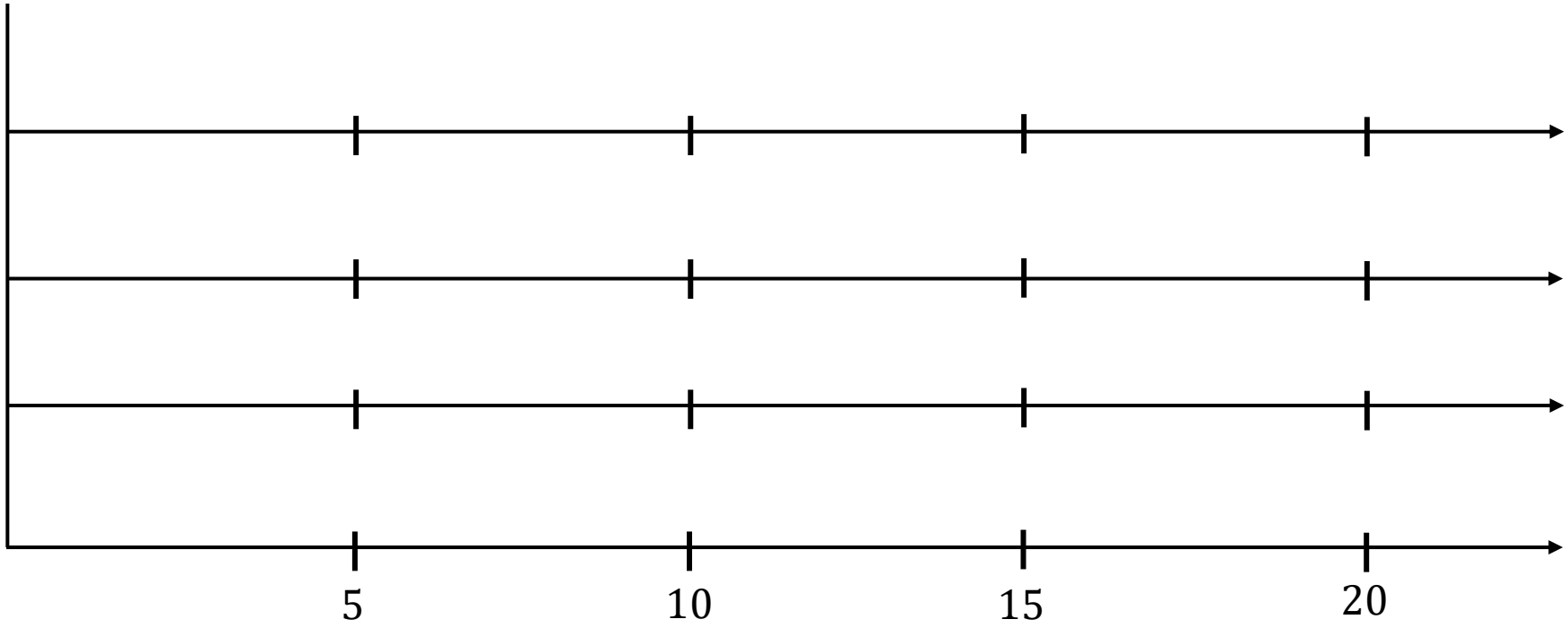
$$\sum_{j \in J} \sum_{s=max\{0,t+1-p_j\}}^{min\{t,T+1-p_j\}} x_{is} \leq m \qquad t = 0, \dots, T-1 \qquad (3)$$

$$x_{jt} \in \{0,1\} \qquad j \in J, t = 0, \dots, T-p_j \qquad (4)$$

Esempio modello

- $n = 8, m = 4$
- $p_j = \{5, 9, 1, 7, 10, 7, 10, 9\}$
- $w_j = \{15, 4, 39, 39, 2, 30, 28, 29\}$

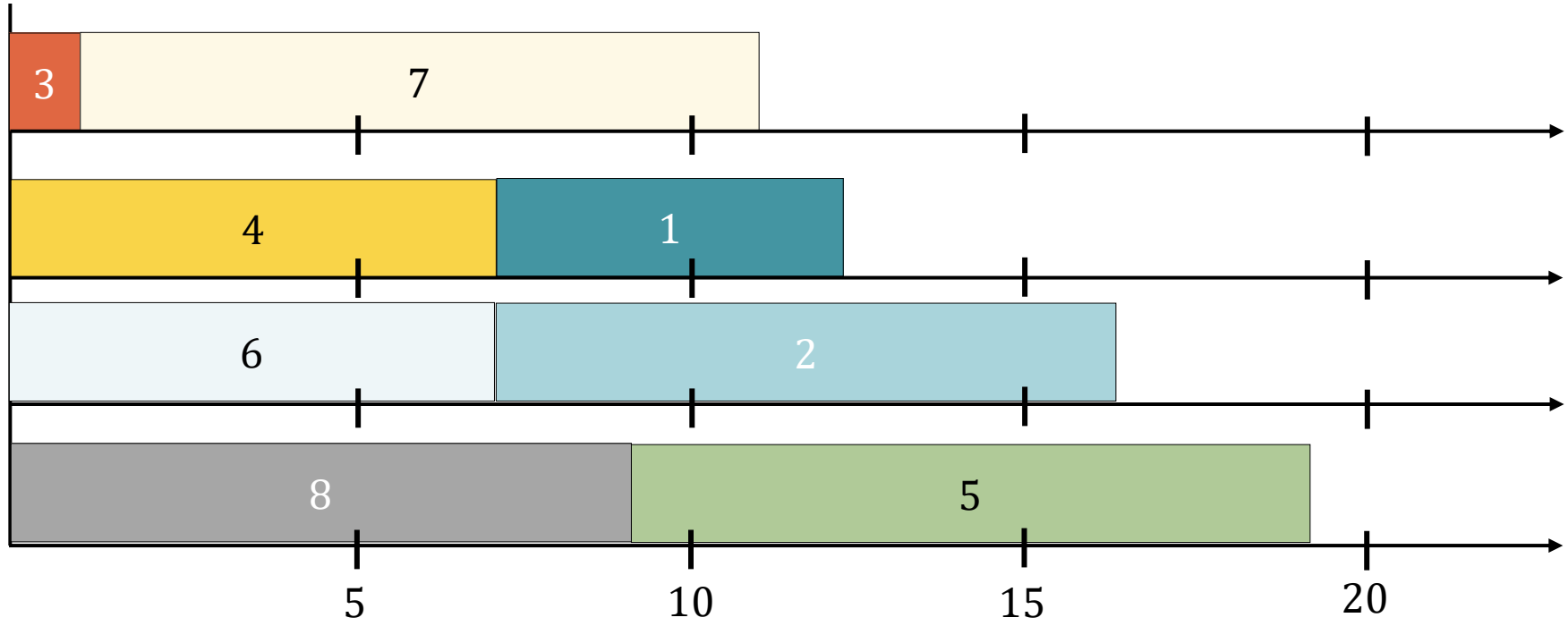




Esempio modello

- $n = 8, m = 4$
- $p_j = \{5, 9, 1, 7, 10, 7, 10, 9\}$
- $w_j = \{15, 4, 39, 39, 2, 30, 28, 29\}$

$$z = \sum_{j=1}^{8} w_j C_j = \mathbf{1373}$$



Algoritmo Euristico Greedy - WSPT

Descrizione

- Si effettui un ordinamento WSPT (Weighted Shortest Processing Time) in cui si ordinano i job secondo un ordinamento decrescente di w_j/p_j.
- Si assegni ciascun job, uno alla volta e secondo l'ordinamento precedentemente elaborato, alla macchina che risulta essere più scarica nell'istante di assegnamento del j-esimo job.

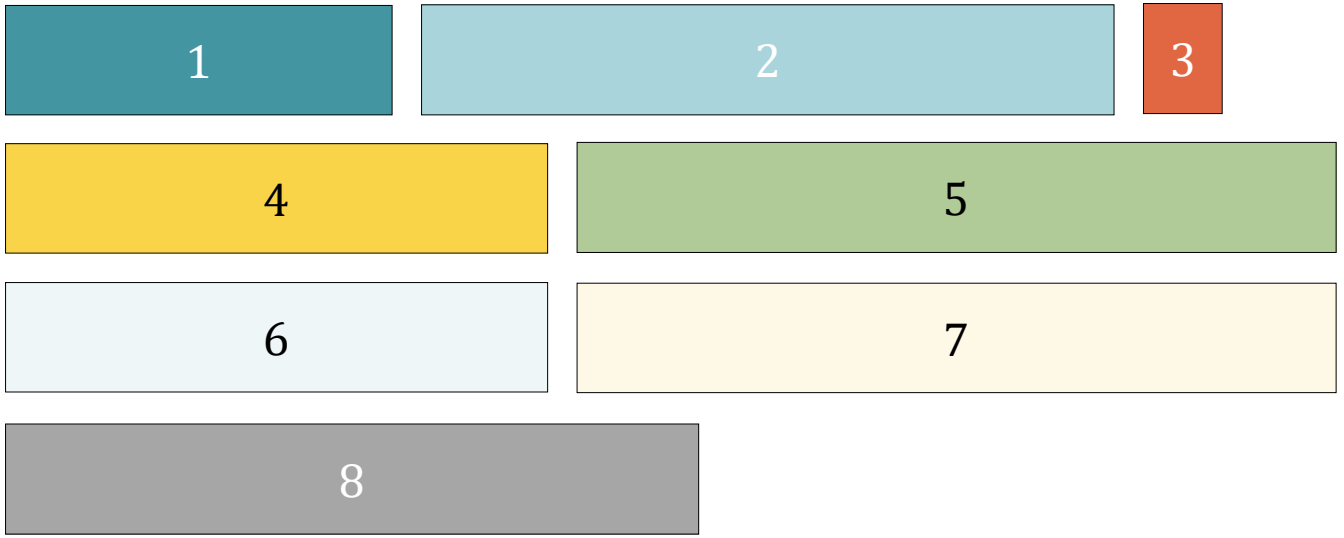
Algoritmo Euristico Greedy - WSPT

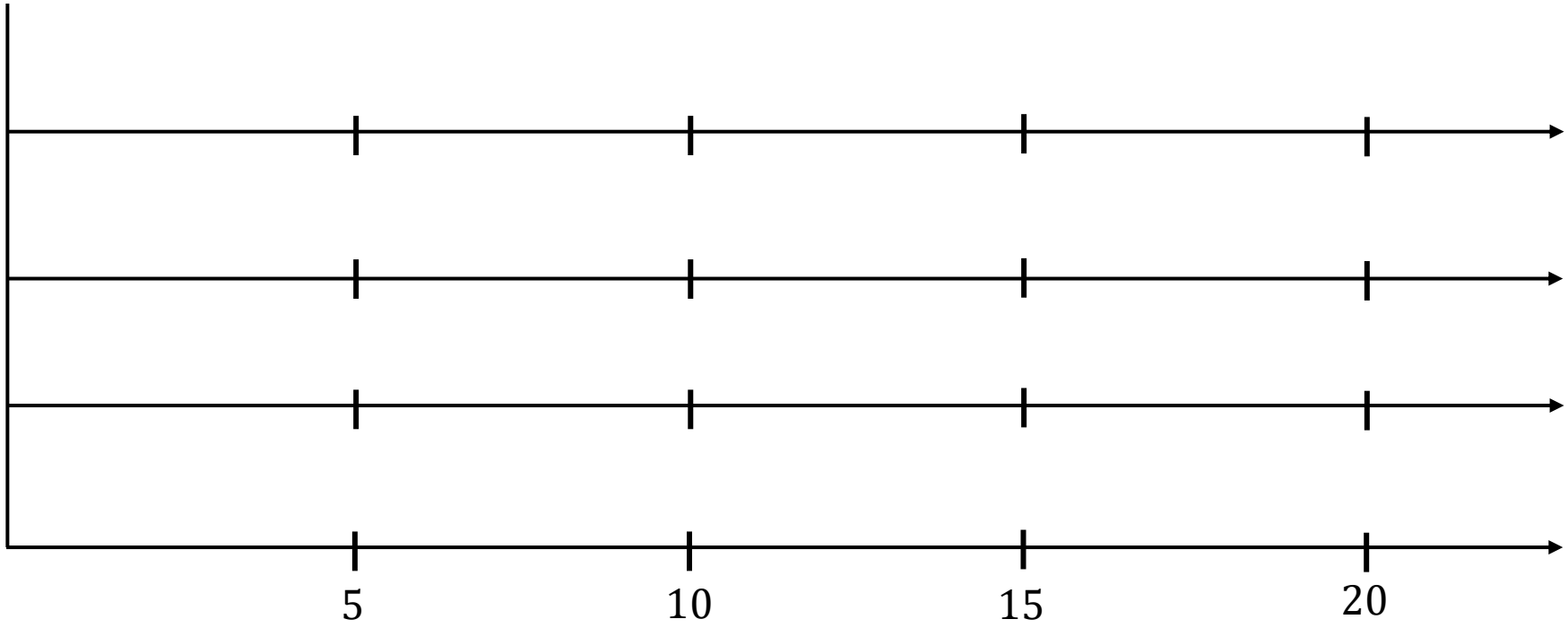
Pseudocodice

1. Ordinare i job per rapporti w_j/p_j decrescenti;
2. Inizializzare il valore della funzione obiettivo a 0; inizializzare il primo istante libero di ciascuna macchina a $t = 0$;
3. **For** (tutti i lavori) **do**
   - 3.1 Selezionare un job alla volta secondo l'ordinamento WSPT
   - 3.2 Individuare la macchina più scarica
   - 3.3 Inserire il job sulla macchina più scarica e aggiornare l'istante di liberazione di tale macchina
   - 3.4 Aggiornare il valore della funzione obiettivo

Esempio euristico WSPT

- $n = 8, m = 4$
- $p_j = \{5, 9, 1, 7, 10, 7, 10, 9\}$
- $w_j = \{15, 4, 39, 39, 2, 30, 28, 29\}$
- $ordine = [3, 4, 6, 8, 1, 7, 2, 5]$

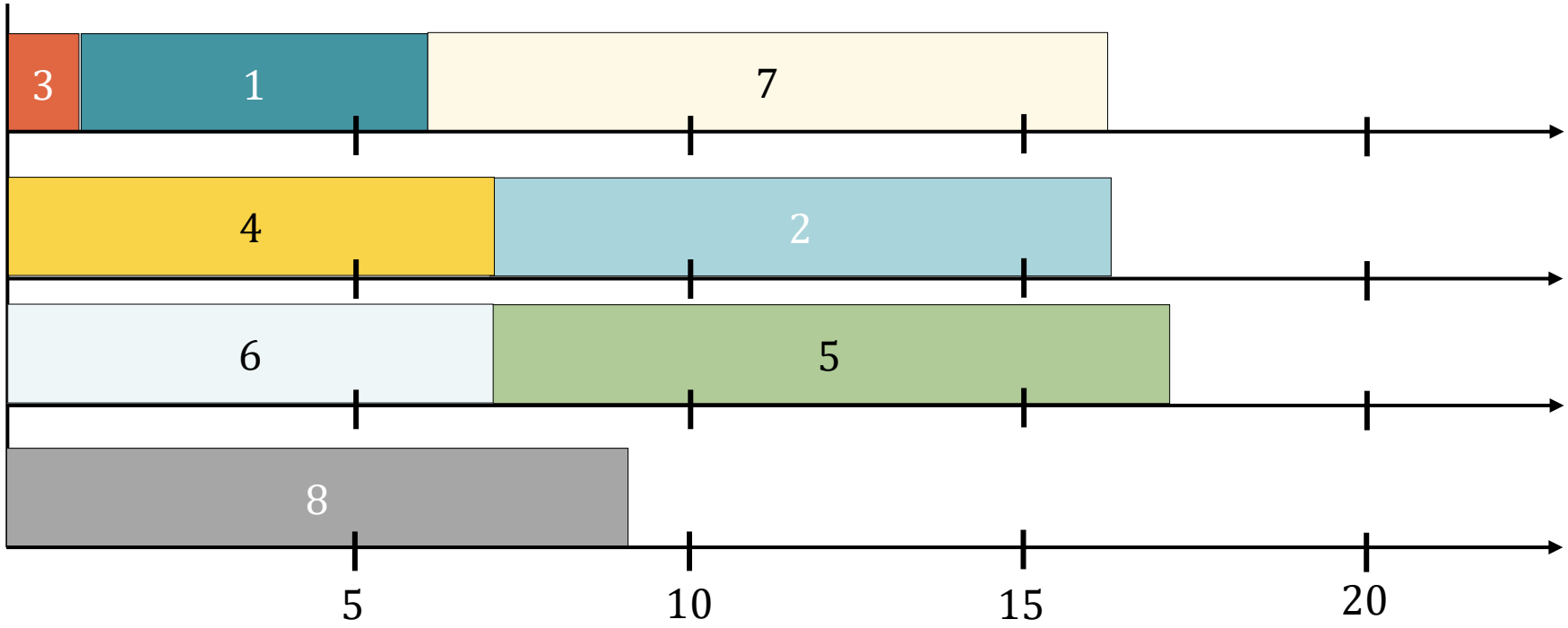




Esempio euristico WSPT

- $n = 8, m = 4$
- $p_j = \{5, 9, 1, 7, 10, 7, 10, 9\}$
- $w_j = \{15, 4, 39, 39, 2, 30, 28, 29\}$
- $ordine = [3, 4, 6, 8, 1, 7, 2, 5]$

$$z = \sum_{j=1}^{8} w_j C_j = \mathbf{1419}$$



Ricerca locale SWAP

Descrizione

- Si è implementata una ricerca locale di tipo SWAP, in ottica best-improvement, per migliorare:

  1. La soluzione ottenuta dall'algoritmo euristico con l'ordinamento WSPT

  2. Le soluzioni ottenute attraverso un approccio Multi-Start (come si vedrà in seguito)

Ricerca locale SWAP

Pseudocodice

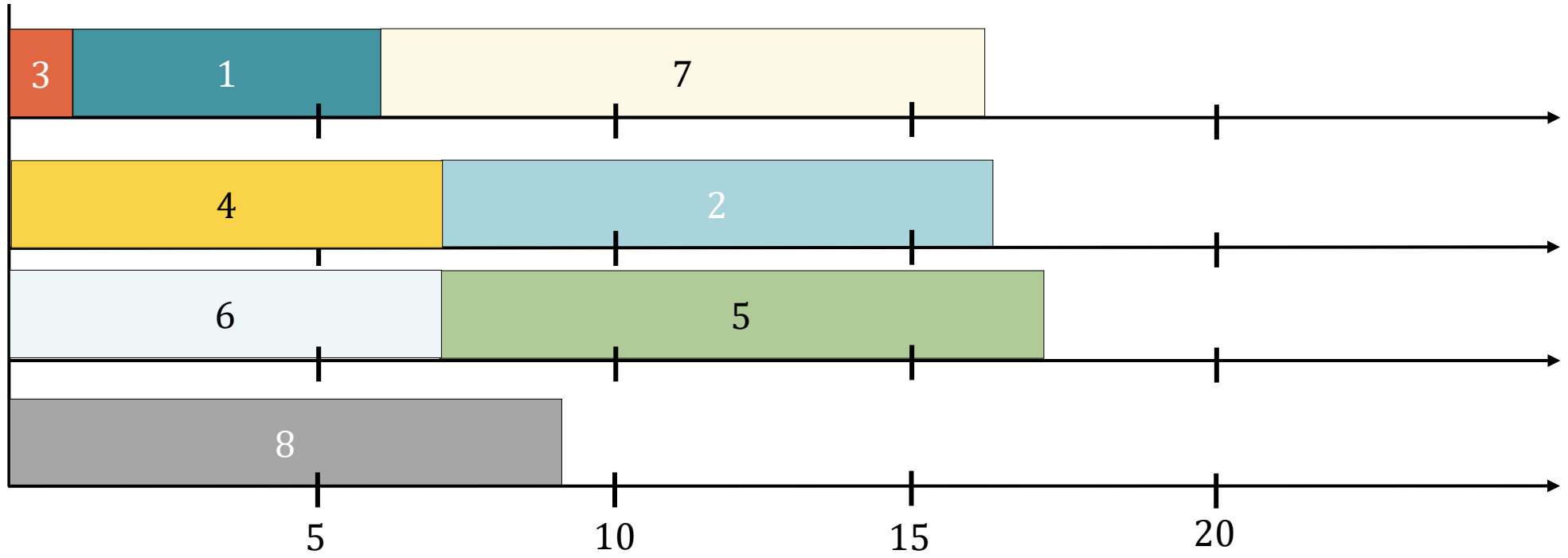
Generare una soluzione euristica *x*
x' := *x* (*x'* := soluzione incumbent); *x** := *x'* (*x** := miglior soluzione trovata);
improved := true
while (*improved*) **do**
    improved := false
    Forall(tutti i lavori) **do**
        c(x_swap) := 0
        Selezionare due job diversi tra loro ed effettuare lo SWAP
        Aggiornare gli istanti di completamento di ciascun job su ogni macchina
        Verificare che i job non abbiano sforato *T* → se sì, **break** (ritornare all'inizio del **while**)
        Aggiornare il valore della soluzione obiettivo *x_swap*
        if(*c(x_swap)* < *c(x')*) **then** *x'* := *x_swap*
    end-for
    if (*c(x')* < *c(x*)*)
        *x** := *x'*
        improved := true
        Effettuare nuovamente lo SWAP a partire dalla soluzione ottima trovata
    end-if
end-while

Esempio swap

- $n = 8, m = 4$
- $p_j = \{5, 9, 1, 7, 10, 7, 10, 9\}$
- $w_j = \{15, 4, 39, 39, 2, 30, 28, 29\}$

$$z = \sum_{j=1}^{8} w_j C_j = \mathbf{1419}$$

$ordine = [3, 4, 6, 8, 1, 7, 2, 5]$



Esempio swap

- $n = 8, m = 4$
- $p_j = \{5, 9, 1, 7, 10, 7, 10, 9\}$
- $w_j = \{15, 4, 39, 39, 2, 30, 28, 29\}$

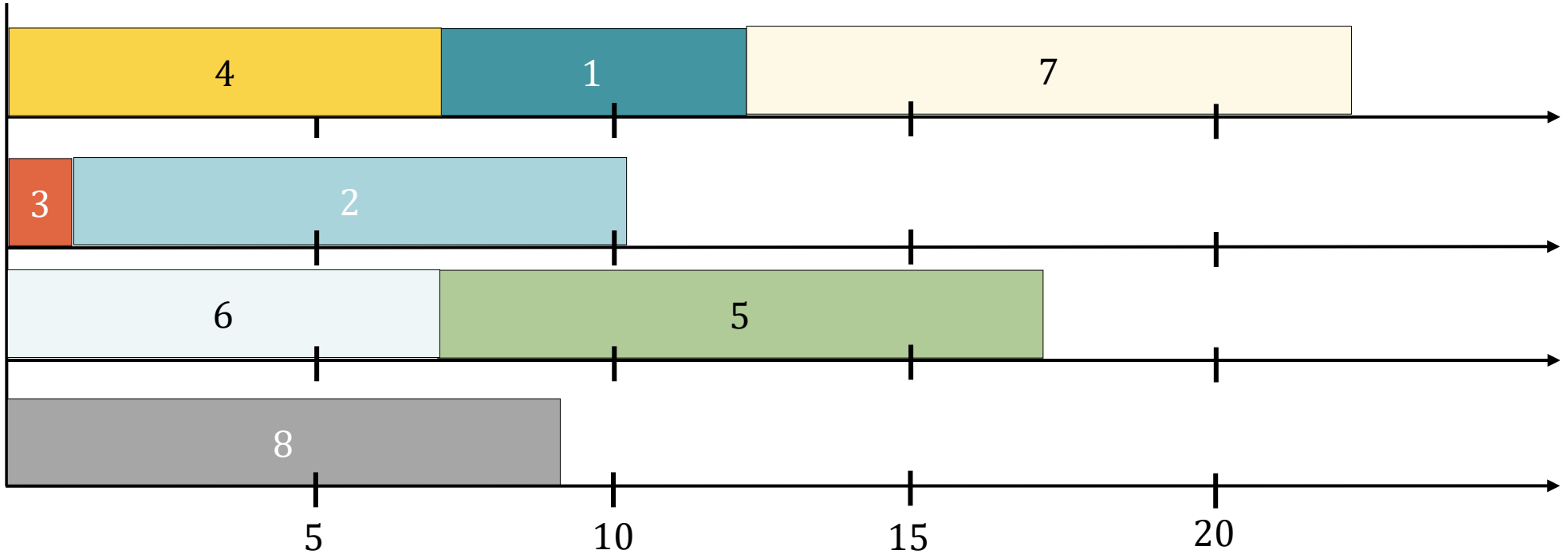
$$z = \sum_{j=1}^{8} w_j C_j = \mathbf{1419}$$

$ordine = [3, 4, 6, 8, 1, 7, 2, 5]$

SWAP

$$z = \sum_{j=1}^{8} w_j C_j = \mathbf{1653}$$

$ordine = [4, 3, 6, 8, 1, 7, 2, 5]$

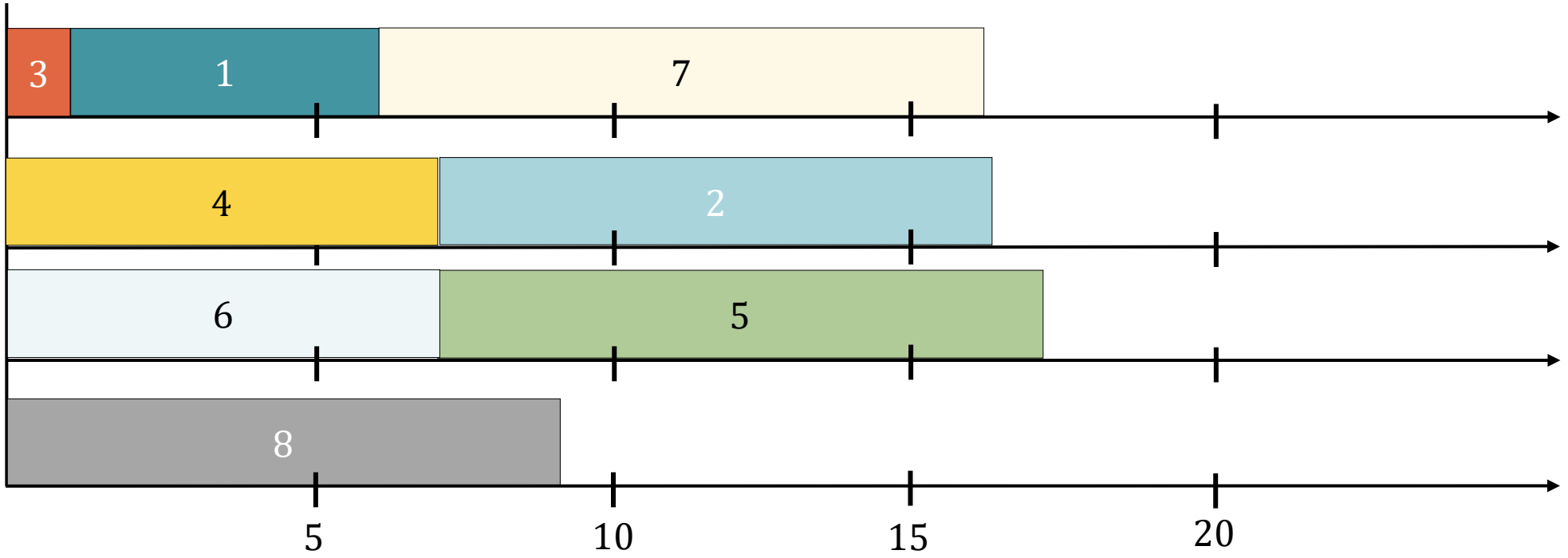


Esempio swap

- $n = 8, m = 4$
- $p_j = \{5, 9, 1, 7, 10, 7, 10, 9\}$
- $w_j = \{15, 4, 39, 39, 2, 30, 28, 29\}$

$$z = \sum_{j=1}^{8} w_j C_j = \mathbf{1419}$$

$ordine = [3, 4, 6, 8, 1, 7, 2, 5]$



Esempio swap

- $n = 8, m = 4$
- $p_j = \{5, 9, 1, 7, 10, 7, 10, 9\}$
- $w_j = \{15, 4, 39, 39, 2, 30, 28, 29\}$

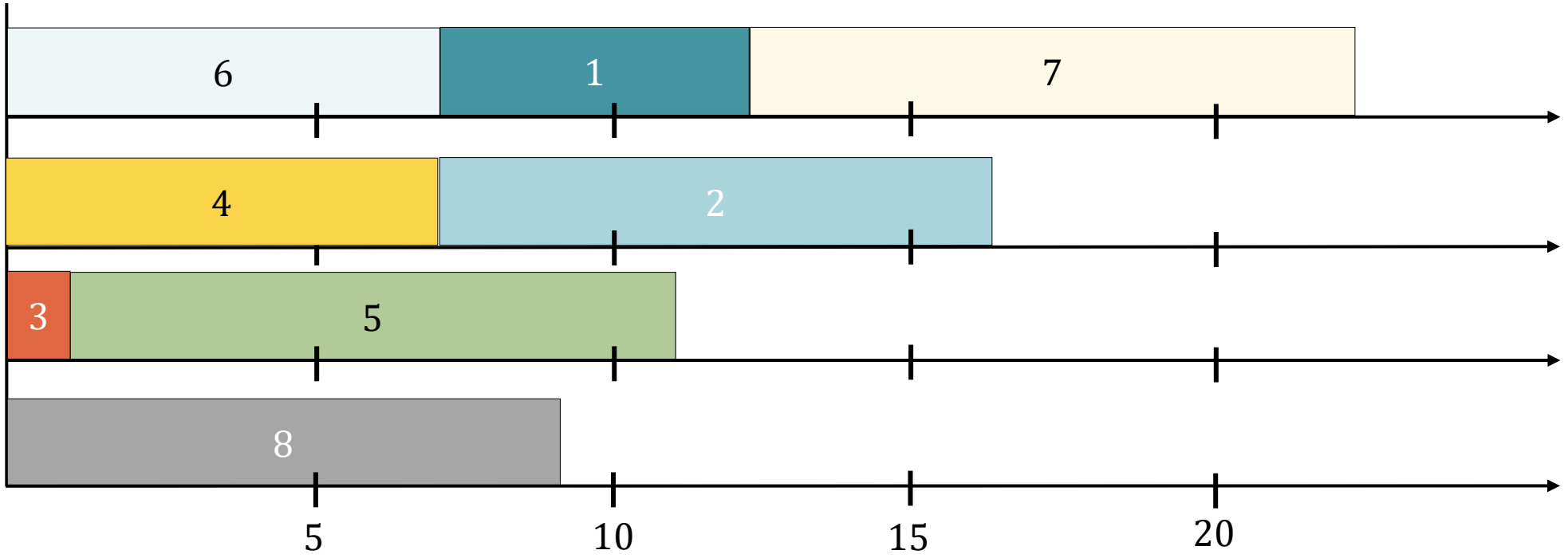
$$z = \sum_{j=1}^{8} w_j C_j = \mathbf{1419}$$

$ordine = [3, 4, 6, 8, 1, 7, 2, 5]$

SWAP

$$z = \sum_{j=1}^{8} w_j C_j = \mathbf{1665}$$

$ordine = [6, 4, 3, 8, 1, 7, 2, 5]$



Multi-start

- Per esplorare al meglio lo spazio delle soluzioni, si è implementata una procedura Multi-Start
- Il Multi-Start esegue più volte l'euristico precedentemente descritto, sostituendo al classico ordinamento WSPT un ordinamento casuale guidato
- Ordinamento casuale guidato implementato andando a generare dei rapporti w_j/p_j randomici nel seguente modo:

$$\frac{w_j}{p_j}_{random} = \left(\frac{w_j}{p_j} \times 0.7\right) + \left(random \times \frac{w_j}{p_j} \times 0.4\right) \; \forall j \in J$$

- Numero di iterazioni = 50
- Ogni soluzione generata dal Multi-Start viene ulteriormente migliorata attraverso una procedura di ricerca locale SWAP best-improvement

Simulated Annealing

Descrizione

- Procedura che permette l'esplorazione degli intorni di molte soluzioni attraverso un'ottica first-improvement
- Accetta occasionalmente soluzioni peggiorative, alla ricerca di eventuali miglioramenti di lungo termine
- Simula il fenomeno dell'annichilimento della materia: all'abbassarsi della temperatura, è meno probabile accettare soluzioni peggiorative → si passa, gradualmente, da una ricerca estensiva ad una ricerca intensiva

Simulated Annealing

Parametri di inizializzazione

Sono stati inizializzati, tramite analisi empiriche, i seguenti dati in input:

- Numero di iterazioni in cui t rimane costante $\Delta_k = 2000$
- Costante moltiplicativa $\alpha = 0.9$
- Temperatura iniziale $t_0 = z\_multi_{worst} - z\_multi_{best}$
- Temperatura critica $t_{critica}$ calcolata come la minima differenza tra tutte le soluzioni trovate dal Multi-start

Criteri di stop:

- Numero di iterazioni massime senza spostamento $iter_{max} = n \cdot 50$
- Numero massimo di iterazioni senza miglioramento $no\_miglioramento_{max} = 10000 \cdot n$ (usato come ulteriore criterio di interruzione)

Simulated Annealing

Pseudocodice

Generare una soluzione euristica x
t:= t_0; *iter* := *0; no_miglioramento* := *0;* Δ_k := 2000; $iter_{max}$: = $n \cdot 50$*;* k:= 0;
$no\_miglioramento_{max}$: = $10000 \cdot n$; x^*:= x;
while *(iter* < $iter_{max}$) **do**
    no_miglioramento := *no_miglioramento* + 1;
    Generare una soluzione $x' \in N(x)$ (trovato con swap random);
    If (*c(x')* < *c(x)*) **then**
        x:= x'; *iter*:= 0;
        if (*c(x)* < *c(x*)*) **then**
            x^*:= x; *no_miglioramento* := 0
        end-if
    elif (*c(x')* > *c(x)*) **then**
        Generare probabilità random $prob \in [0,1]$;
        if $\left(prob \leq e^{-\left(\frac{c(x')-c(x)}{t}\right)}\right)$ **then**
            x:= x'; *iter*:= 0;
        else *iter*:= *iter* + 1;
        end-if
    end-if
    k:= k+1;
    if (k:= Δ_k **and** $t > t_{critica}$) **then**
        $t := \alpha \cdot t$; k:= 0;
    end-if
    if ($no\_miglioramento \geq no\_miglioramento_{max}$) **then break***;*
    end-if
end-while

Risultati computazionali

Descrizione delle istanze

- Generate 80 istanze attraverso un generatore di istanze sviluppato con Python
- Istanze generate seguendo diversi criteri*
- Istanze differenziate attraverso la combinazione dei seguenti elementi:
  - Massimo tempo di processamento ≤ [20, 50, 75, 100]
  - Numero di job = [20, 50, 75, 100]
  - Numero di macchine = [3, 5, 8, 10, 12]

*Bülbül, K. , & ¸S en, H. (2017). An exact extended formulation for the unrelated parallel machine total weighted completion time problem. Journal of Scheduling, 20 (4), 373–389 .
Kowalczyk, D. , & Leus, R. (2018). A branch-and-price algorithm for parallel machine scheduling using ZDDs and generic branching. INFORMS Journal on Computing, 30 (4), 768–782 .

Risultati computazionali

Istanze con tempi di processamento ≤ 20

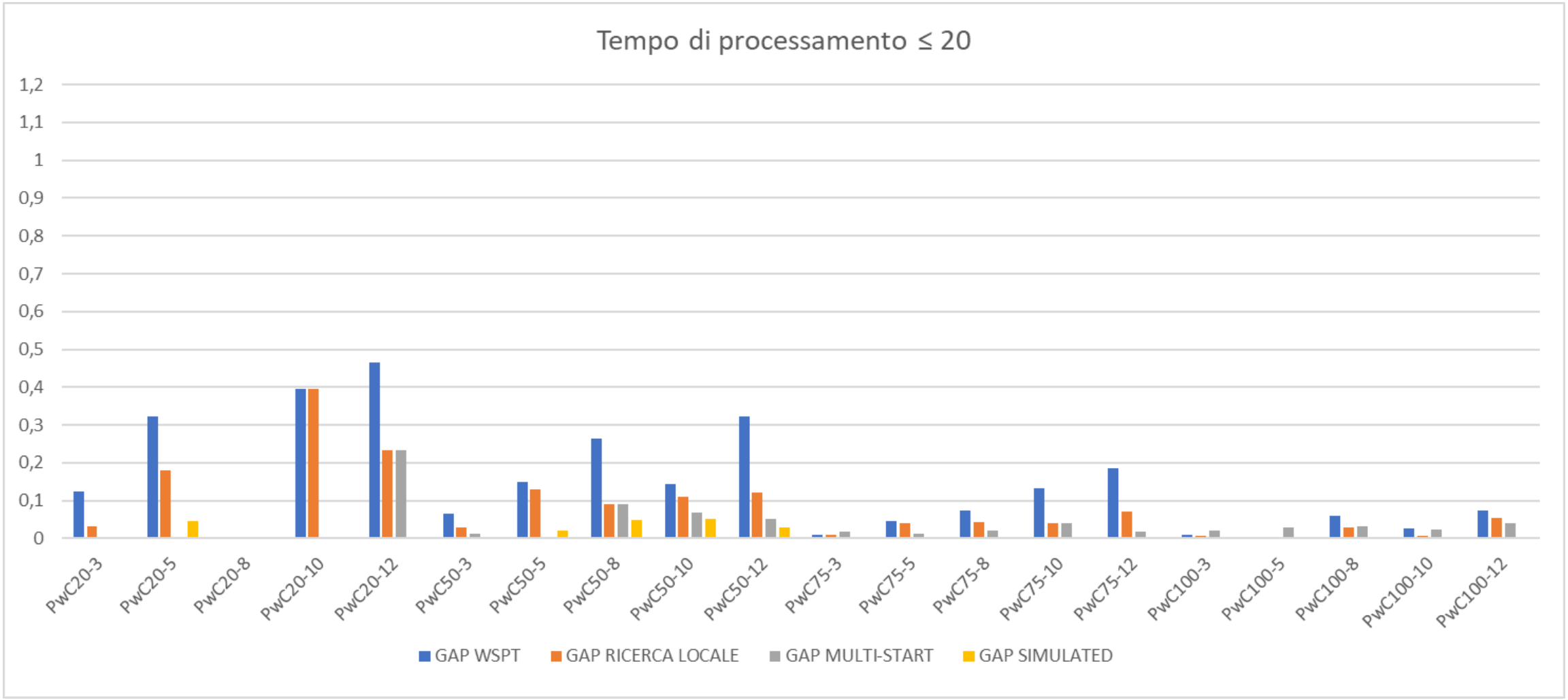
ISTANZA	WSPT	GAP WSPT	RICERCA LOCALE	GAP RICERCA LOCALE	MULTI-START	GAP MULTI-START	SIMULATED	GAP SIMULATED	MODELLO
PwC20-3	6488	0,123	6482	0,031	6480	0	6480	0	**6480**
PwC20-5	4504	0,977	4468	0,179	4460	0	4462	0,045	**4460**
PwC50-3	22964	0,065	22956	0,030	22952	0,013	**22949**	**0**	x
PwC50-5	14630	0,150	14627	0,130	**14608**	**0**	14611	0,021	x
PwC75-3	68745	0,010	68745	0,010	68750	0,017	**68738**	**0**	x
PwC75-5	44718	0,045	44716	0,040	44704	0,013	**44698**	**0**	x
PwC100-3	96664	0,010	96660	0,006	96673	0,020	**96654**	**0**	x
PwC100-5	74463	0,001	**74462**	**0**	74484	0,030	74463	0,001	x

Risultati computazionali

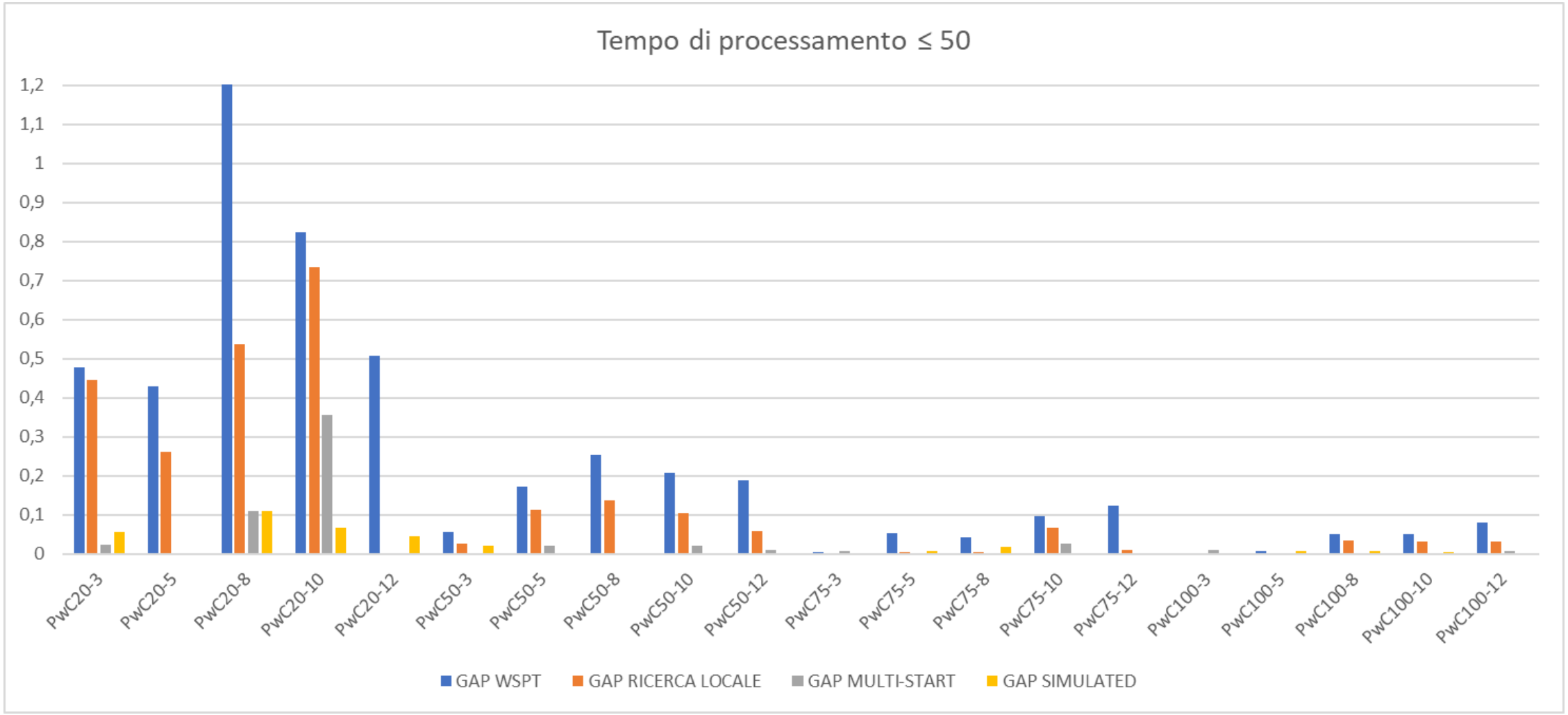
Istanze con tempi di processamento ≤ 100

ISTANZA	WSPT	GAP WSPT	RICERCA LOCALE	GAP RICERCA LOCALE	MULTI-START	GAP MULTI-START	SIMULATED	GAP SIMULATED	MODELLO
PwC20-10	12737	0,847	12666	0,292	12631	0,018	12629	0	**12629**
PwC20-12	11743	0,221	11722	0,042	11717	0	11722	0,042	**11717**
PwC50-10	53648	0,221	53594	0,121	53531	0,003	**53529**	**0**	x
PwC50-12	45007	0,344	44949	0,215	44873	0,046	**44852**	**0**	x
PwC75-10	94897	0,138	94802	0,037	**94766**	**0**	94835	0,072	x
PwC75-12	87610	0,262	87513	0,151	87445	0,074	**87380**	**0**	x
PwC100-10	174235	0,052	174159	0,009	**174143**	**0**	174199	0,032	x
PwC100-12	132083	0,132	132027	0,090	**131908**	**0**	131941	0,025	x

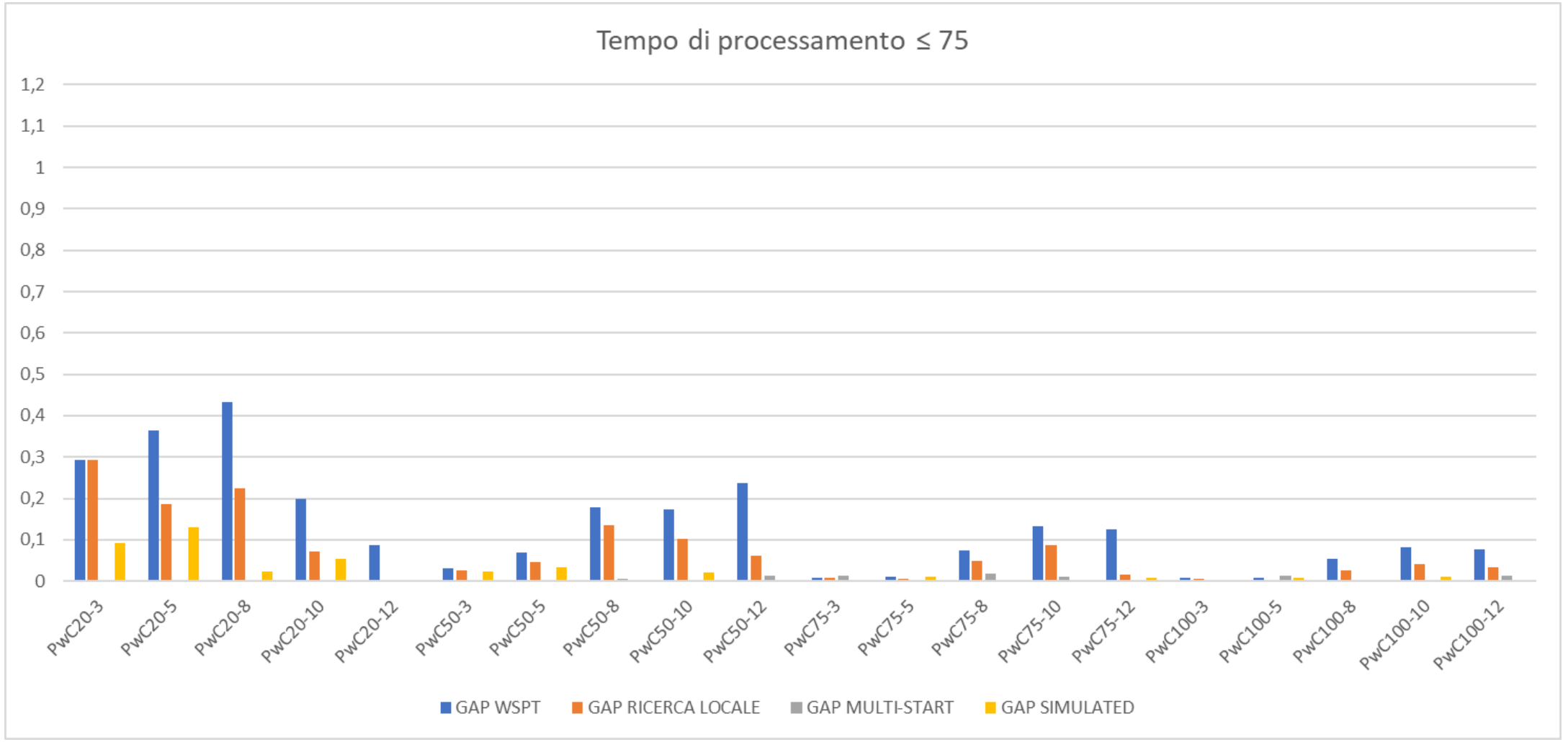
Risultati computazionali - GAP



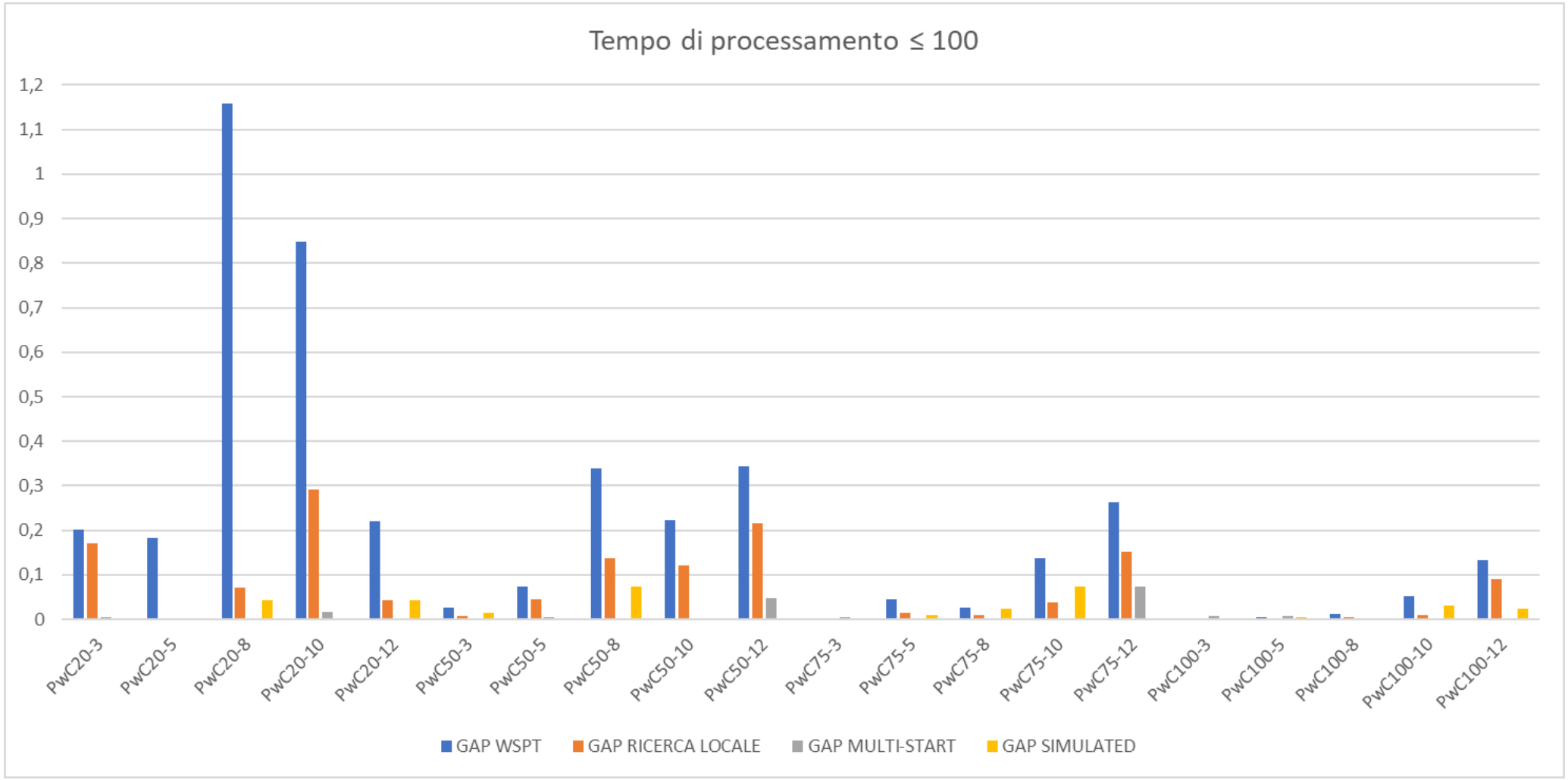
Risultati computazionali - GAP



Risultati computazionali - GAP

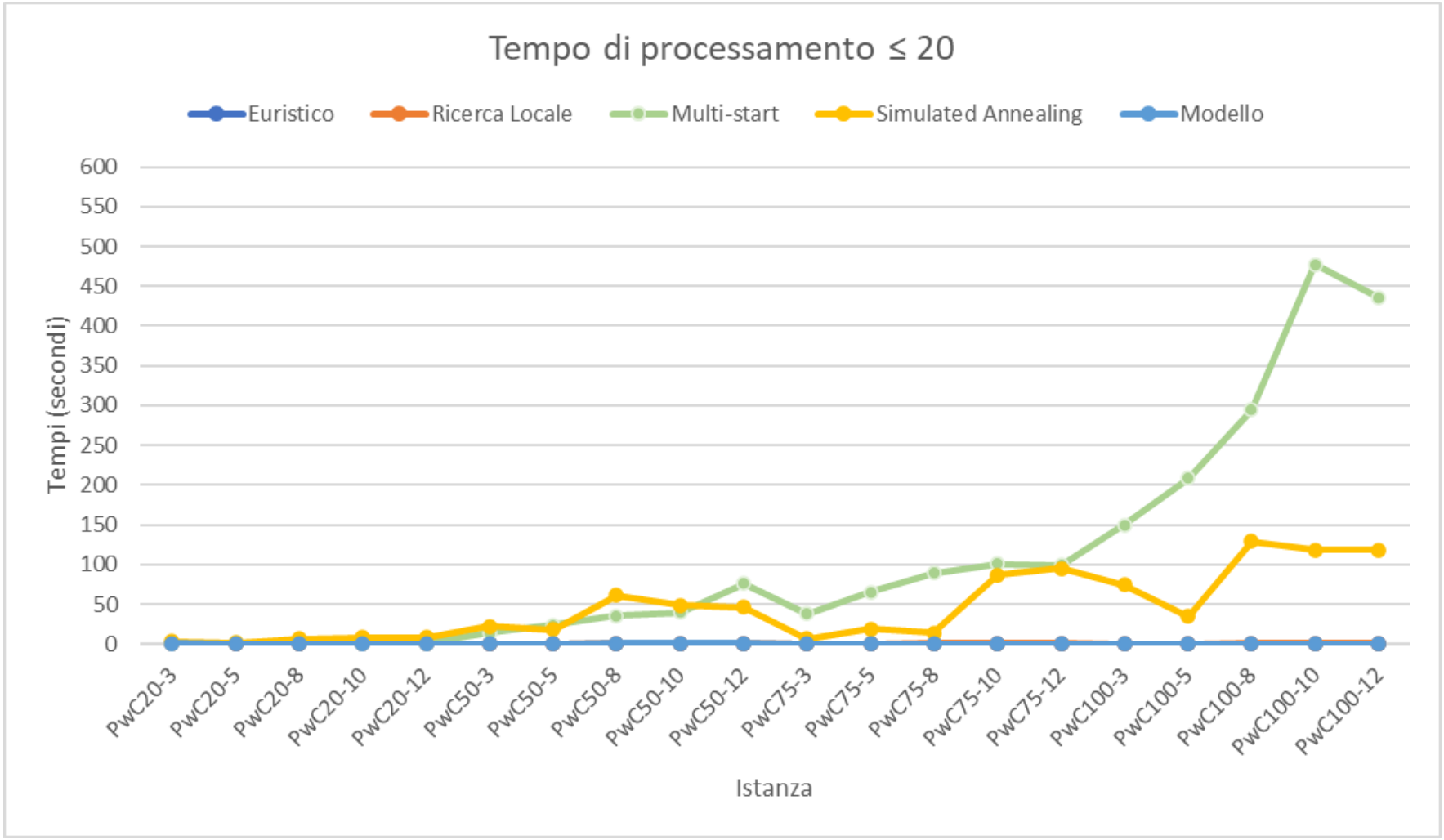


Risultati computazionali - GAP



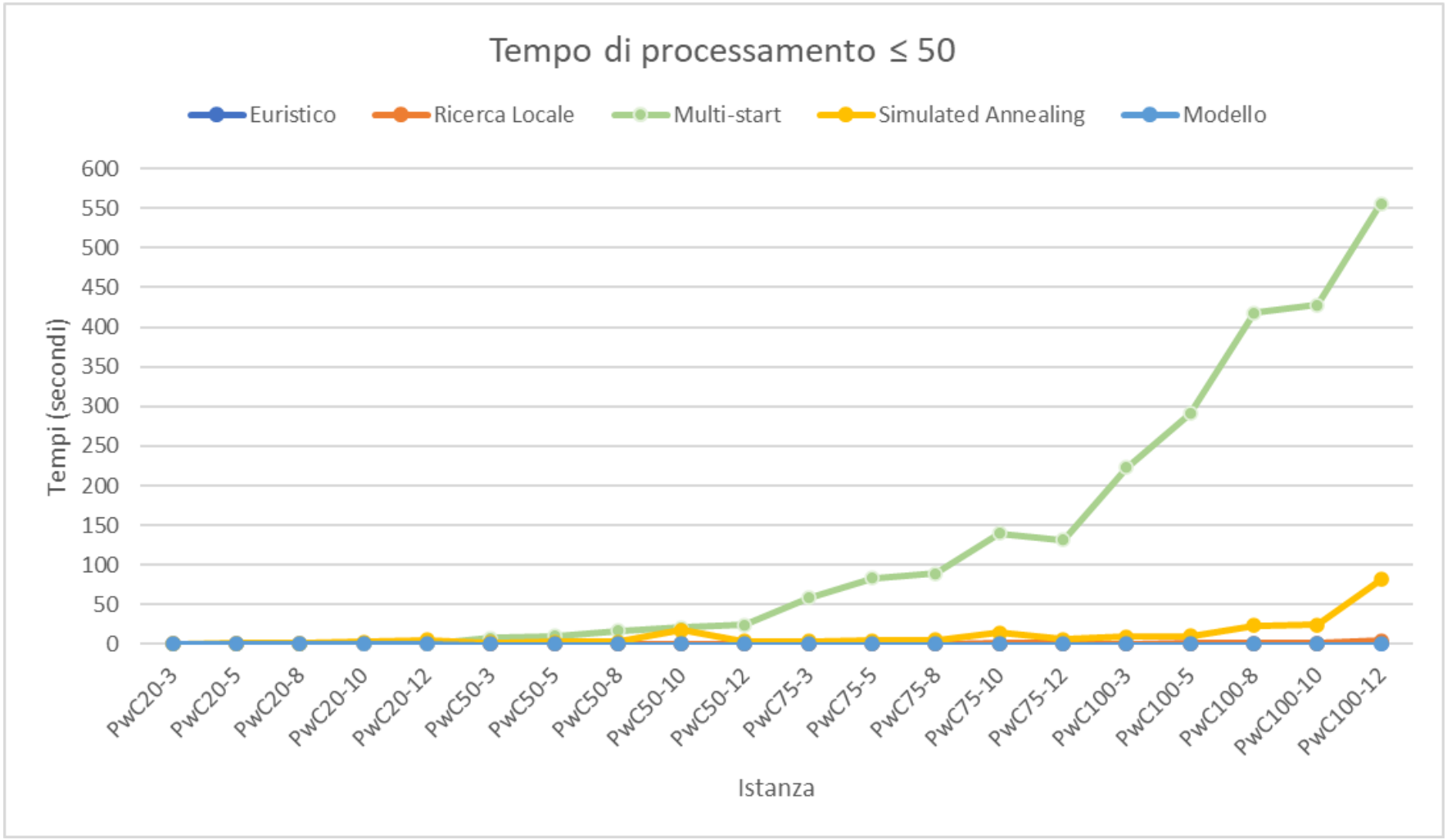
Tempi computazionali

Test computazionali eseguiti su Intel i5-8265U 1.6GHz – 16 GB di RAM – Windows 10 64 bit



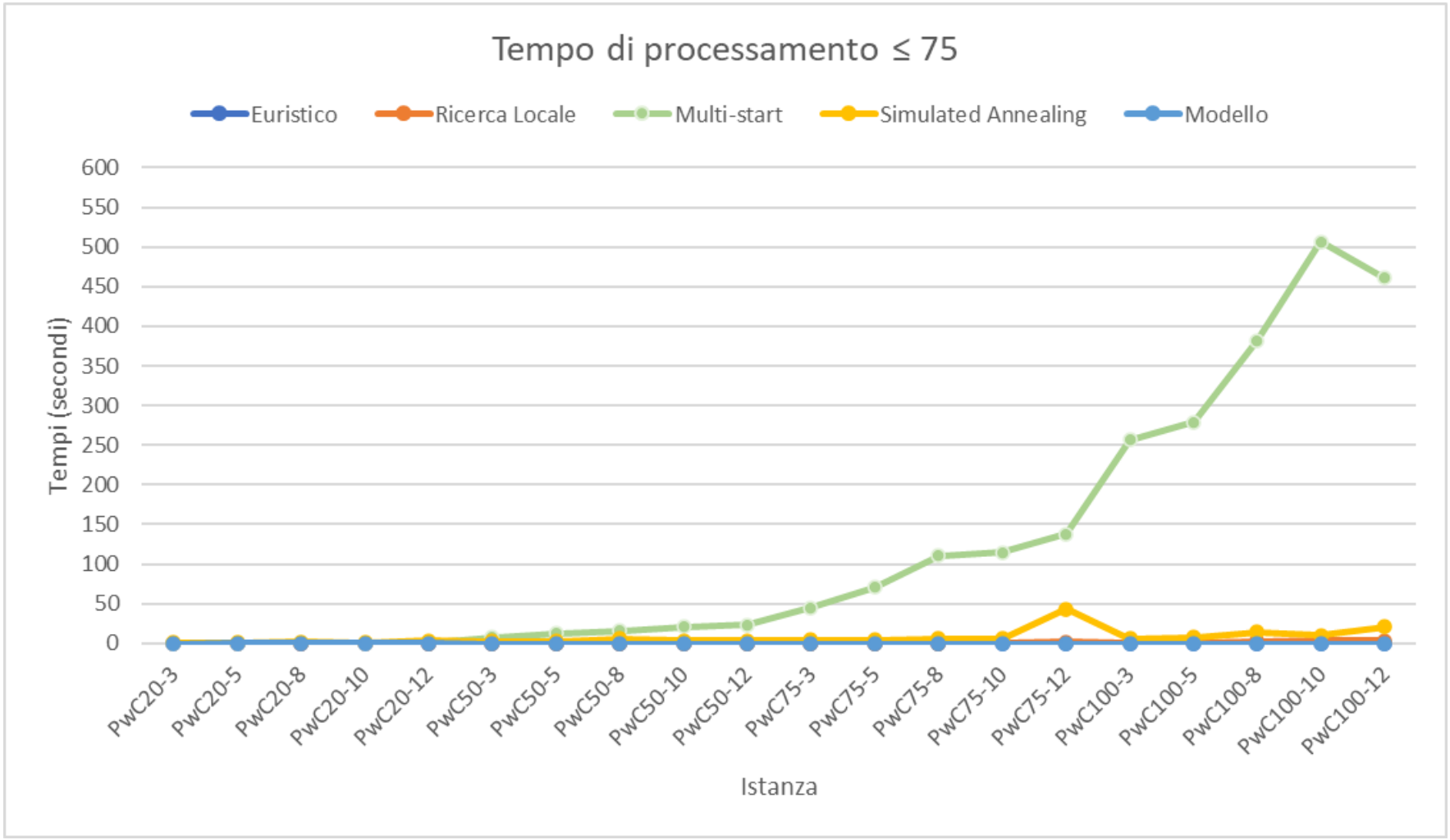
Tempi computazionali

Test computazionali eseguiti su Intel i5-8265U 1.6GHz – 16 GB di RAM – Windows 10 64 bit



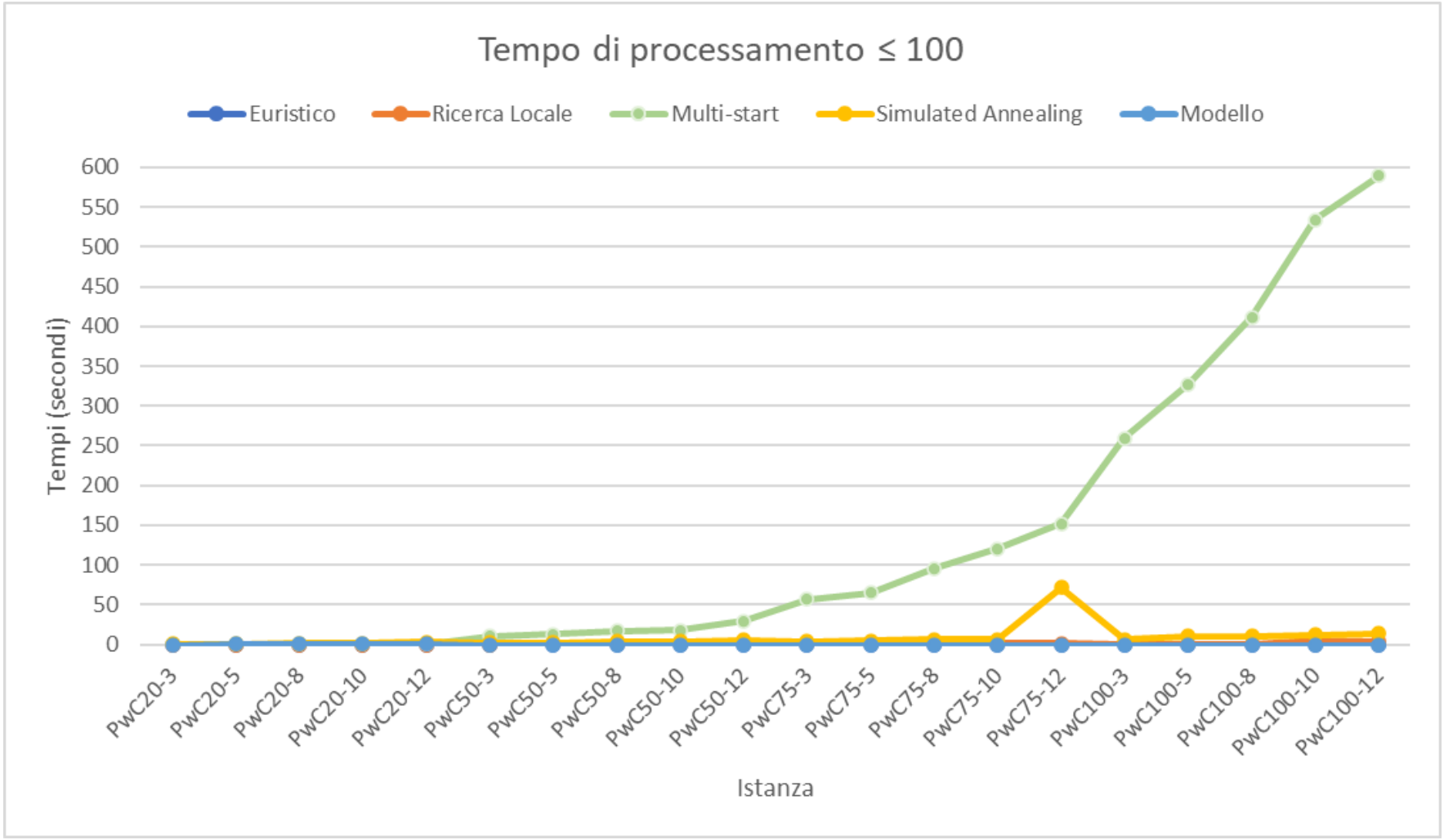
Tempi computazionali

Test computazionali eseguiti su Intel i5-8265U 1.6GHz – 16 GB di RAM – Windows 10 64 bit



Tempi computazionali

Test computazionali eseguiti su Intel i5-8265U 1.6GHz – 16 GB di RAM – Windows 10 64 bit



Conclusioni

- I GAP dei vari algoritmi sono, in valore assoluto, molto bassi (inferiori all'1%)
- Multi-start + SWAP e Simulated Annealing ottengono i GAP migliori
- I tempi di esecuzione del Multi-Start tendono ad avere una crescita esponenziale all'aumentare del numero di job
- Per istanze grandi, il Simulated Annealing ha il rapporto tempo-prestazioni più conveniente, poiché permette di trovare soluzioni paragonabili/migliori del Multi-Start con un risparmio di tempo considerevole

Grazie per l’attenzione

Gruppo 18

Bertè Sonia
Lochi Lorenza Maria
Sawan Omar
Verdi Federico