Ricerca Operativa
Homework 3

Maddalena Ghiotti, Aldo Sambo

July 2022

1 Introduzione

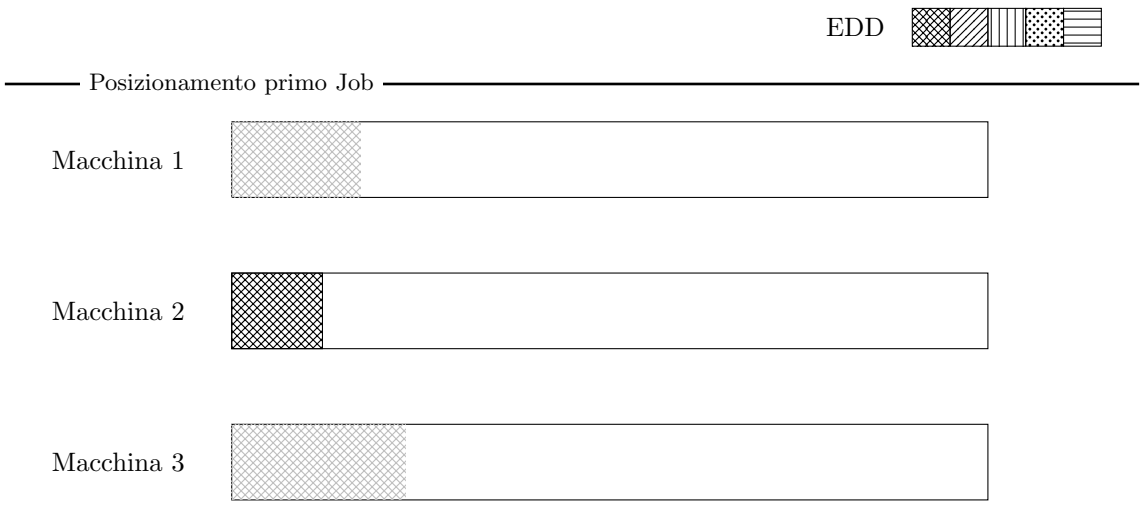
L'obiettivo del problema è minimizzare la lateness massima L_{max} su macchina multipla, con tempi di setup dipendenti dalla sequenza e macchine potenzialmente diverse.
Sono proposte di seguito un'euristica costruttiva, tre euristiche iterative e diverse euristiche miste. Analogamente a quanto fatto in HW2, gli algoritmi verranno poi confrontati tra di loro al fine di stabilire il miglior compromesso efficacia-efficienza.
L'insieme dei job assegnati ad ogni macchina e il loro ordine di esecuzione sono rappresentati nel codice Matlab da una matrice $m \times n$, dove m è il numero delle macchine ed n il numero dei job. Denotiamo d'ora in avanti tale matrice con il termine *configurazione*. Ogni riga contiene, nell'ordine, i numeri dei job assegnati alla macchina in questione e poi uno zero in ogni posizione vuota successiva. Denotiamo l'insieme ordinato degli elementi non nulli della riga con il termine *sequenza*.

2 Euristica costruttiva

Si propone un'unica euristica costruttiva, che tiene conto sia delle date di consegna, sia dei tempi di setup, sia dei processing time. Tale euristica, che denotiamo con EDD:

- Crea un vettore di job disposti secondo l'ordine crescente delle loro date di consegna (vedi euristica costruttiva EDD di HW2).
- Seleziona da questo vettore, a partire dal primo elemento, un job alla volta e lo assegna alla macchina in cui esso, inserito dopo i job (eventualmente) già presenti su quella macchina, risulta avere tempo di completamento minore.

Il funzionamento di questa euristica si capisce facilmente dal disegno in fig. 1. In grigio, sono rappresentati i possibili inserimenti del job in questione nelle $m = 3$ macchine considerate. In nero, poi, è stato ripassato solamente l'inserimento migliore, in termini di tempi di completamento.



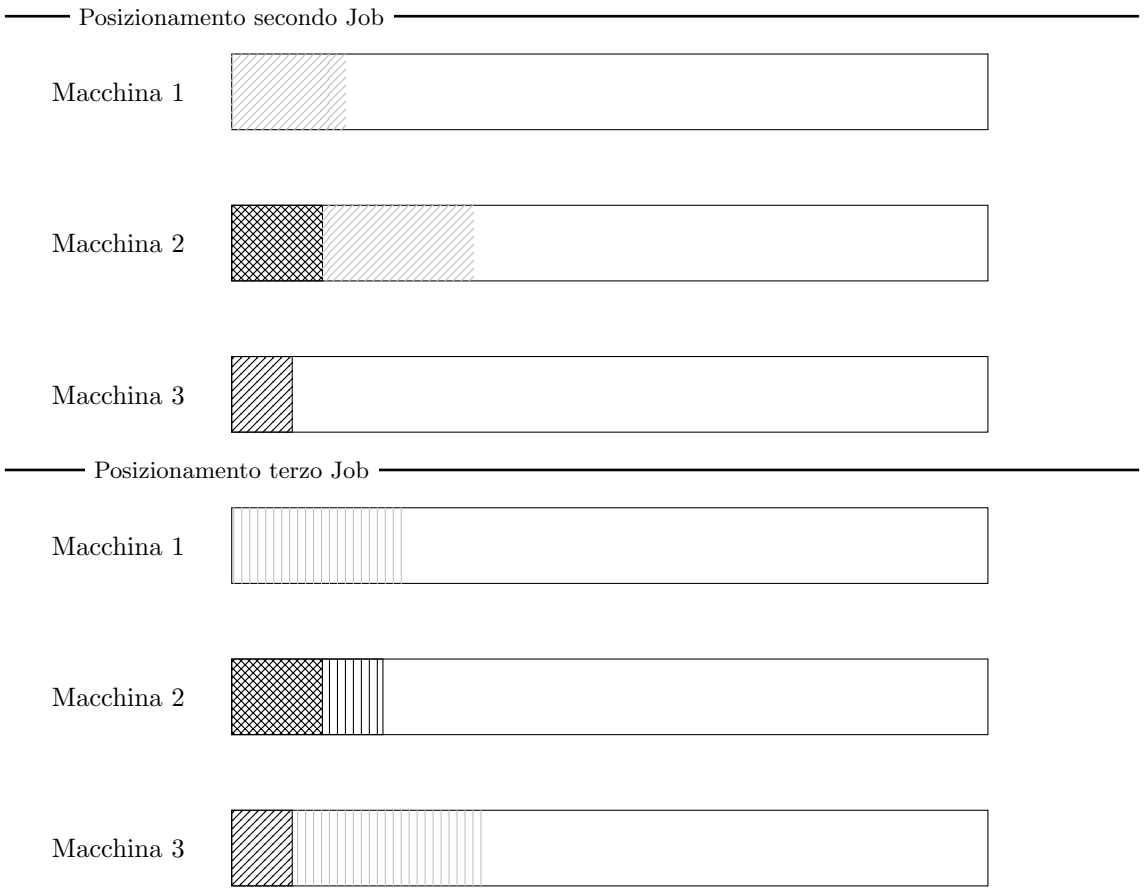


fig. 1

3 Euristiche iterative

Si propongono di seguito le tre euristiche da noi pensate per esplorare localmente lo spazio delle soluzioni, a partire da una configurazione data.
Ognuna di esse, in prima fase, si occupa di definire un vicinato, ossia un intorno di soluzioni centrato nella configurazione data. In questo caso, si noti che il vicinato è un insieme di configurazioni e non di sequenze, come accade in HW2. Questa prima fase è quella che vedete descritta, per ciascuna euristica, a seguire.
Tra le soluzioni offerte dal vicinato viene poi selezionata esclusivamente quella con lateness massima minore e, se migliorante rispetto alla configurazione di partenza, al termine dell'iterazione, ciascun algoritmo la sceglie come nuova configurazione corrente e riparte da essa con l'esplorazione del nuovo vicinato. Se ad una certa iterazione non venisse trovata, nell'intorno della configurazione corrente, alcuna soluzione migliorante, ciò signfica che la configurazione corrente è un minimo locale, e l'algoritmo termina.
Abbiamo quindi optato, come in HW2, per metodi euristici di Ricerca Locale di tipo best-improvement.

3.1 Algoritmo latenessMax

Sia data una configurazione iniziale, eventualmente randomica. Questa euristica, inizialmente, individua il job avente lateness massima tra tutti i job nella configurazione, $JobL_{max}$ (evidenziato in fig. 2 tramite un asterisco). Successivamente, definisce il vicinato secondo i 3 criteri seguenti:

- Nella macchina in cui si trova $JobL_{max}$, sposta quest'ultimo in ciascuna posizione precedente alla sua attuale (applica, cioè, l'algoritmo latenessMax di HW2 alla macchina di $JobL_{max}$), lasciando nel frattempo le altre macchine inalterate. A ogni anticipazione genera una nuova soluzione.
- Per ogni macchina diversa da quella di $JobL_{max}$, inserisce quest'ultimo in ogni posizione fino a quella subito precedente il primo job (della macchina di arrivo) con tempo di completamento maggiore o uguale di quello di $JobL_{max}$. Qualora questo non esista, inserisce fino alla fine della sequenza

(fino a sostituire il primo 0). Lascia nel frattempo inalterate le macchine non coinvolte. A ogni inserimento genera una nuova soluzione.

- Per ogni macchina diversa da quella di $JobL_{max}$, scambia quest'ultimo con l'ultimo job (della macchina di arrivo) avente tempo di completamento strettamente inferiore a quello di $JobL_{max}$, lasciando nel frattempo inalterate le macchine non coinvolte. A ogni scambio genera una nuova soluzione.

Nel disegno fig. 2 è meglio illustrato il funzionamento dell'algoritmo. In alto, si vede un esempio di macchina j qualsiasi diversa dalla macchina contenente $JobL_{max}$, in cui sono evidenziati in grigio i job con tempo di completamento maggiore o uguale rispetto a quello di $JobL_{max}$.
Le frecce continue rappresentano gli spostamenti di $JobL_{max}$ nella macchina stessa (in basso) e nella macchina j (in alto); la freccia tratteggiata rappresenta invece lo scambio dei job sopra descritto.

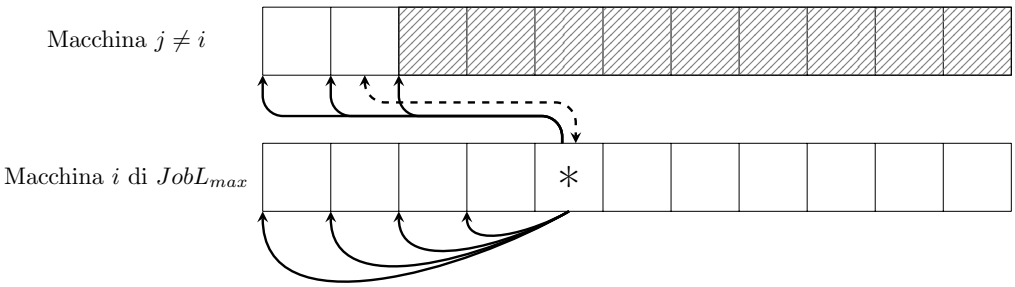


fig. 2

3.2 Algoritmo setupMax

Sia data una configurazione iniziale, eventualmente randomica. Come l'algoritmo latenessMax, l'euristica inizialmente individua il job con tempo di setup maggiore, $JobSetUp_{max}$ (evidenziato in fig. 3 tramite un asterisco), tra tutti quelli assegnati alle diverse macchine. L'esplorazione del vicinato segue i 4 criteri seguenti:

- Nella macchina in cui si trova $JobSetUp_{max}$, applica l'algoritmo setupMax di HW2, effettuando:
  - Scambio dei due job precedenti $JobSetUp_{max}$.
  - Scambio del job precedente e di $JobSetUp_{max}$.
  - Scambio di $JobSetUp_{max}$ e del job successivo.

  Lascia nel frattempo inalterate le altre macchine. A ogni scambio genera una nuova soluzione.
- Inserisce $JobSetUp_{max}$ in ogni altra macchina, in posizione subito precedente l'ultimo job con tempo di completamento strettamente inferiore rispetto a quello di $JobSetUp_{max}$. Lascia nel frattempo inalterate le macchine non coinvolte. A ogni inserimento genera una nuova soluzione.
- Inserisce $JobSetUp_{max}$ in ogni altra macchina, in posizione subito successiva l'ultimo job con tempo di completamento strettamente inferiore rispetto a quello di $JobSetUp_{max}$. Se non esiste un job con tempo di completamento maggiore o uguale, $JobSetUp_{max}$ è posto alla fine della sequenza (in sostituzione del primo 0). Lascia nel frattempo inalterate le macchine non coinvolte. A ogni inserimento genera una nuova soluzione.
- Scambia $JobSetUp_{max}$ con l'ultimo job avente tempo di completamento strettamente inferiore a quello di $JobSetUp_{max}$, nella sequenza di ogni altra macchina. Lascia nel frattempo inalterate le macchine non coinvolte. A ogni scambio genera una nuova soluzione.

Nel disegno fig. 3 è meglio illustrato il funzionamento dell'algoritmo. In alto, si vede un esempio di macchina j qualsiasi diversa dalla macchina contenente $JobSetUp_{max}$, in cui sono evidenziati in grigio i job con tempo di completamento maggiore o uguale rispetto a quello di $JobSetUp_{max}$.
Le frecce continue rappresentano gli scambi nella macchina di $JobSetUp_{max}$ e quello tra macchine diverse sopra descritto; le frecce tratteggiate rappresentano invece gli inserimenti di $JobSetUp_{max}$ in altre macchine.

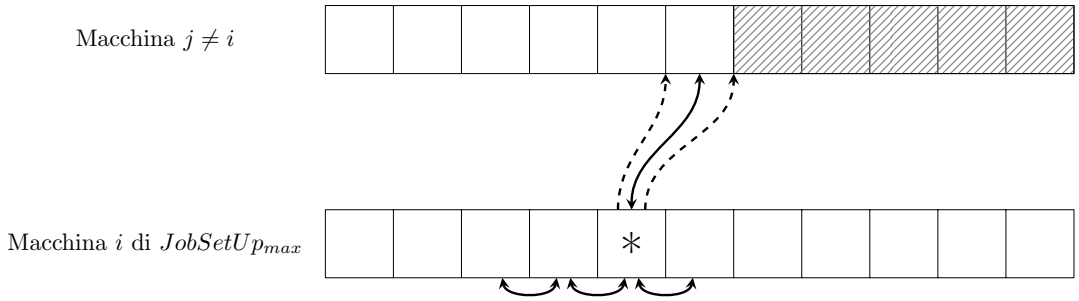
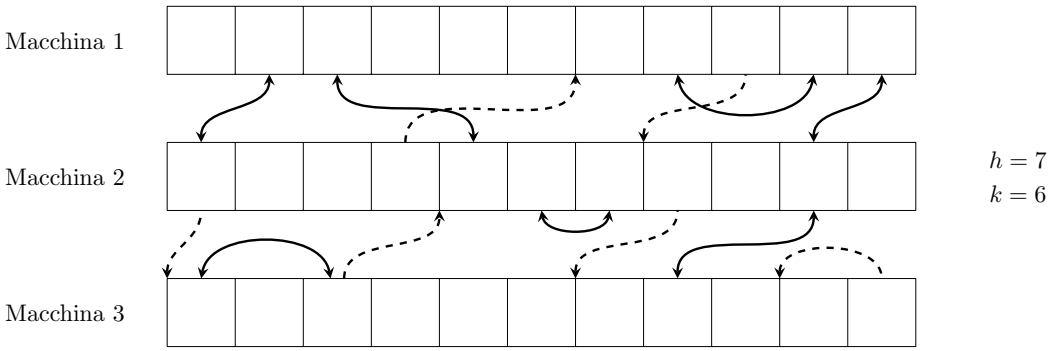


fig. 3

3.3 Algoritmo Rand(*h,k*)

L'euristica iterativa Rand, a partire da una configurazione eventualmente randomica, esplora il vicinato secondo i criteri:

- h scambi di due job casuali della configurazione, lasciando nel frattempo inalterate le macchine non coinvolte. Ogni scambio genera una nuova soluzione.
- k spostamenti di un job casuale della configurazione in una posizione casuale diversa da quella di partenza, tenendo in considerazione come possibili posizioni anche: prima di tutti i job della sequenza e alla fine di tutta la sequenza. Lascia nel frattempo inalterate le macchine non coinvolte. Ogni spostamento genera una nuova soluzione.



3.4 Approccio Multistart

Come per HW2 è possibile applicare ogni euristica iterativa a più configurazioni di partenza, nel nostro caso randomiche, e considerare come soluzione finale quella a cui corrisponde la lateness massima minore tra quelle ottenute. In questo modo, aumenta la probabilità che tra le configurazioni di partenza ve ne sia una più promettente per l'applicazione dell'algoritmo specifico.

4 Euristiche miste

Proponiamo le sette euristiche miste viste in HW2 anche nel caso di macchina multipla. Sono riportati qui di seguito gli algoritmi dettagliati.

4.1 Algoritmo EDD + latenessMax

A partire dalla soluzione generata tramite l'euristica costruttiva EDD, applica l'euristica iterativa latenessMax.

4.2 Algoritmo EDD + setupMax

A partire dalla soluzione generata tramite l'euristica costruttiva EDD, applica l'euristica iterativa setupMax.

4.3 Algoritmo EDD + Rand(*h,k*)

A partire dalla soluzione generata tramite l'euristica costruttiva EDD, applica l'euristica iterativa Rand(h,k).

4.4 Algoritmo EDD + (setupMax + latenessMax)

A partire dalla soluzione generata tramite l'euristica costruttiva EDD:

1. Applica l'euristica iterativa setupMax (**3.2**)
2. Applica l'euristica iterativa latenessMax (**3.1**) alla soluzione restituita da 1.
3. Se la soluzione restituita da 2 è migliore di quella ottenuta da 1, poni come soluzione corrente quella restituita da 2 e torna allo step 1. Altrimenti, termina.

4.5 Algoritmo EDD + (latenessMax + setupMax)

A partire dalla soluzione generata tramite l'euristica costruttiva EDD:

1. Applica l'euristica iterativa latenessMax (**3.1**)
2. Applica l'euristica iterativa setupMax (**3.2**) alla soluzione restituita da 1.
3. Se la soluzione restituita da 2 è migliore di quella ottenuta da 1, poni come soluzione corrente quella restituita da 2 e torna allo step 1. Altrimenti, termina.

4.6 Algoritmo EDD + (setupMax + latenessMax + Rand(*h,k*))

A partire dalla soluzione generata tramite l'euristica costruttiva EDD:

1. Applica l'euristica iterativa setupMax (**3.2**)
2. Applica l'euristica iterativa latenessMax (**3.1**) alla soluzione restituita da 1.
3. Applica l'euristica iterativa Rand(h,k) (**3.3**) alla soluzione restituita da 2.
4. Se la soluzione restituita da 3 è migliore di quella ottenuta da 1, poni come soluzione corrente quella restituita da 3 e torna allo step 1. Altrimenti, termina.

4.7 Algoritmo EDD + (setupMax + latenessMax) + Rand(*h,k*)

A partire dalla soluzione generata tramite l'euristica costruttiva EDD:

1. Applica l'euristica iterativa setupMax (**3.2**)
2. Applica l'euristica iterativa latenessMax (**3.1**) alla soluzione restituita da 1.
3. Se la soluzione restituita da 2 è migliore di quella ottenuta da 1, poni come soluzione corrente quella restituita da 2 e torna allo step 1. Altrimenti, vai a step 4.
4. Applica l'euristica iterativa Rand(h,k) (**3.3**) alla soluzione restituita da 2.
5. Se la soluzione restituita da 4 è migliore di quella ottenuta da 2, poni come soluzione corrente quella restituita da 4 e torna allo step 1. Altrimenti, termina.

5 Confronto tra gli algoritmi

5.1 Iterative

Confrontiamo tra loro le euristiche iterative sopra descritte, allo scopo di selezionare la migliore come rapporto efficacia-efficienza.

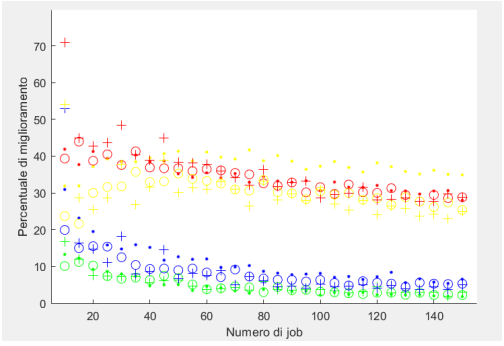


fig. 4

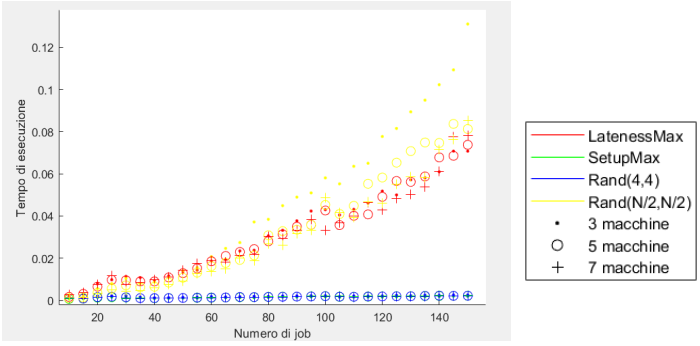


fig. 5

Come possiamo notare dai grafici, le euristiche più efficaci sono anche le meno efficienti e viceversa, e gli algoritmi sono separabili in due categorie: gli efficaci, LatenessMax (**3.1**) e Rand(N/2, N/2), e gli efficienti, SetupMax (**3.2**) e Rand(4,4), senza vie di mezzo. Un buon trade-off tempi-prestazioni è dunque difficile da identificare. Anche gli efficaci si possono tuttavia considerare algoritmi veloci, in termini assoluti, quindi probabilmente la scelta migliore ricade su LatenessMax, più costante e più conveniente per numeri bassi di job (vedi fig. 4) rispetto a Rand($N/2, N/2$). E' comunque interessante osservare la somiglianza tra gli andamenti di queste due euristiche, la quale fa intuire che le dimensioni del vicinato siano simili e che quindi il numero di nuove configurazioni esplorate da LatenessMax ad ogni iterazione sia circa pari a $N/2$, con N numero dei job.
Interessante anche osservare come la prestazione di tutti gli algoritmi cali nettamente all'aumentare del numero di job (vedi fig. 4), diversamente da quanto osservato in HW2, dove le euristiche che scalano con la dimensione del problema sembrano avere efficacia costante.
Per quanto riguarda il comportamento delle euristiche al variare del numero di macchine, possiamo subito notare che gli algoritmi Rand(h, k) (**3.3**) diminuiscono di efficacia e aumentano di efficienza con l'incremento del numero di macchine. Diversamente da Rand($N/2, N/2$), LatenessMax rimane più costante e, come anche per SetupMax, è difficile trovare una correlazione tra il numero di macchine e la percentuale di miglioramento o il tempo di esecuzione.

5.2 Iterative vs Costruttive+Iterative

Vediamo, al variare del numero di job, come si comportano le euristiche iterative a partire da una configurazione casuale rispetto alle stesse partendo dalla configurazione generata da EDD (**2**).

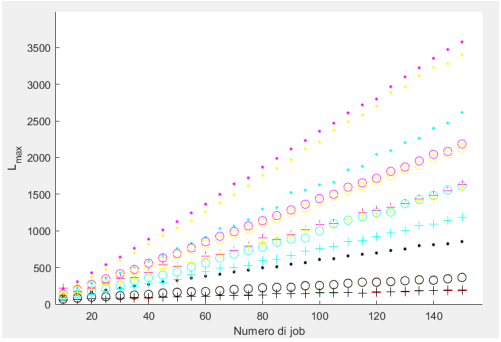


fig. 6

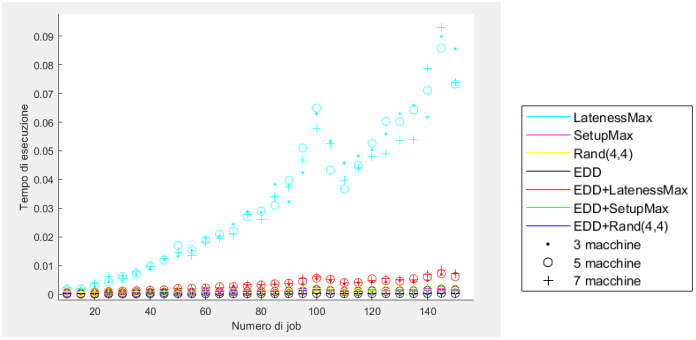


fig. 7

La sproporzione tra iterative semplici e già solo EDD rende inapprezzabili le migliorie delle iterative applicate a EDD, che risultano praticamente sovrapposte. Inoltre si può vedere come le tempistiche di algoritmi iterativi a partire da soluzioni casuali siano peggiori degli stessi con configurazione di partenza EDD. Come già osservato in HW2, dunque, le euristiche itertive diventano effettivamente convenienti, dal momento in cui si prende come configurazione di partenza il risultato di un'euristica costruttiva.

5.3 Miste

Qui di seguito confrontiamo efficacia ed efficienza delle euristiche miste, tramite rappresentazioni grafiche ed esempi numerici.

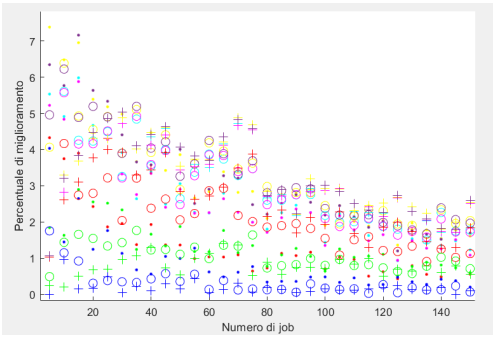


fig. 8

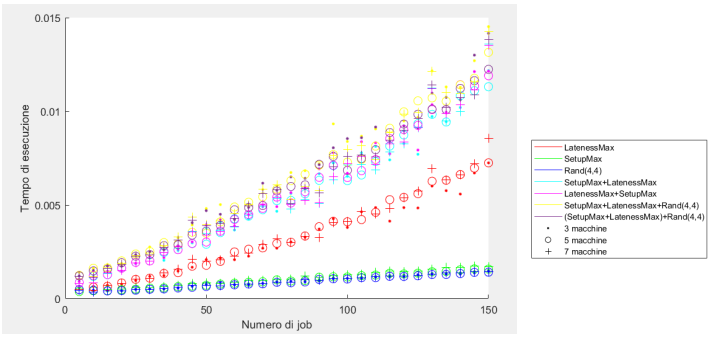


fig. 9

Analogamente al caso con macchina singola studiato in HW2, più sono le euristiche iterative utilizzate in uno stesso algoritmo, più questo è efficace, ma allo stesso tempo meno efficiente. Dunque, anche in questo caso la scelta dell'euristica migliore spetta al fruitore.
Una differenza che si può osservare rispetto alle euristiche miste su macchina singola trattate, è la convenienza, anche con sequenza di partenza EDD, dell'algoritmo LatenessMax (**4.1**) rispetto a SetupMax (**4.2**). Ciò non ci stupisce, dato che l'euristica costruttiva ideata per macchina multipla già tiene conto sia delle date di consegna dei job, che dei solo tempi di setup e processing time. In generale, l'efficacia di tutte le euristiche miste sembrerebbe diminuire all'aumentare del numero di job (vedi fig. 8), come abbiamo già visto accadere con le iterative (**5.1**). Dato che invece i tempi di esecuzione degli algoritmi hanno un andamento inverso (vedi fig. 7), finchè il tempo di esecuzione rimane limitato, cioè per un numero di job ridotto, potrebbe convenire l'utilizzo dell'algoritmo con la massima efficacia, EDD+(setupMax+latenessMax+Rand) (**4.6**) oppure EDD+(setupMax+latenessMax)+Rand (**4.7**). All'aumentare del numero di job, un buon compromesso potrebbe invece essere EDD+latenessMax (**4.1**). Gli andamenti delle prestazioni al variare del numero di macchine sono molto differenziati:

- L'efficacia di EDD+SetupMax ed EDD+Rand(4,4) sembra diminuire all'aumentare del numero di macchine;
- L'efficacia di EDD+LatenessMax sembra aumentare all'aumentare del numero di macchine;
- L'efficacia di EDD+(setupMax+latenessMax+Rand) e EDD+(setupMax+latenessMax)+Rand sembra diminuire all'aumentare del numero di macchine per un numero di job inferiore a 30 circa, mentre sembra aumentare per un numero di job superiore a 50 circa;

In generale, per un numero di job sufficientemente elevato, la differenza di efficacia al variare del numero di macchine diventa minima e non notiamo più alcuna correlazione tra queste due quantità.

Come in HW2, riportiamo in una tabella i valori espliciti per EDD ed ogni euristica mista. Nelle colonne troviamo la lateness massima media, il miglioramento percentuale medio rispetto ad EDD e il tempo medio di esecuzione dell'algoritmo, calcolati a partire da 10000 esperimenti con numero di job $n = 100$, numero di macchine $m = 5$ e numero di scambi e spostamenti dell'algoritmo Rand $h = 50, k = 50$.

Euristiche	L_{max}	Miglioramento (%)	Tempo (s)
EDD	257	-	0.000391247
EDD + latenessMax	253	1.56	0.00606033
EDD + setupMax	255	0.89	0.00176464
EDD + Rand	250	2.61	0.00735483
EDD + (setupMax + latenessMax)	251	2.37	0.0100074
EDD + (latenessMax + setupMax)	251	2.33	0.0102304
EDD + (setupMax + latenessMax + Rand)	244	5.06	0.025786
EDD + (setupMax + latenessMax) + Rand	244	4.94	0.02355

6 Conclusioni finali e considerazioni

Abbiamo cercato, nell'ideazione delle euristiche, di mantenere il più possibile una simmetria con HW2: ci sembrava comunque importante rimarcare alcuni concetti e riportare alcune procedure per completezza. Invece, abbiamo sorvolato su altri concetti già apertamente spiegati: da ciò ne consegue la lunghezza più ridotta del report.
Come per HW2, la parte più interessante del report è sicuramente il confronto tra le diverse euristiche e la loro analisi.