Treball de Fi de Màster

Màster Universitari en Enginyeria Industrial (MUEI)

Programació bicriteri de treballs en grups de màquines en paral·lel amb elegibilitat

ANNEXOS

Autor: Oriol Sentmartí

Director: Manel Mateo

Convocatòria: Maig 2018

Escola Tècnica Superior

d'Enginyeria Industrial de Barcelona

Sumari

Annex A. Exemplars i presentació dels resultats

En aquest annex s'explica, per una banda, com és un exemplar de 20 peces i per l'altra, s'explica l'estructura dels resultats obtinguts un com executat el programa.

A.1. Exemplars del problema

Per qualsevol dels exemplars l'estructura de les seves dades és la mateixa. Tot i així, a continuació s'explica el cas concret d'un exemplar de 20 peces, 3 nivells i la següent distribució de màquines (2,1,1).

En primer lloc apareix la informació comuna dels diferents exemplars; el nombre de nivells *k*, seguit del nombre de màquines de cada nivell m_k i el número d'exemplars a resoldre:

	3		➔ *g*, nombre de nivells
2	**1**	**1**	➔ m_k, nombre de màquines per nivell
	1000		➔ número d'exemplars a resoldre

Taula A.1 Primera introducció de dades del problema

En segon lloc s'introdueixen les dades especifiques de cada exemplar. Primer s'especifica el número d'exemplar del qual són les dades, el nombre de màquines que té cada grup i després s'introdueix n_k, el nombre total de peces en el següent ordre: el número de la peça, el nivell d'aquesta (1, 2 o 3 en aquest cas), el temps de pre-procés r_j, el temps de porcés p_j i el temps de post-procés q_j (Taula A.2):

6				
5		10		5
1	1	2	7	1
2	1	18	1	6
3	1	12	4	3
4	1	13	9	8
5	1	11	9	7
6	2	17	7	19
7	2	5	1	18
8	2	8	5	4
9	2	2	9	9
10	2	15	8	13
11	2	1	4	19
12	2	3	3	20
13	2	10	7	8
14	2	18	8	10
15	2	18	6	18
16	3	14	5	2
17	3	18	3	16
18	3	3	8	5
19	3	15	5	4
20	3	2	8	20
Número peça	Nivell de la peça	r_j	p_j	q_j

- ➔ Número de l'exemplar
- ➔ Número de peces de cada nivell
- ➔ Número de peça, nivell de la peça, rj, pj, qj

Taula A.2 Dades de les peces de l'exemplar 6 de 20 peces

A.2. Presentació dels resultats del problema

Un cop executat el programa s'obtenen els corresponents resultats. En tots els casos l'estructura de presentació és la mateixa. Primer s'indica els diferents valors de F_{MAX} obtinguts per a cada penalització (c) i un cop mostrats tots els valors s'indica el percentatge de solucions no dominades obtingudes respecte les solucions estudiades, el temps d'execució i el número d'exemplar:

Fmax	Penalització
372	0
359	1
349	2
339	3
329	4
325	5
319	6
315	7
310	8
305	9
301	10
295	11
292	12
286	13
284	14
277	15
276	16
268	17
268	18
261	19
259	20
256	21
254	22
253	23
251	24
251	25
247	26
248	27
243	28
246	29
246	30
83,87096	➔ Percentatge de solucions no dominades
351	➔ Temps de resolució de l'exemplar
407	➔ Número d'exemplar

Taula A.3 Presentació dels resultats de l'exemplar de 407 de 200 peces

Annex B. Comparació de les solucions obtingudes per les diferents heurístiques

S'han comparat les heurístiques dos a dos a partir de l'equació 8.2 i s'ha trobat la mitjana de cada heurística.

$$ND = \frac{Solucions\ de\ X\ no\ dominades\ per\ Y}{Total\ de\ solucions\ no\ dominades\ X \cup Y} \qquad \text{(eq. 8.2)}$$

A continuació es presenten tots els valors obtinguts d'aquestes comparacions separats per exemplars i grups de màquines.

B.1. 20 peces

(1,1,1)	1A	1B	2A	2B	Mitjana
1A		0,5586312	0,1041797	0,5655795	0,4094635
1B	0,047096		0,0377868	0,0850038	0,0566289
2A	0,062673	0,5500943		0,5587693	0,3905123
2B	0,052567	0,0764834	0,0482881		0,0591127

Taula B.1 Comparació de les heurístiques en el cas de 20 peces i 3 màquines en la disposició (1,1,1) (g=3)

(2,1,1)	1A	1B	2A	2B	Mitjana
1A		0,5849215	0,0796156	0,5649604	0,4098325
1B	0,043450		0,0419962	0,0593683	0,0482716
2A	0,070321	0,5842391		0,5638247	0,4061281
2B	0,063204	0,1568094	0,0596517		0,0932218

Taula B.2 Comparació de les heurístiques en el cas de 20 peces i 4 màquines en la disposició (2,1,1) (g=3)

(2,2,1)	1A	1B	2A	2B	Mitjana
1A		0,590268	0,051695	0,576874	0,406279
1B	0,050662		0,051017	0,044223	0,048634
2A	0,052577	0,590569		0,577641	0,406929
2B	0,060778	0,119046	0,059971		0,079931

Taula B.3 Comparació de les heurístiques en el cas de 20 peces i 5 màquines en la disposició (2,2,1) (g=3)

(3,2,1)	1A	1B	2A	2B	Mitjana
1A		0,517734	0,031088	0,508823	0,352548
1B	0,034223		0,035645	0,035083	0,034984
2A	0,025994	0,51652		0,505701	0,349405
2B	0,042237	0,115312	0,045611		0,06772

Taula B.4 Comparació de les heurístiques en el cas de 20 peces i 6 màquines en la disposició (3,2,1) (g=3)

B.2. 50 peces

(1,1,1)	1A	1B	2A	2B	Mitjana
1A		0,7257424	0,1676191	0,739506	0,5442892
1B	0,046372		0,0420594	0,184209	0,0908803
2A	0,070156	0,7146242		0,728633	0,5044709
2B	0,046102	0,0771339	0,0480797		0,0571053

Taula B.5 Comparació de les heurístiques en el cas de 50 peces i 3 màquines en la disposició (1,1,1) (g=3)

(2,1,1)	1A	1B	2A	2B	Mitjana
1A		0,7434928	0,1149328	0,7279328	0,5287861
1B	0,062728		0,0546117	0,1020490	0,0731296
2A	0,124711	0,749321		0,7342462	0,5360926
2B	0,079849	0,2106138	0,0707709		0,1204112

Taula B.6 Comparació de les heurístiques en el cas de 50 peces i 4 màquines en la disposició (2,1,1) (g=3)

(2,2,1)	1A	1B	2A	2B	Mitjana
1A		0,752206	0,091304	0,730716	0,524742
1B	0,054916		0,049554	0,065251	0,056574
2A	0,120034	0,759828		0,737224	0,539029
2B	0,077875	0,195310	0,071858		0,115014

Taula B.7 Comparació de les heurístiques en el cas de 50 peces i 5 màquines en la disposició (2,2,1) (g=3)

(3,2,1)	1A	1B	2A	2B	Mitjana
1A		0,661720	0,055811	0,644306	0,453945
1B	0,039323		0,034364	0,044622	0,039437
2A	0,085212	0,674064		0,657888	0,472388
2B	0,063148	0,211522	0,054034		0,109568

Taula B.8 Comparació de les heurístiques en el cas de 50 peces i 6 màquines en la disposició (3,2,1) (g=3)

B.3. 100 peces

(2,1,1)	1A	1B	2A	2B	Mitjana
1A		0,835297	0,1444539	0,8208079	0,6001862
1B	0,059701		0,0413756	0,1308098	0,0772954
2A	0,153713	0,8540417		0,8384692	0,6154080
2B	0,075315	0,2218711	0,0569748		0,1180535

Taula B.9 Comparació de les heurístiques en el cas de 100 peces i 4 màquines en la disposició (2,1,1) (g=3)

(2,2,1)	1A	1B	2A	2B	Mitjana
1A		0,8285072	0,1126319	0,8174248	0,5861880
1B	0,061448		0,0560884	0,0877106	0,0684155
2A	0,133102	0,8356828		0,8244522	0,5977458
2B	0,073204	0,1994168	0,0673301		0,1133170

Taula B.10 Comparació de les heurístiques en el cas de 100 peces i 5 màquines en la disposició (2,2,1) (g=3)

(3,1,1)	1A	1B	2A	2B	Mitjana
1A		0,744327	0,095811	0,729969	0,523369
1B	0,044530		0,035768	0,076178	0,052159
2A	0,122265	0,749803		0,740155	0,537408
2B	0,063153	0,274006	0,048646		0,128602

Taula B.11 Comparació de les heurístiques en el cas de 100 peces i 5 màquines en la disposició (3,1,1) (g=3)

(3,3,1)	1A	1B	2A	2B	Mitjana
1A		0,7638341	0,0668027	0,7475008	0,5260458
1B	0,028246		0,0233371	0,0531198	0,0349011
2A	0,110054	0,7750182		0,7615381	0,5488700
2B	0,046539	0,2818101	0,0384598		0,1222696

Taula B.12 Comparació de les heurístiques en el cas de 100 peces i 7 màquines en la disposició (3,3,1) (g=3)

(4,1,1)	1A	1B	2A	2B	Mitjana
1A		0,564012	0,079727	0,548042	0,397261
1B	0,039931		0,031947	0,034861	0,035579
2A	0,082908	0,5679		0,553447	0,401418
2B	0,059029	0,211112	0,053579		0,107906

Taula B.13 Comparació de les heurístiques en el cas de 100 peces i 6 màquines en la disposició (4,1,1) (g=3)

(4,4,1)	1A	1B	2A	2B	Mitjana
1A		0,5867852	0,0332337	0,5751178	0,3983789
1B	0,015891		0,0095416	0,0337406	0,0197244
2A	0,073201	0,5934775		0,5840933	0,4169238
2B	0,031418	0,2152766	0,0240447		0,0902463

Taula B.14 Comparació de les heurístiques en el cas de 100 peces i 9 màquines en la disposició (4,4,1) (g=3)

(5,1,1)	1A	1B	2A	2B	Mitjana
1A		0,528702	0,018256	0,520601	0,355853
1B	0,021214		0,021464	0,022054	0,021577
2A	0,037660	0,532857		0,526756	0,365758
2B	0,029202	0,224053	0,028702		0,093986

Taula B.15 Comparació de les heurístiques en el cas de 100 peces i 7 màquines en la disposició (5,1,1) (g=3)

(5,5,1)	1A	1B	2A	2B	Mitjana
1A		0,557202	0,007277	0,553185	0,372555
1B	0,006		0,006	0,025857	0,012619
2A	0,00598	0,558016		0,553492	0,372496
2B	0,00972	0,163992	0,010345		0,061353

Taula B.16 Comparació de les heurístiques en el cas de 100 peces i 9 màquines en la disposició (5,5,1) (g=3)

(2,2,1,1)	1A	1B	2A	2B	Mitjana
1A		0,8577805	0,1627147	0,8399774	0,6201575
1B	0,047396		0,0310619	0,1112851	0,0632476
2A	0,173346	0,8748328		0,8552008	0,6344598
2B	0,066424	0,2972791	0,0505508		0,1380847

Taula B.17 Comparació de les heurístiques en el cas de 100 peces i 6 màquines en la disposició (2,2,1,1) (g=4)

(2,2,2,2)	1A	1B	2A	2B	Mitjana
1A		0,8789546	0,1824158	0,8574399	0,6396034
1B	0,02942		0,0221969	0,0997395	0,0504521
2A	0,163833	0,8850372		0,8663576	0,6384091
2B	0,051205	0,3063679	0,0408425		0,1328053

Taula B.18 Comparació de les heurístiques en el cas de 100 peces i 8 màquines en la disposició (2,2,2,2) (g=4)

(3,3,2,1)	1A	1B	2A	2B	Mitjana
1A		0,79596	0,108582	0,766995	0,557179
1B	0,018587		0,014754	0,068801	0,034047
2A	0,145434	0,802806		0,781543	0,576594
2B	0,047892	0,39336	0,036722		0,159325

Taula B.19 Comparació de les heurístiques en el cas de 100 peces i 9 màquines en la disposició (3,3,2,1) (g=4)

(3,3,3,3)	1A	1B	2A	2B	Mitjana
1A		0,804835	0,120217	0,783741	0,569598
1B	0,013783		0,009424	0,067419	0,030209
2A	0,142028	0,808721		0,783888	0,578212
2B	0,035082	0,408208	0,036491		0,159927

Taula B.20 Comparació de les heurístiques en el cas de 100 peces i 12 màquines en la disposició (3,3,3,3) (g=4)

(4,4,2,2)	1A	1B	2A	2B	Mitjana
1A		0,6142154	0,0638410	0,6046549	0,4275704
1B	0,007841		0,0055476	0,0264875	0,013292
2A	0,096734	0,6183833		0,6100568	0,4417247
2B	0,01913	0,3078862	0,0142453		0,113754

Taula B.21 Comparació de les heurístiques en el cas de 100 peces i 12 màquines en la disposició (4,4,2,2) (g=4)

(4,4,4,4)	1A	1B	2A	2B	Mitjana
1A		0,616509	0,07069	0,608051	0,43175
1B	0,007023		0,00716	0,032544	0,015576
2A	0,088257	0,618266		0,609349	0,438624
2B	0,014966	0,322414	0,015438		0,117606

Taula B.22 Comparació de les heurístiques en el cas de 100 peces i 12 màquines en la disposició (4,4,4,4) (g=4)

(2,2,2,1,1)	1A	1B	2A	2B	Mitjana
1A		0,883897	0,196355	0,8633869	0,6478796
1B	0,030622		0,0237539	0,1021099	0,052162
2A	0,173776	0,887896		0,8682834	0,6433184
2B	0,050241	0,3591471	0,0431673		0,1508517

Taula B.23 Comparació de les heurístiques en el cas de 100 peces i 8 màquines en la disposició (2,2,2,1,1) (g=5)

(2,2,2,2,2)	1A	1B	2A	2B	Mitjana
1A		0,8893114	0,1992484	0,8706307	0,6530635
1B	0,025808		0,0212776	0,1002423	0,0491093
2A	0,174737	0,8904043		0,8732297	0,6461238
2B	0,043916	0,3643791	0,0389913		0,1490956

Taula B.24 Comparació de les heurístiques en el cas de 100 peces i 10 màquines en la disposició (2,2,2,2,2) (g=5)

(3,3,3,2,2)	1A	1B	2A	2B	Mitjana
1A		0,80806	0,134948	0,789756	0,577588
1B	0,014907		0,012074	0,076286	0,034422
2A	0,149338	0,810693		0,792954	0,584328
2B	0,03448	0,434890	0,029446		0,166272

Taula B.25 Comparació de les heurístiques en el cas de 100 peces i 13 màquines en la disposició (3,3,3,2,2) (g=5)

(3,3,3,3,3)	1A	1B	2A	2B	Mitjana
1A		0,812061	0,136732	0,794113	0,580969
1B	0,0116		0,011032	0,075334	0,032655
2A	0,14687	0,812836		0,793738	0,584481
2B	0,030066	0,441473	0,028852		0,166797

Taula B.26 Comparació de les heurístiques en el cas de 100 peces i 15 màquines en la disposició (3,3,3,3,3) (g=5)

(4,4,4,2,2)	1A	1B	2A	2B	Mitjana
1A		0,6270367	0,0666284	0,6201471	0,4379374
1B	0,003857		0,0026928	0,0312246	0,0125914
2A	0,100997	0,6276986		0,6183198	0,4490053
2B	0,010896	0,3419204	0,0114378		0,121418

Taula B.27 Comparació de les heurístiques en el cas de 100 peces i 16 màquines en la disposició (4,4,4,2,2) (g=5)

(4,4,4,4,4)	1A	1B	2A	2B	Mitjana
1A		0,626767	0,066254	0,620039	0,437687
1B	0,004357		0,002308	0,034417	0,013694
2A	0,10143	0,628193		0,619979	0,449867
2B	0,011138	0,341211	0,009720		0,120689

Taula B.28 Comparació de les heurístiques en el cas de 100 peces i 20 màquines en la disposició (4,4,4,4,4) (g=5)

B.4. 200 peces

(2,1,1)	1A	1B	2A	2B	Mitjana
1A		0,876584	0,1626587	0,8694614	0,6362347
1B	0,066481		0,0442311	0,1658691	0,0921937
2A	0,179679	0,9003914		0,8909235	0,6569979
2B	0,074095	0,2297439	0,0533648		0,1190680

Taula B.29 Comparació de les heurístiques en el cas de 200 peces i 4 màquines en la disposició (2,1,1) (g=3)

(2,2,1)	1A	1B	2A	2B	Mitjana
1A		0,8839879	0,1268137	0,8593336	0,6233784
1B	0,052100		0,0411985	0,0965669	0,0632886
2A	0,160216	0,8947558		0,8744197	0,6431306
2B	0,077682	0,2790570	0,0616157		0,1394517

Taula B.30 Comparació de les heurístiques en el cas de 200 peces i 5 màquines en la disposició (2,2,1) (g=3)

(3,1,1)	1A	1B	2A	2B	Mitjana
1A		0,784212	0,172867	0,758800	0,571960
1B	0,069710		0,041165	0,090409	0,067095
2A	0,262236	0,804629		0,785253	0,617373
2B	0,101019	0,315393	0,069351		0,161921

Taula B.31 Comparació de les heurístiques en el cas de 200 peces i 5 màquines en la disposició (3,1,1) (g=3)

(3,3,1)	1A	1B	2A	2B	Mitjana
1A		0,8111638	0,1156811	0,7947754	0,5738735
1B	0,041973		0,0374591	0,0772925	0,0522414
2A	0,161585	0,8162037		0,8029036	0,5935641
2B	0,057474	0,338482	0,0517368		0,1492309

Taula B.32 Comparació de les heurístiques en el cas de 200 peces i 7 màquines en la disposició (3,3,1) (g=3)

(4,1,1)	1A	1B	2A	2B	Mitjana
1A		0,599777	0,072509	0,583283	0,418523
1B	0,059408		0,044359	0,049619	0,051129
2A	0,130181	0,621317		0,603028	0,451509
2B	0,085129	0,233406	0,066730		0,128421

Taula B.33 Comparació de les heurístiques en el cas de 200 peces i 6 màquines en la disposició (4,1,1) (g=3)

(4,4,1)	1A	1B	2A	2B	Mitjana
1A		0,6486679	0,0574836	0,6383585	0,4481700
1B	0,016154		0,0129422	0,0332167	0,0207711
2A	0,113877	0,656817		0,6514432	0,4740458
2B	0,029055	0,2461815	0,0193804		0,0982055

Taula B.34 Comparació de les heurístiques en el cas de 200 peces i 9 màquines en la disposició (4,4,1) (g=3)

(5,1,1)	1A	1B	2A	2B	Mitjana
1A		0,519807	0,026499	0,498610	0,348305
1B	0,028048		0,022571	0,023395	0,024671
2A	0,066719	0,533414		0,519315	0,373149
2B	0,061631	0,234276	0,052151		0,116019

Taula B.35 Comparació de les heurístiques en el cas de 200 peces i 7 màquines en la disposició (5,1,1) (g=3)

(5,5,1)	1A	1B	2A	2B	Mitjana
1A		0,575070	0,008763	0,572076	0,385303
1B	0,004583		0,003639	0,018653	0,008958
2A	0,016834	0,577465		0,573315	0,389205
2B	0,008389	0,221307	0,008742		0,079479

Taula B.36 Comparació de les heurístiques en el cas de 200 peces i 11 màquines en la disposició (5,5,1) (g=3)

(2,2,1,1)	1A	1B	2A	2B	Mitjana
1A		0,8919477	0,1396391	0,8644143	0,6320004
1B	0,046239		0,0404931	0,0979395	0,0615573
2A	0,164945	0,8976846		0,8739741	0,6455346
2B	0,074224	0,2651777	0,0635837		0,1343283

Taula B.37 Comparació de les heurístiques en el cas de 200 peces i 6 màquines en la disposició (2,2,1,1) (g=4)

(2,2,2,2)	1A	1B	2A	2B	Mitjana
1A		0,9023611	0,1391501	0,8804851	0,6406654
1B	0,036177		0,0305445	0,0800685	0,0489300
2A	0,156415	0,9075401		0,887312	0,6504224
2B	0,057088	0,2817255	0,0479033		0,1289055

Taula B.38 Comparació de les heurístiques en el cas de 100 peces i 8 màquines en la disposició (2,2,2,2) (g=4)

(3,3,2,1)	1A	1B	2A	2B	Mitjana
1A		0,83254	0,126009	0,814772	0,591107
1B	0,025522		0,022021	0,077377	0,041640
2A	0,158082	0,838584		0,821815	0,606160
2B	0,042335	0,378516	0,038407		0,153086

Taula B.39 Comparació de les heurístiques en el cas de 100 peces i 9 màquines en la disposició (3,3,2,1) (g=4)

(3,3,3,3)	1A	1B	2A	2B	Mitjana
1A		0,843925	0,140687	0,827024	0,603879
1B	0,018022		0,020927	0,072228	0,037059
2A	0,148733	0,841363		0,823383	0,604493
2B	0,033089	0,392200	0,038843		0,154711

Taula B.40 Comparació de les heurístiques en el cas de 100 peces i 12 màquines en la disposició (3,3,3,3) (g=4)

(4,4,2,2)	1A	1B	2A	2B	Mitjana
1A		0,6608581	0,0681159	0,6446277	0,4578672
1B	0,005785		0,0033606	0,0327359	0,0139604
2A	0,118324	0,6670684		0,6542794	0,4798907
2B	0,024118	0,2771982	0,017469		0,1062618

Taula B.41 Comparació de les heurístiques en el cas de 100 peces i 12 màquines en la disposició (4,4,2,2) (g=4)

(4,4,4,4)	1A	1B	2A	2B	Mitjana
1A		0,670137	0,080514	0,65666	0,469104
1B	0,003945		0,003760	0,036794	0,014833
2A	0,126493	0,673838		0,662059	0,487463
2B	0,018819	0,304445	0,018005		0,113757

Taula B.42 Comparació de les heurístiques en el cas de 100 peces i 16 màquines en la disposició (4,4,4,4) (g=4)

(2,2,2,1,1)	1A	1B	2A	2B	Mitjana
1A		0,9074373	0,1673032	0,8829347	0,6525584
1B	0,041275		0,0348136	0,1186123	0,0649003
2A	0,176616	0,9127554		0,8926281	0,6606666
2B	0,066428	0,305836	0,0544259		0,1422300

Taula B.43 Comparació de les heurístiques en el cas de 200 peces i 8 màquines en la disposició (2,2,2,1,1) (g=5)

(2,2,2,2,2)	1A	1B	2A	2B	Mitjana
1A		0,9064873	0,1715702	0,8832445	0,6537673
1B	0,042648		0,0365145	0,1217086	0,0669571
2A	0,175038	0,9113676		0,8918429	0,6594161
2B	0,06639	0,3070584	0,0553956		0,1429478

Taula B.44 Comparació de les heurístiques en el cas de 200 peces i 12 màquines en la disposició (2,2,2,2,2) (g=5)

(3,3,3,2,2)	1A	1B	2A	2B	Mitjana
1A		0,845427	0,149315	0,820923	0,605222
1B	0,021701		0,019047	0,072493	0,037747
2A	0,178068	0,849097		0,825132	0,617432
2B	0,045135	0,446555	0,045142		0,178944

Taula B.45 Comparació de les heurístiques en el cas de 200 peces i 13 màquines en la disposició (3,3,3,2,2) (g=5)

(3,3,3,3,3)	1A	1B	2A	2B	Mitjana
1A		0,845413	0,149811	0,821929	0,605718
1B	0,022077		0,019179	0,073749	0,038335
2A	0,178003	0,849003		0,824689	0,617232
2B	0,044318	0,446513	0,045737		0,178856

Taula B.46 Comparació de les heurístiques en el cas de 200 peces i 15 màquines en la disposició (3,3,3,3,3) (g=5)

(4,4,4,2,2)	1A	1B	2A	2B	Mitjana
1A		0,6727535	0,0928305	0,6527843	0,4727894
1B	0,007323		0,0061433	0,0343691	0,0159452
2A	0,128016	0,6763446		0,6603829	0,4882479
2B	0,028290	0,3428205	0,021982		0,1310309

Taula B.47 Comparació de les heurístiques en el cas de 200 peces i 16 màquines en la disposició (4,4,4,2,2) (g=5)

(4,4,4,4,4)	1A	1B	2A	2B	Mitjana
1A		0,672852	0,093413	0,652263	0,472842
1B	0,007323		0,006421	0,035833	0,016526
2A	0,128177	0,676121		0,659636	0,487978
2B	0,028956	0,343133	0,022819		0,131636

Taula B.48 Comparació de les heurístiques en el cas de 200 peces i 20 màquines en la disposició (4,4,4,4,4) (g=5)

Annex C. Criteris d'elecció de peça de la Fase 2

En aquest annex s'explica l'estudi realitzat per escollir el millor criteri per escollir la millor peça candidata *jc* a canviar de nivell en la primera iteració de la Fase 2. S'han escollit diferents criteris per cada jc ∈ JC:

1. $jm = max_{jc}(p_{jc})$
2. $jm = min_{jc}(p_{jc})$
3. Si en la Fase 1 s'utilitza el *procediment A* en la Fase 2 utilitzar el *procediment B* i viceversa.

Per tal de determinar el millor criteri es farà un comparativa dels mateixos exemplars utilitzant els 3 criteris per escollir *jm* de la Fase 2.

Es compararan la següents heurístiques:

- Algorisme 2A amb el criteri *1* → 2A1
- Algorisme 2A amb el criteri *2* → 2A2
- Algorisme 2A amb el criteri *3* → 2A3
- Algorisme 2B amb el criteri *1* → 2B1
- Algorisme 2B amb el criteri *2* → 2B2
- Algorisme 2B amb el criteri *3* → 2B3

Inicialment s'estudiarà una combinació de màquines per cadascun dels exemplars de peces. Si en aquest estudi no s'obtenen resultats concloents s'ampliarà la mostra.

Els resultats que s'estudiaran són:

- 20 peces (3,2,1)
- 50 peces (2,1,1)
- 100 peces (4,4,1)
- 200 peces (5,1,1)

En total s'analitzaran un total de 24 casos, combinació dels 4 exemplars amb les 6 combinacions presentades.

A continuació es mostren les taules amb el percentatges de solucions no dominades d'una heurística respecte l'altra. S'ha calculat aquest paràmetre a partir de l'equació 8.2.

$$ND = \frac{Solucions\ de\ X\ no\ dominades\ per\ Y}{Total\ de\ solucions\ no\ dominades\ X \cup Y} \qquad \text{(eq. 8.2)}$$

C.1. Exemplars de 20 peces

A la Taula C.1 s'observa com el criteri amb més solucions no dominades respecte la resta és la Heurística 2A amb el criteri 1.

(3,2,1)	2A1	2A2	2A3	2B1	2B2	2B3	Mitjana
2A1		0,085745	0,061646	0,505701	0,516068	0,504813	0,334795
2A2	0,033682		0,058092	0,48674	0,494851	0,485875	0,311848
2A3	0,027270	0,072857		0,490903	0,501756	0,491079	0,316773
2B1	0,045611	0,061902	0,056799		0,106808	0,026106	0,059445
2B2	0,038968	0,050954	0,047412	0,044438		0,047703	0,045895
2B3	0,043249	0,061929	0,054198	0,033521	0,117333		0,062046

Taula C.1 Comparació de les heurístiques en el cas de 20 peces i 6 màquines en la disposició (3,2,1) (g=3)

C.2. Exemplars de 50 peces

A la Taula C.2 s'observa com es repeteixen els resultats dels exemplars de 20 peces. El criteri que presenta més solucions no dominades respecte la resta és Heurística 2A amb el criteri 1.

(2,1,1)	2A1	2A2	2A3	2B1	2B2	2B3	Mitjana
2A1		0,111819	0,196946	0,734246	0,743279	0,73079	0,503416
2A2	0,104627		0,18744	0,733802	0,742413	0,730902	0,499837
2A3	0,06991	0,076086		0,692456	0,704299	0,691999	0,44695
2B1	0,070771	0,068969	0,107628		0,179831	0,098674	0,105174
2B2	0,064114	0,061285	0,096271	0,108278		0,112823	0,088554
2B3	0,072986	0,071082	0,107672	0,130406	0,204683		0,117366

Taula C.2 Comparació de les heurístiques en el cas de 50 peces i 4 màquines en la disposició (2,1,1) (g=3)

C.3. Exemplars de 100 peces

A la Taula C.3 s'observa com en el cas de 100 peces es repeteix la tendència i el criteri que presenta més solucions no dominades en comparació amb la resta de criteris és l'algorisme 2A amb el criteri 1.

(4,4,1)	2A1	2A2	2A3	2B1	2B2	2B3	Mitjana
2A1		0,110211	0,095110	0,584093	0,595975	0,595288	0,396135
2A2	0,023906		0,047507	0,560577	0,568330	0,571740	0,354412
2A3	0,014008	0,068775		0,564112	0,581522	0,576606	0,361005
2B1	0,024045	0,046023	0,036733		0,191008	0,164870	0,092536
2B2	0,016233	0,023334	0,025734	0,051361		0,093890	0,042110
2B3	0,009826	0,030802	0,021623	0,039082	0,116621		0,043591

Taula C.3 2 Comparació de les heurístiques en el cas de 100 peces i 9 màquines en la disposició (4,4,1) (g=3)

C.4. Exemplars de 200 peces

Finalment, a la Taula C.4, es pot comprovar que totes les mostres realitzades segueixen el mateix patró i que la combinació que obté menys solucions no dominades en comparació amb la resta d'heurístiques és l'algorisme 2A amb el criteri 1.

(5,1,1)	2A1	2A2	2A3	2B1	2B2	2B3	Mitjana
2A1		0,082024	0,061265	0,519315	0,525839	0,517648	0,341218
2A2	0,026148		0,045297	0,494638	0,500229	0,492352	0,311733
2A3	0,01664	0,051424		0,498524	0,507504	0,499238	0,314666
2B1	0,052151	0,057847	0,06531		0,200172	0,049563	0,085009
2B2	0,041903	0,051179	0,053052	0,061376		0,058425	0,053187
2B3	0,053125	0,065268	0,065345	0,105136	0,224614		0,102698

Taula C.4 Comparació de les heurístiques en el cas de 200 peces i 7 màquines en la disposició (5,1,1) (g=3)

C.5. Elecció del criteri de la Fase 2

A partir dels resultats obtinguts es pot concloure que el millor criteri per escollir la millor peça candidata a canviar de nivell en la Fase 2 és el Criteri 1. És a dir, escollint la peça amb jm = $\max_{jc}(p_{jc})$.

Observant els resultats es pot veure que aquest criteri és el que ens permet obtenir més solucions no dominades en una de les heurístiques. Essent aquest l'objectiu d'aquest estudi.

Annex D. Codi dels procediments proposats

En aquest annex s'inclouen els codis de programació dels algorismes dissenyats a la memòria.

Inicialment s'hi inclouen les funcions comunes per tots els algorismes i posteriorment el programa final de cada un.

Tots els procediments s'han implementat en el llenguatge C# mitjançant el programari Microsoft Visual Studio 2010. Per facilitar la comprensió s'han afegit alguns aclariments seguits del símbol //.

D.1. Variables

```
        public static int numPeces = 21;
        public static int numNivells = 8;
        public static int numMaquines = 15;
        int[] Maquines;
        List<int[]> PecesIni;
        int[] PecaParticular; //S'utilitzarà per crear el vector peces
        public int r = 0;
        public int p = 0;
        public int q = 0;
        public int nivellPeca = 1;
        public List<int[]> rp;
        public int[] rpi; //serveix per crear rp
        public int[] PecesIniE; //per tornar a crear peces ini
        public List<int[]> pq;
        public int[] pqi; //serveix per crear pq
        public List<int[]> conjuntJr; //conjunt Jr per fer l'heurística inicial
        public List<int[]> conjuntJq; //conjunt Jq per fer l'heurística inicial
        public bool boolFaseA; //per utilitzar a la fase A
        public int compararrppq; //per comparar durant la fase A
        public List<int[]>[] solucioA; //el valor que retorna la fase A: [conjunt JR,
conjunt JM; conjunt JQ]
        public List<List<int[]>> LlistaTempsMaquines; //llista que ens indica la
màquina i els instants de temps que està ocupada [[[num maquina, nivell],[t ocupada, t
lliure, num peça],[t ocupada, t lliure, num peça]],[...],[...]]
```

D.2. Codi de programació comú

A continuació es detallen totes les funcions que s'han utilitzat en la programació de les heurístiques.

A partir d'aquestes funcions s'ha generat el codi principal de les 4 heurístiques presentades al treball: l'Heurística 1A, l'Heurística 1B, l'Heurística 2A i l'Heurística 2B.

```
        public List<int[]> pecesIni2(List<List<int[]>> LlistaTempsMaquines, int
nivelldesti, int[] maquines, int[] PecaCanviNivell)
        {
            int indexmaquina = 0;
            List<int[]> PecesIni2 = new List<int[]>();
```

```
            List<int[]> llistaPeces = new List<int[]>();

            for (int i = 0; i < nivelldesti; i++)
            {
                indexmaquina = indexmaquina + maquines[i];
            }
            for (int e = indexmaquina; e < indexmaquina + maquines[nivelldesti]; e++)
            {
                for (int r = 1; r < LlistaTempsMaquines[e].Count(); r++)
                {
                    llistaPeces.Add(LlistaTempsMaquines[e][r]);
                }
            }
            llistaPeces.Add(PecaCanviNivell);  //afegim la peça que volem canviar

            for (int i = 0; i < llistaPeces.Count(); i++)
            {
                int[] PecesIni3 = new int[5];
                PecesIni3[0] = llistaPeces[i][2]; //r
                PecesIni3[1] = llistaPeces[i][1] - llistaPeces[i][0]; //p
                PecesIni3[2] = llistaPeces[i][3]; //q
                PecesIni3[3] = llistaPeces[i][4]; //nivell
                PecesIni3[4] = llistaPeces[i][5]; //num peça
                PecesIni2.Add(PecesIni3);
            }
            return PecesIni2;
        }


        public int[] llistaMaquines2(int numMaquines, int numNivells)
        {
            Maquines = new int[numNivells];
            int m = 0;
            for (int i = 0; i < numNivells; i++)
            {
                for (int e = i; e < numNivells; e++)
                {
                    if (m < numMaquines)
                    {
                        Maquines[e]++;
                        m++;
                    }
                }
            }
            if (m < numMaquines)
            {
                Maquines[0] = Maquines[0] + numMaquines - m;
            }
            return Maquines;
        }


        public List<int[]> ordenarrp(List<int[]> PecesIni)  // ordena les peces de r+p
major a menor
        {
            rp = new List<int[]>();
            for (int e = 0; e < PecesIni.Count(); e++)  //sumem r+p
            {
                rpi = new int[5];
                rpi[0] = PecesIni[e][0];  // r
                rpi[1] = PecesIni[e][1] + PecesIni[e][0]; // r + p
```

```
            rpi[2] = PecesIni[e][2]; // q
            rpi[3] = PecesIni[e][3]; // nivell peça
            rpi[4] = PecesIni[e][4]; // enumeració peça

            rp.Add(rpi);
        }

        int col1 = 0, col2 = 0, col3 = 0, col4 = 0, col5 = 0;          //ordenem
        for (int e = 0; e < PecesIni.Count(); e++)
        {
            for (int i = 0; i < PecesIni.Count(); i++)
            {
                if (rp[e][1] < rp[i][1])
                {
                    col1 = rp[e][0];
                    col2 = rp[e][1];
                    col3 = rp[e][2];
                    col4 = rp[e][3];
                    col5 = rp[e][4];

                    rp[e][0] = rp[i][0];
                    rp[e][1] = rp[i][1];
                    rp[e][2] = rp[i][2];
                    rp[e][3] = rp[i][3];
                    rp[e][4] = rp[i][4];

                    rp[i][0] = col1;
                    rp[i][1] = col2;
                    rp[i][2] = col3;
                    rp[i][3] = col4;
                    rp[i][4] = col5;
                }
            }
        }
        return rp;
    }


    public List<int[]> PecesIniRP(List<int[]> rpordenat)
    {
        PecesIni = new List<int[]>();
        for (int e = 0; e < rpordenat.Count(); e++)  //sumem r+p
        {
            PecesIniE = new int[5];
            PecesIniE[0] = rpordenat[e][0];  // r
            PecesIniE[1] = rpordenat[e][1] - rpordenat[e][0]; // p
            PecesIniE[2] = rpordenat[e][2]; // q
            PecesIniE[3] = rpordenat[e][3]; // nivell peça
            PecesIniE[4] = rpordenat[e][4]; // enumeració peça

            PecesIni.Add(PecesIniE);
        }
        return PecesIni;
    }


    public List<int[]> ordenarpq(List<int[]> PecesIni)
    {
        pq = new List<int[]>();
        for (int e = 0; e < PecesIni.Count(); e++)  //sumem q+p
        {
```

```
                pqi = new int[5];
                pqi[0] = PecesIni[e][0];   //r
                pqi[1] = PecesIni[e][1] + PecesIni[e][2]; // p + q
                pqi[2] = PecesIni[e][2]; //q
                pqi[3] = PecesIni[e][3]; //nivell
                pqi[4] = PecesIni[e][4]; // num peça

                pq.Add(pqi);
            }

            int colp1 = 0, colp2 = 0, colp3 = 0, colp4 = 0, colp5 = 0;
            for (int e = 0; e < PecesIni.Count(); e++)
            {
                for (int i = 0; i < PecesIni.Count(); i++)
                {
                    if (pq[e][1] < pq[i][1])
                    {
                        colp1 = pq[e][0];
                        colp2 = pq[e][1];
                        colp3 = pq[e][2];
                        colp4 = pq[e][3];
                        colp5 = pq[e][4];

                        pq[e][0] = pq[i][0];
                        pq[e][1] = pq[i][1];
                        pq[e][2] = pq[i][2];
                        pq[e][3] = pq[i][3];
                        pq[e][4] = pq[i][4];

                        pq[i][0] = colp1;
                        pq[i][1] = colp2;
                        pq[i][2] = colp3;
                        pq[i][3] = colp4;
                        pq[i][4] = colp5;
                    }
                }
            }
            return pq;
        }


        public List<int[]> PecesIniPQ(List<int[]> pqordenat)
        {
            PecesIni = new List<int[]>();
            for (int e = 0; e < pqordenat.Count(); e++)  //sumem r+p
            {
                PecesIniE = new int[5];
                PecesIniE[0] = pqordenat[e][0];  // q
                PecesIniE[1] = pqordenat[e][1] - pqordenat[e][2]; ; // p
                PecesIniE[2] = pqordenat[e][2];  // q
                PecesIniE[3] = pqordenat[e][3]; // nivell peça
                PecesIniE[4] = pqordenat[e][4]; // enumeració peça

                PecesIni.Add(PecesIniE);
            }
            return PecesIni;
        }


        public List<int[]> Pas2faseA(List<int[]> listRP, List<int[]> conjuntJr, int[]
ListaMaquines, int nivell) // Segon pas del pas A de la Fase 0
```

```
        {
            for (int i = 0; i < listRP.Count() - ListaMaquines[nivell]; i++)
//comparem r+p < r
            {
                if (listRP.Count() <= ListaMaquines[nivell])
                {
                    break;
                }
                else if (listRP[i][1] <= listRP[i + ListaMaquines[nivell]][0])
                {
                    listRP[i][1] = listRP[i][1] - listRP[i][0];
                    conjuntJr.Add(listRP[i]);           //afegim rp a Jr
                    listRP.RemoveAt(i);           // treiem rp a J
                    i = i - 1;
                }
                else
                {
                    break;
                }
            }
            return conjuntJr;
        }


        public List<int[]> Pas3faseA(List<int[]> listPQ, List<int[]> conjuntJq, int[]
ListaMaquines, int nivell) // Tercer pas del pas A de la Fase 0
        {
            for (int i = 0; i < listPQ.Count() - ListaMaquines[nivell]; i++)
//comparem p + q < p
            {
                if (listPQ.Count() <= ListaMaquines[nivell])
                {
                    break;
                }
                else if (listPQ[i][1] <= listPQ[i + ListaMaquines[nivell]][2])
                {
                    listPQ[i][1] = listPQ[i][1] - listPQ[i][2];
                    conjuntJq.Add(listPQ[i]);           //afegim pq a Jq
                    listPQ.RemoveAt(i);         // treiem rp a J
                    i = i - 1;
                }
                else
                {
                    break;
                }
            }
            return conjuntJq;
        }


        public List<int[]>[] FaseA(int numNivells, int[] LlistaMaquines, int nivell,
List<int[]> PecesIni)
        {
            solucioA = new List<int[]>[3];
            int compararrppq = 0;
            conjuntJr = new List<int[]>();
            conjuntJq = new List<int[]>();
            bool boolFaseA = true;

            while (boolFaseA == true)
            {
```

```
                rp = ordenarrp(PecesIni);
                conjuntJr = Pas2faseA(rp, conjuntJr, LlistaMaquines, nivell);
                compararrppq = rp.Count();
                PecesIni = PecesIniRP(rp);
                pq = ordenarpq(PecesIni);
                conjuntJq = Pas3faseA(pq, conjuntJq, LlistaMaquines, nivell);
                if (compararrppq == pq.Count                {
                    boolFaseA = false;
                }
                PecesIni = PecesIniPQ(pq);
            }
            solucioA[0] = conjuntJr;
            solucioA[1] = PecesIni;
            solucioA[2] = conjuntJq;
            return solucioA;
        }


        public void escriureFaseA(List<int[]>[] solucio)
        {
            Console.WriteLine();
            Console.WriteLine("conjunt JR:");
            for (int i = 0; i < solucio[0].Count(); i++)
            {
                Console.WriteLine(solucio[0][i][0] + " " + solucio[0][i][1] + " " +
solucio[0][i][2] + " " + solucio[0][i][3] + " " + solucio[0][i][4]);
            }
            Console.WriteLine();
            Console.WriteLine("conjunt JM:");
            for (int i = 0; i < solucio[1].Count(); i++)
            {
                Console.WriteLine(solucio[1][i][0] + " " + solucio[1][i][1] + " " +
solucio[1][i][2] + " " + solucio[1][i][3] + " " + solucio[1][i][4]);
            }
            Console.WriteLine();
            Console.WriteLine("conjunt JQ:");
            for (int i = 0; i < solucio[2].Count(); i++)
            {
                Console.WriteLine(solucio[2][i][0] + " " + solucio[2][i][1] + " " +
solucio[2][i][2] + " " + solucio[2][i][3] + " " + solucio[2][i][4]);
            }
        }


        public List<List<int[]>> tempsMaquines(int[] Maquines)
        {
            LlistaTempsMaquines = new List<List<int[]>>(1);
            for (int i = 0; i < Maquines.Length; i++)
            {
                for (int e = 0; e < Maquines[i]; e++)
                {
                    int[] MaquinaNivell = { e, i };
                    List<int[]> LlistaMaquina = new List<int[]>();
                    LlistaMaquina.Add(MaquinaNivell);
                    LlistaTempsMaquines.Add(LlistaMaquina);
                }
            }
            return LlistaTempsMaquines;
        }
```

```
        public List<List<int[]>> assignacioJM(List<List<int[]>> tempsMaquines,
List<int[]>[] solucioA, int[] Maquines, int nivell)
        {
            int[] ordenarJM = { 0, 0, 0, 0, 0 };
            for (int e = 0; e < solucioA[1].Count(); e++)
            {
                for (int i = 0; i < solucioA[1].Count(); i++)
                {
                    if (solucioA[1][e][0] < solucioA[1][i][0])
                    {
                        ordenarJM = solucioA[1][e];
                        solucioA[1][e] = solucioA[1][i];
                        solucioA[1][i] = ordenarJM;
                    }
                }
            }
            List<int[]> conjuntJM = solucioA[1];
            int actualitzador = 0;
            for (int i = 0; i < Maquines[nivell] - actualitzador; i++)
            {
                if (conjuntJM.Count() != 0)
                {
                    int[] PecaMaquina = { conjuntJM[i][0], conjuntJM[i][0] +
conjuntJM[i][1], conjuntJM[i][0], conjuntJM[i][2], conjuntJM[i][3], conjuntJM[i][4] };
                    conjuntJM.RemoveAt(i);
                    tempsMaquines[i + actualitzador].Add(PecaMaquina);
                    i--;
                    actualitzador++;
                }
            }
            if (conjuntJM.Count() != 0)
            {
                for (int e = 0; e < conjuntJM.Count(); e++)
                {
                    int a = tempsMaquines[0][tempsMaquines[0].Count() - 1][1];
                    int b = 0;
                    int iinici = 0;
                    for (int i = iinici; i < iinici + Maquines[nivell]; i++)
                    {
                        if (a > tempsMaquines[i][tempsMaquines[i].Count() - 1][1])
                        {
                            a = tempsMaquines[i][tempsMaquines[i].Count() - 1][1];
                            b = tempsMaquines[i][0][0];
                        }
                    }
                    int[] TIninci = { conjuntJM[e][0],
tempsMaquines[b][tempsMaquines[b].Count() - 1][1] };
                    int maxim = TIninci.Max();
                    int[] PecaMaquina = { maxim, maxim + conjuntJM[e][1],
conjuntJM[e][0], conjuntJM[e][2], conjuntJM[e][3], conjuntJM[e][4] };
                    tempsMaquines[b].Add(PecaMaquina);
                    conjuntJM.RemoveAt(e);
                    e--;

                    if (conjuntJM.Count() == 0)
                    {
                        break;
                    }
                }
            }
            return tempsMaquines;
```

```
        }


        public List<List<int[]>> assignacioJQ(List<List<int[]>> tempsMaquines,
List<int[]>[] solucioA, int[] Maquines, int nivell)
        {
            int[] ordenarJQ = { 0, 0, 0, 0, 0 };
            for (int e = 0; e < solucioA[2].Count(); e++)
            {
                for (int i = 0; i < solucioA[2].Count(); i++)
                {
                    if (solucioA[2][e][0] < solucioA[2][i][0])
                    {
                        ordenarJQ = solucioA[2][e];
                        solucioA[2][e] = solucioA[2][i];
                        solucioA[2][i] = ordenarJQ;
                    }
                }
            }
            List<int[]> conjuntJQ = solucioA[2];
            if (conjuntJQ.Count() != 0)
            {
                for (int e = 0; e < conjuntJQ.Count(); e++)
                {
                    int a = tempsMaquines[0][tempsMaquines[0].Count() - 1][1];
                    int b = 0;
                    int iinici = 0;
                    for (int i = iinici; i < iinici + Maquines[nivell]; i++)
                    {
                        if (a > tempsMaquines[i][tempsMaquines[i].Count() - 1][1])
                        {
                            a = tempsMaquines[i][tempsMaquines[i].Count() - 1][1];
                            b = tempsMaquines[i][0][0];
                        }
                    }
                    int[] TIninci = { conjuntJQ[e][0],
tempsMaquines[b][tempsMaquines[b].Count() - 1][1] };
                    int maxim = TIninci.Max();
                    int[] PecaMaquina = { maxim, maxim + conjuntJQ[e][1],
conjuntJQ[e][0], conjuntJQ[e][2], conjuntJQ[e][3], conjuntJQ[e][4] };
                    tempsMaquines[b].Add(PecaMaquina);
                    conjuntJQ.RemoveAt(e);
                    e--;

                    if (conjuntJQ.Count() == 0)
                    {
                        break;
                    }

                }
            }
            return tempsMaquines;
        }


        public List<List<int[]>> assignacioJR(List<List<int[]>> tempsMaquines,
List<int[]>[] solucioA, int[] Maquines, int nivell)
        {
            int[] ordenarJR = { 0, 0, 0, 0, 0 };
            for (int e = 0; e < solucioA[0].Count(); e++)
            {
```

```
                for (int i = 0; i < solucioA[0].Count(); i++)
                {
                    if (solucioA[0][e][2] > solucioA[0][i][2])
                    {
                        ordenarJR = solucioA[0][e];
                        solucioA[0][e] = solucioA[0][i];
                        solucioA[0][i] = ordenarJR;
                    }
                }
            }
            List<int[]> conjuntJR = solucioA[0];
            if (conjuntJR.Count() != 0)
            {
                for (int e = 0; e < conjuntJR.Count(); e++)
                {
                    int a = tempsMaquines[0][tempsMaquines[0].Count() - 1][1] +
tempsMaquines[0][tempsMaquines[0].Count() - 1][3] - tempsMaquines[0][1][0];

                    int b = 0;
                    int iinici = 0;
                    if (nivell != 0)
                        iinici = 0;
                    for (int i = iinici; i < iinici + Maquines[nivell]; i++)
{
                        if (a > tempsMaquines[i][tempsMaquines[i].Count() - 1][1] +
tempsMaquines[i][tempsMaquines[i].Count() - 1][3] - tempsMaquines[i][1][0])
                        {
                            a = tempsMaquines[i][tempsMaquines[i].Count() - 1][1] +
tempsMaquines[i][tempsMaquines[i].Count() - 1][3] - tempsMaquines[i][1][0];
                            b = tempsMaquines[i][0][0];
                        }
                    }
                    int[] TIninci = { conjuntJR[e][3], a };
                    int maxim = TIninci.Max();
                    int[] PecaMaquina = { tempsMaquines[b][1][0] - conjuntJR[e][1],
tempsMaquines[b][1][0], conjuntJR[e][0], conjuntJR[e][2], conjuntJR[e][3],
conjuntJR[e][4] };
                    tempsMaquines[b].Insert(1, PecaMaquina);
                    conjuntJR.RemoveAt(e);
                    e--;
                    int TempsInici = tempsMaquines[b][1][0];
int TempsNecesari = tempsMaquines[b][1][2] + tempsMaquines[b][1][3];
                    if (TempsInici < TempsNecesari)
                    {
                        for (int g = 1; g < tempsMaquines[b].Count(); g++)
                        {
                            tempsMaquines[b][g][0] = tempsMaquines[b][g][0] +
TempsNecesari - TempsInici;
                            tempsMaquines[b][g][1] = tempsMaquines[b][g][1] +
TempsNecesari - TempsInici;
                        }
                    }
                    if (conjuntJR.Count() == 0)
                    {
                        break;
                    }
                }
            }


            return tempsMaquines;
```

```
        }


        public List<List<int[]>> FaseB(List<int[]>[] solucioA, int[] Maquines, int
nivell)
        {
            List<List<int[]>> LlistaTempsMaquines = tempsMaquines(Maquines);
            List<List<int[]>> sequenciaJM = assignacioJM(LlistaTempsMaquines,
solucioA, Maquines, nivell);
            List<List<int[]>> sequenciaJQ = assignacioJQ(sequenciaJM, solucioA,
Maquines, nivell);
            List<List<int[]>> sequenciaJR = assignacioJR(sequenciaJQ, solucioA,
Maquines, nivell);

            for (int e = 0; e < Maquines[0]; e++)
            {
                for (int i = 1; i < sequenciaJR[e].Count() - 1; i++)                {
                    sequenciaJR[e][i].Concat(new int[] { i });
                }
            }
            return sequenciaJR;
        }



        public List<List<int[]>> HeuristicaInicial(int[] maquinesss, int numNivells,
List<int[]> PecesIni, int nivelldesti)
        {
            List<int[]>[] solucioA = FaseA(numNivells, maquinesss, nivelldesti,
PecesIni);
            List<List<int[]>> solucioB = FaseB(solucioA, maquinesss, nivelldesti);
            return solucioB;
        }




    }

    public class Arborescent //classe on es desenvoluparà el procés arborescent
    {
        Definicio def = new Definicio(); //creem l'objecte

        public List<List<int[]>> HeuristicaPasAnterior;

        public int Fmaxima(List<List<int[]>> heuristica)
        {
            int Fmax = 0;
            for (int i = 0; i < heuristica.Count(); i++)
            {
                for (int e = 1; e < heuristica[i].Count(); e++)
                {
                    if (Fmax < heuristica[i][e][1] + heuristica[i][e][3])
                    {
                        Fmax = heuristica[i][e][1] + heuristica[i][e][3];
                    }
                }
            }
            return Fmax;
        }
```

```
        public int FmaximaOrigen(List<List<int[]>> heuristica, int indexmaquina)
//mira la Fmax de la màquina que li diem
        {
            int Fmax = 0;
            for (int e = 1; e < heuristica[indexmaquina].Count(); e++)
            {
                if (Fmax < heuristica[indexmaquina][e][1] +
heuristica[indexmaquina][e][3])
                {
                    Fmax = heuristica[indexmaquina][e][1] +
heuristica[indexmaquina][e][3];
                }
            }
            return Fmax;
        }

        public int Fminima(List<List<int[]>> heuristica)
        {
            int Fmin = heuristica[0][heuristica[0].Count() - 1][1] +
heuristica[0][heuristica[0].Count() - 1][3];
            for (int i = 0; i < heuristica.Count(); i++)
            {
                for (int e = 1; e < heuristica[i].Count(); e++)
                {
                    if (Fmin > heuristica[i][e][1] + heuristica[i][e][3])
                    {
                        Fmin = heuristica[i][e][1] + heuristica[i][e][3];
                    }
                }
            }
            return Fmin;
        }

        public List<List<int[]>> CopiarLlista(List<List<int[]>> Heuristica)
        {
            List<List<int[]>> HipotesiInicial = new List<List<int[]>>();
            for (int e = 0; e < Heuristica.Count(); e++)
            {
                List<int[]> Heuristica2 = new List<int[]>();
                for (int r = 0; r < Heuristica[e].Count(); r++)
                {
                    Heuristica2.Add(Heuristica[e][r]);
                }
                HipotesiInicial.Add(Heuristica2);
            }
            return HipotesiInicial;
        }


        public int[] maqOrigen(List<List<int[]>> Heuristica)
        {
            int maquina = 0;
            int nivellOrigen = 0;
            int Fmax = 0;
            int[] origen = { 0, 0, 0 };
            for (int i = 0; i < Heuristica.Count(); i++)
            {
                if (Heuristica[i].Count() > 1)
                {
                    for (int e = 1; e < Heuristica[i].Count(); e++)
                    {
```

```
                        if (Heuristica[i][e][1] + Heuristica[i][e][3] > Fmax)
                        {
                            Fmax = Heuristica[i][e][1] + Heuristica[i][e][3];
                            maquina = i;
                            nivellOrigen = Heuristica[i][0][1];
                        }
                        else if (Heuristica[i][e][1] + Heuristica[i][e][3] == Fmax)
                        {
                            if (nivellOrigen < Heuristica[i][0][1])
                            {
                                Fmax = Heuristica[i][e][1] + Heuristica[i][e][3];
                                maquina = i;
                                nivellOrigen = Heuristica[i][0][1];
                            }
                            else if (nivellOrigen == Heuristica[i][0][1])
                            {
                                if (Heuristica[maquina].Count() >
Heuristica[i].Count())
                                {
                                    Fmax = Heuristica[i][e][1] + Heuristica[i][e][3];
                                    maquina = i;
                                    nivellOrigen = Heuristica[i][0][1];
                                }
                            }
                        }
                    }
                }
            }
            origen[0] = maquina;
            origen[1] = nivellOrigen;
            origen[2] = Fmax;
            return origen;
        }


        public List<List<int[]>> canvi1Peca(int[] maqOrigen, List<List<int[]>>
Heuristica, int[] maquines, int numNivells, int index)
        {
            HeuristicaPasAnterior = CopiarLlista(Heuristica);
            Definicio objecte = new Definicio();            List<int[]> pecesInicials
= objecte.pecesIni2(Heuristica, Heuristica[maqOrigen[0]][0][1] + 1, maquines,
Heuristica[maqOrigen[0]][index]);
            List<List<int[]>> Solucio = objecte.HeuristicaInicial(maquines,
numNivells, pecesInicials, maqOrigen[1] + 1);
            int marcador = 0;
            for (int i = 0; i < maqOrigen[1] + 1; i++)
            {
                marcador = marcador + maquines[i];
            }
            for (int i = 0; i < HeuristicaPasAnterior.Count(); i++)
            {
                if (HeuristicaPasAnterior[i][0][1] == maqOrigen[1] + 1)
                {
                    Solucio[i - marcador][0] = HeuristicaPasAnterior[i][0];
                    HeuristicaPasAnterior[i] = Solucio[i - marcador];
                }
            }
            HeuristicaPasAnterior[maqOrigen[0]].RemoveAt(index);
            for (int e = 1; e < HeuristicaPasAnterior[maqOrigen[0]].Count(); e++)
            {
                if (e == 1)  //si no hi ha altres peces a la maquina
```

```
                {
                    int a = HeuristicaPasAnterior[maqOrigen[0]][e][0];
                    HeuristicaPasAnterior[maqOrigen[0]][e][0] =
HeuristicaPasAnterior[maqOrigen[0]][e][2];
                    HeuristicaPasAnterior[maqOrigen[0]][e][1] =
HeuristicaPasAnterior[maqOrigen[0]][e][1] - a +
HeuristicaPasAnterior[maqOrigen[0]][e][0];
                }
                else
                {
                    int a = HeuristicaPasAnterior[maqOrigen[0]][e][0];
                    HeuristicaPasAnterior[maqOrigen[0]][e][0] =
Math.Max(HeuristicaPasAnterior[maqOrigen[0]][e][2],
HeuristicaPasAnterior[maqOrigen[0]][e - 1][1]);
                    HeuristicaPasAnterior[maqOrigen[0]][e][1] =
HeuristicaPasAnterior[maqOrigen[0]][e][1] - a +
HeuristicaPasAnterior[maqOrigen[0]][e][0];
                }
            }
            return HeuristicaPasAnterior;
        }


        public List<List<int[]>> canvi1PecaIndex(int[] maqOrigen, List<List<int[]>>
Heuristica, int[] maquines, int numNivells, int index)
        {
            HeuristicaPasAnterior = CopiarLlista(Heuristica);
            HeuristicaPasAnterior[maqOrigen[0]].RemoveAt(index);
            for (int e = 1; e < HeuristicaPasAnterior[maqOrigen[0]].Count(); e++)
            {
                if (e == 1)  //si no hi ha altres peces a la maquina
                {
                    int a = HeuristicaPasAnterior[maqOrigen[0]][e][0];
                    HeuristicaPasAnterior[maqOrigen[0]][e][0] =
HeuristicaPasAnterior[maqOrigen[0]][e][2];
                    HeuristicaPasAnterior[maqOrigen[0]][e][1] =
HeuristicaPasAnterior[maqOrigen[0]][e][1] - a +
HeuristicaPasAnterior[maqOrigen[0]][e][0];
                }
                else
                {
                    int a = HeuristicaPasAnterior[maqOrigen[0]][e][0];
                    HeuristicaPasAnterior[maqOrigen[0]][e][0] =
Math.Max(HeuristicaPasAnterior[maqOrigen[0]][e][2],
HeuristicaPasAnterior[maqOrigen[0]][e - 1][1]);
                    HeuristicaPasAnterior[maqOrigen[0]][e][1] =
HeuristicaPasAnterior[maqOrigen[0]][e][1] - a +
HeuristicaPasAnterior[maqOrigen[0]][e][0];
                }
            }
            return HeuristicaPasAnterior;
        }


        public int indexPecaCanvi(int[] maqOrigen, List<List<int[]>> Heuristica, int[]
maquines, int numNivells)
        {
            int index = 0;
            int Fmax = FmaximaOrigen(Heuristica, maqOrigen[0]);
            int lenpeca = 0;
            for (int i = 1; i < Heuristica[maqOrigen[0]].Count(); i++)
```

```
            {
                if (Heuristica[maqOrigen[0]][i][4] > maqOrigen[1])  //mirem que la
peça pugui canviar de nivell
                {
                    List<List<int[]>> HipotesiInicial = CopiarLlista(Heuristica);
                    List<List<int[]>> Hipotesi = canvi1PecaIndex(maqOrigen,
HipotesiInicial, maquines, numNivells, i);
                    if (Fmax < FmaximaOrigen(Heuristica, maqOrigen[0]))   //busquem el
max de Fmax
                    {
                        Fmax = FmaximaOrigen(Heuristica, maqOrigen[0]);
                        index = i;
                    }
                    else if (Fmax == FmaximaOrigen(Heuristica, maqOrigen[0]) & lenpeca
< Heuristica[maqOrigen[0]][i][1] - Heuristica[maqOrigen[0]][i][0])   //busquem el min
de Fmax
                    {
                        lenpeca = Heuristica[maqOrigen[0]][i][1] -
Heuristica[maqOrigen[0]][i][0];  //en cas d'empat, treiem la peça que té la durada més
llarga
                        Fmax = FmaximaOrigen(Heuristica, maqOrigen[0]);
                        index = i;
                    }
                }
            }
            if (index == 0)
            {
                //comprovem que les altres màquines del seu nivell tenen peces que
poden canviar de nivell
                bool baixarNivell = false;
                int e = 0;
                while (e < maquines[maqOrigen[1]] & baixarNivell == false)
                {
                    for (int i = 1; i < Heuristica[e].Count(); i++)
                    {
                        if (Heuristica[e][i][4] > maqOrigen[1])
                        {
                            baixarNivell = true;
                        }
                    }
                    e++;
                }
                if (baixarNivell == false)
                {
                    Console.WriteLine("Programa acabat! no es poden baixar més peces
de nivell 1");
                    return -1;  //donem aquest valor perquè vegi que no hi ha més
solucions
                }
            }

            return index;
        }

        public int indexPecaCanviMin(int[] maqOrigen, List<List<int[]>> Heuristica,
int[] maquines, int numNivells)
        {
            int index = 0;
            int Fmax = FmaximaOrigen(Heuristica, maqOrigen[0]);
            int lenpeca = 0;
            for (int i = 1; i < Heuristica[maqOrigen[0]].Count(); i++)
```

```
            {
                if (Heuristica[maqOrigen[0]][i][4] > maqOrigen[1])
                {
                    List<List<int[]>> HipotesiInicial = CopiarLlista(Heuristica);
                    List<List<int[]>> Hipotesi = canvi1PecaIndex(maqOrigen,
HipotesiInicial, maquines, numNivells, i);
                    if (Fmax > FmaximaOrigen(Heuristica, maqOrigen[0]))
                    {
                        Fmax = FmaximaOrigen(Heuristica, maqOrigen[0]);
                        index = i;
                    }
                    else if (Fmax == FmaximaOrigen(Heuristica, maqOrigen[0]) & lenpeca
< Heuristica[maqOrigen[0]][i][1] - Heuristica[maqOrigen[0]][i][0])
                    {
                        lenpeca = Heuristica[maqOrigen[0]][i][1] -
Heuristica[maqOrigen[0]][i][0];
                        Fmax = FmaximaOrigen(Heuristica, maqOrigen[0]);
                        index = i;
                    }
                }
            }
            if (index == 0)
                //comprovem que les altres màquines del seu nivell tenen peces que
poden canviar de nivell
                bool baixarNivell = false;
                int e = 0;
                while (e < maquines[maqOrigen[1]] & baixarNivell == false)
                {
                    for (int i = 1; i < Heuristica[e].Count(); i++)
                    {
                        if (Heuristica[e][i][4] > maqOrigen[1])
                        {
                            baixarNivell = true;
                        }
                    }
                    e++;
                }
                if (baixarNivell == false)
                {
                    Console.WriteLine("Programa acabat! no es poden baixar més peces
de nivell 1");
                    return -1;  //donem aquest valor perquè vegi que no hi ha més
solucions
                }
            }

            return index;
        }

        public List<int[]> pecesIniI0(List<List<int[]>> LlistaTempsMaquines, int
nivelldesti, int[] maquines)
        {
            int indexmaquina = 0;
            List<int[]> PecesIni2 = new List<int[]>();
            List<int[]> llistaPeces = new List<int[]>();
            for (int i = 0; i < nivelldesti; i++)
            {
                indexmaquina = indexmaquina + maquines[i];
            }
            for (int e = indexmaquina; e < indexmaquina + maquines[nivelldesti]; e++)
//fem un recorregut per trobar les peces que volem
```

```
            {
                for (int r = 1; r < LlistaTempsMaquines[e].Count(); r++)
                {
                    llistaPeces.Add(LlistaTempsMaquines[e][r]);
                }
            }

            for (int i = 0; i < llistaPeces.Count(); i++)
            {
                int[] PecesIni3 = new int[5];
                PecesIni3[0] = llistaPeces[i][2]; //r
                PecesIni3[1] = llistaPeces[i][1] - llistaPeces[i][0]; //p
                PecesIni3[2] = llistaPeces[i][3]; //q
                PecesIni3[3] = llistaPeces[i][4]; //nivell
                PecesIni3[4] = llistaPeces[i][5]; //num peça
                PecesIni2.Add(PecesIni3);
            }
            return PecesIni2;  //ja tenim el nou vector de peces que volem canviar
        }


        public List<List<int[]>> canvi1PecaI0(int[] maqOrigen1, List<List<int[]>>
Heuristica, int[] maquines, int numNivells)// ho fem servir per recalcular
l'heuristica en cas d'empat
        {
            List<List<int[]>> Heuristica2 = CopiarLlista(Heuristica);
            List<int[]> PecesIni = pecesIniI0(Heuristica2, maqOrigen1[1], maquines);
            int marcador = 0;
            for (int i = 0; i < maqOrigen1[1]; i++)
            {
                marcador = marcador + maquines[i];
            }
            int Pmax = 0;
            int IndexPecesIni = 0;
            int index = 0;
            for (int i = 0; i < PecesIni.Count(); i++)
            {
                if (PecesIni[i][3] > maqOrigen1[1])
                {
                    if (Pmax < PecesIni[i][1])
                    {
                        Pmax = PecesIni[i][1];
                        IndexPecesIni = PecesIni[i][4];
                    }
                }
            }
            List<List<int[]>> Heuristica3 = def.HeuristicaInicial(maquines,
numNivells, PecesIni, maqOrigen1[1]);
            int contadorcomodi = 0;
            for (int w = 0; w < Heuristica2.Count(); w++)
            {
                if (Heuristica2[w][0][1] == maqOrigen1[1])
                {
                    int[] memo = Heuristica2[w][0];
                    Heuristica2[w] = Heuristica3[contadorcomodi];
                    Heuristica2[w][0] = memo;
                    contadorcomodi++;
                }
            }
```

```
            for (int e = marcador; e < marcador + maquines[maqOrigen1[1]]; e++)
//treiem la peça de la maquina on sigui i tornem a començar
            {
                for (int r = 1; r < Heuristica2[e].Count(); r++)
                {
                    if (IndexPecesIni == Heuristica2[e][r][5])
                    {
                        index = r;
                        maqOrigen1[0] = e;
                        maqOrigen1[2] = FmaximaOrigen(Heuristica2, e);
                    }
                }
            }
            if (index == 0)
            {
                Console.WriteLine("Programa acabat! no es poden baixar més peces de
nivell 2");
                int[] llista = { 59, 0, 0, 0, 0 };
                List<int[]> llista2 = new List<int[]>();
                llista2.Add(llista);
                List<List<int[]>> llista3 = new List<List<int[]>>();
                llista3.Add(llista2);
                return llista3;
            }
            List<List<int[]>> NovaHeuristica2 = canvi1Peca(maqOrigen1, Heuristica2,
maquines, numNivells, index);
            return NovaHeuristica2;
        }

        public int indexPecaCanviP(int[] maqOrigen, List<List<int[]>> Heuristica,
int[] maquines, int numNivells)
        {
            int index = 0;
            int Fmax = FmaximaOrigen(Heuristica, maqOrigen[0]);
            int P = Heuristica[maqOrigen[0]][0][1] - Heuristica[maqOrigen[0]][0][0];
            for (int i = 1; i < Heuristica[maqOrigen[0]].Count(); i++)
            {
                if (Heuristica[maqOrigen[0]][i][4] > maqOrigen[1])  //mirem que la
peça pugui canviar de nivell
                {
                    if (Heuristica[maqOrigen[0]][i][1] -
Heuristica[maqOrigen[0]][i][0] > P)
                    {
                        P = Heuristica[maqOrigen[0]][i][1] -
Heuristica[maqOrigen[0]][i][0];
                        index = i;
                    }
                }
            }
            return index;
        }

        public List<List<int[]>> CanviPecaEscollida(int[] maqOrigen, List<List<int[]>>
Heuristica, int[] maquines, int numNivells) //fa el canvi de la peça que dona el
millor resultat
        {
            int index = indexPecaCanvi(maqOrigen, Heuristica, maquines, numNivells);
            if (index == 0)  //si maq origen no té suficients peces de nivells
inferiors per baixar, tornem a recalcular l'heurística inicial amb les peces que
queden
            {
```

```
                List<List<int[]>> NovaHeuristica2 = canvi1PecaI0(maqOrigen,
Heuristica, maquines, numNivells);
                return NovaHeuristica2;
            }
            else if (index == -1)
            {
                int[] llista = { 59, 0, 0, 0, 0 };
                List<int[]> llista2 = new List<int[]>();
                llista2.Add(llista);
                List<List<int[]>> llista3 = new List<List<int[]>>();
                llista3.Add(llista2);
                return llista3;
            }
            else
            {
                List<List<int[]>> NovaHeuristica1 = canvi1Peca(maqOrigen, Heuristica,
maquines, numNivells, index);
                return NovaHeuristica1;
            }
        }

        public List<List<int[]>> CanviPecaEscollidaMin(int[] maqOrigen,
List<List<int[]>> Heuristica, int[] maquines, int numNivells) //fa el canvi de la peça
que dona el millor resultat
        {
            int index = indexPecaCanviMin(maqOrigen, Heuristica, maquines,
numNivells);
            if (index == 0)  //si maq origen no té suficients peces de nivells
inferiors per baixar, tornem a recalcular l'heurística inicial amb les peces que
queden
            {
                List<List<int[]>> NovaHeuristica2 = canvi1PecaI0(maqOrigen,
Heuristica, maquines, numNivells);
                return NovaHeuristica2;
            }
            else if (index == -1)
            {
                int[] llista = { 59, 0, 0, 0, 0 };
                List<int[]> llista2 = new List<int[]>();
                llista2.Add(llista);
                List<List<int[]>> llista3 = new List<List<int[]>>();
                llista3.Add(llista2);
                return llista3;
            }
            else
            {
                List<List<int[]>> NovaHeuristica1 = canvi1Peca(maqOrigen, Heuristica,
maquines, numNivells, index);
                return NovaHeuristica1;
            }
        }

        public List<List<int[]>> CanviPecaEscollidaP(int[] maqOrigen,
List<List<int[]>> Heuristica, int[] maquines, int numNivells) //fa el canvi de la peça
que dona el millor resultat
        {
            int index = indexPecaCanviP(maqOrigen, Heuristica, maquines, numNivells);
            if (index == 0)  //si maq origen no té suficients peces de nivells
inferiors per baixar, tornem a recalcular l'heurística inicial amb les peces que
queden
            {
```

```
                List<List<int[]>> NovaHeuristica2 = canvi1PecaI0(maqOrigen,
Heuristica, maquines, numNivells);
                return NovaHeuristica2;
            }
            else if (index == -1)
            {
                int[] llista = { 59, 0, 0, 0, 0 };
                List<int[]> llista2 = new List<int[]>();
                llista2.Add(llista);
                List<List<int[]>> llista3 = new List<List<int[]>>();
                llista3.Add(llista2);
                return llista3;
            }
            else
            {
                List<List<int[]>> NovaHeuristica1 = canvi1Peca(maqOrigen, Heuristica,
maquines, numNivells, index);
                return NovaHeuristica1;
            }
        }

    }
```

D.3. Codi algorisme 1A

Aquest programa llegeix totes les mostres dels exemplars de peces i presenta les solucions de totes elles que obté l'Heurística 1A.

```
class Programa
    {
        public static void Main()  //programa que treballa a partir de les mostres.
        {

            Definicio ob = new Definicio(); //creem l'objecte
            Arborescent arb = new Arborescent();
            llegir lec = new llegir();



            //
            //
            string fitxer = "EX100(200)5-3B.txt";
            string nomfitxer = "Exemple1.txt";
            int grandaria_mostra = 100;
            int exemplars = 200;
            //
            //




            string[] lines = lec.llegir2(fitxer);
            int numNivells = lec.numeroNivells(lines);
            int[] maquinesss = lec.llistaMaquines(lines);
            int inici = 5; //
            using (System.IO.StreamWriter file = new
System.IO.StreamWriter(@"C:/Users/osent/Documents/ETSEIB/Master/TFM/C#/ConsoleApplicat
ionComparar/ConsoleApplicationComparar/bin/Debug/" + nomfitxer)) { }
```

```
            var watch = System.Diagnostics.Stopwatch.StartNew(); //activem el rellotge
aquí per calcular el temps total de resolució de tots els exemplars

            for (int mostra = 0; mostra < exemplars; mostra++)
            {
                int fi = inici + grandaria_mostra;
                Solucio solucio1 = new Solucio();  //obrim la classe solució
                List<Solucio> Solucions = new List<Solucio> { }; //creem una llista
per guardar totes les solucions -- en cada mostra de solucions la llista començarà des
de 0
                int[] maquinaanterior = {0,0,0}; //ho farem servir per marcar les
peces
                var watch2 = System.Diagnostics.Stopwatch.StartNew();  //activem el
segon rellotge per calcular el temps que utilitza en cada mostra
                List<int[]> PecesIni = lec.peces(lines, inici, fi);
                List<List<int[]>> solucioB = ob.HeuristicaInicial(maquinesss,
numNivells, PecesIni, 0);
                ob.escriureFaseB(solucioB);

                int Fmax = arb.Fmaxima(solucioB);
                //ho escrivim a un TXT
                using (System.IO.StreamWriter file = new
System.IO.StreamWriter(@"C:/Users/osent/Documents/ETSEIB/Master/TFM/C#/ConsoleApplicat
ionComparar/ConsoleApplicationComparar/bin/Debug/" + nomfitxer, true))
                {
                    file.WriteLine(Fmax);
                }
                solucio1.penalitzacio = 0;
                solucio1.sequencia = solucioB;
                solucio1.Fmax = Fmax;
                Solucions.Add(solucio1);


                int[] maquinaorigen1 = arb.maqOrigen(solucioB);
                maquinaanterior = maquinaorigen1; //assignem la maquina anterior
                Console.WriteLine();
                Console.WriteLine("maquina origen:  " + maquinaorigen1[0]);

                int i = 0;
                while (maquinaorigen1[1] < numNivells - 1)
                {
                    Console.WriteLine();
                    List<List<int[]>> Solucio = arb.CanviPecaEscollida(maquinaorigen1,
solucioB, maquinesss, numNivells);
                    solucioB = arb.CopiarLlista(Solucio);
                    if (solucioB[0][0][0] == 59) //si no podem baixar cap peça de
nivell perquè no en queden sortim de l'iteració i continuem amb la següent mostra
                    {
                        break;
                    }
                    Fmax = arb.Fmaxima(Solucio);
                    maquinaorigen1 = arb.maqOrigen(solucioB);
                    Solucio solucio2 = new Solucio();
solucio2.penalitzacio = i + 1;
                    solucio2.sequencia = Solucio;
                    solucio2.Fmax = Fmax;
                    if (maquinaanterior[1] != maquinaorigen1[1]) //si el nivell origen
cambia marquem la peça
                    {
                        solucio2.marcat = true;
                    }
```

```
                    maquinaanterior = maquinaorigen1;
                    Solucions.Add(solucio2);
                    using (System.IO.StreamWriter file = new
System.IO.StreamWriter(@"C:/Users/osent/Documents/ETSEIB/Master/TFM/C#/ConsoleApplicat
ionComparar/ConsoleApplicationComparar/bin/Debug/"+nomfitxer, true))
                    {
                       file.WriteLine(Fmax);
                    }
                    Console.WriteLine();
                    Console.WriteLine();
                    Console.WriteLine("Penalització = " + (i + 1) + "             ");
                    ob.escriureFaseB(Solucio);
                    Console.WriteLine();
                    Console.WriteLine("maquina origen:  " + maquinaorigen1[0]);
                    i++;

                }
                //calculem la qualitat
                float eficiencia1 = solucio1.eficiencia(Solucions);
                long temps = watch2.ElapsedMilliseconds;
                watch2.Stop(); //parem el rellotge parcial
                using (System.IO.StreamWriter file = new
System.IO.StreamWriter(@"C:/Users/osent/Documents/ETSEIB/Master/TFM/C#/ConsoleApplicat
ionComparar/ConsoleApplicationComparar/bin/Debug/" + nomfitxer, true))
                {
                    file.WriteLine("-     "+temps);
                }

                Console.WriteLine();
                Console.WriteLine();

                inici = fi + 2;
            }
            watch.Stop();
            long temps2 = watch.ElapsedMilliseconds;
            Console.ReadKey();
            System.Environment.Exit(1);
        }
```

D.4. Codi algorisme 1B

Aquest programa llegeix totes les mostres dels exemplars de peces i presenta les solucions de totes elles que obté l'Heurística 1B.

```
class Programa
    {
        public static void Main        {

            Definicio ob = new Definicio();
            Arborescent arb = new Arborescent();
            llegir lec = new llegir();




            //
            //
            string fitxer = "EX020-3B.txt";
```

```
            string nomfitxer = "Exemple2.txt";
            int grandaria_mostra = 20;
            int exemplars = 1000;
            //
            //




            string[] lines = lec.llegir2(fitxer);
            int numNivells = lec.numeroNivells(lines);
            int[] maquinesss = lec.llistaMaquines(lines);
            int inici = 5;

            using (System.IO.StreamWriter file = new
System.IO.StreamWriter(@"C:/Users/osent/Documents/ETSEIB/Master/TFM/C#/ConsoleApplicat
ionComparar/ConsoleApplicationComparar/bin/Debug/" + nomfitxer)) { }
            var watch = System.Diagnostics.Stopwatch.StartNew();

            for (int mostra = 0; mostra < exemplars; mostra++)
            {
                int fi = inici + grandaria_mostra;
                Solucio solucio1 = new Solucio();
                List<Solucio> Solucions = new List<Solucio> { };
                int[] maquinaanterior = { 0, 0, 0 };
                var watch2 = System.Diagnostics.Stopwatch.StartNew();
                List<int[]> PecesIni = lec.peces(lines, inici, fi);
                List<List<int[]>> solucioB = ob.HeuristicaInicial(maquinesss,
numNivells, PecesIni, 0);
                ob.escriureFaseB(solucioB);
                int Fmax = arb.Fmaxima(solucioB);
                using (System.IO.StreamWriter file = new
System.IO.StreamWriter(@"C:/Users/osent/Documents/ETSEIB/Master/TFM/C#/ConsoleApplicat
ionComparar/ConsoleApplicationComparar/bin/Debug/" + nomfitxer, true))
                {
                    file.WriteLine(Fmax);
                }
                solucio1.penalitzacio = 0;
                solucio1.sequencia = solucioB;
                solucio1.Fmax = Fmax;
                Solucions.Add(solucio1);
                //-------

                int[] maquinaorigen1 = arb.maqOrigen(solucioB);
                maquinaanterior = maquinaorigen1;
                Console.WriteLine();
                Console.WriteLine("maquina origen:  " + maquinaorigen1[0]);

                int i = 0;
                while (maquinaorigen1[1] < numNivells - 1)
                {
                    Console.WriteLine();
                    List<List<int[]>> Solucio = arb.CanviPecaEscollida(maquinaorigen1,
solucioB, maquinesss, numNivells);
                    solucioB = arb.CopiarLlista(Solucio);
                    if (solucioB[0][0][0] == 59)
                    {
                        break;
                    }
                    Fmax = arb.Fmaxima(Solucio);
                    maquinaorigen1 = arb.maqOrigen(solucioB);
```

```
                    Solucio solucio2 = new Solucio();
                    solucio2.penalitzacio = i + 1;
                    solucio2.sequencia = Solucio;
                    solucio2.Fmax = Fmax;
                    if (maquinaanterior[1] != maquinaorigen1[1])
                    {
                        solucio2.marcat = true;
                    }
                    maquinaanterior = maquinaorigen1;
                    Solucions.Add(solucio2);

                    using (System.IO.StreamWriter file = new
System.IO.StreamWriter(@"C:/Users/osent/Documents/ETSEIB/Master/TFM/C#/ConsoleApplicat
ionComparar/ConsoleApplicationComparar/bin/Debug/" + nomfitxer, true))
                    {
                        file.WriteLine(Fmax);
                    }
                    Console.WriteLine();
                    Console.WriteLine();
                    Console.WriteLine("Penalització = " + (i + 1) + "             ");
                    ob.escriureFaseB(Solucio);
                    Console.WriteLine();
                    Console.WriteLine("maquina origen:  " + maquinaorigen1[0]);
                    i++;
                }
                float eficiencia1 = solucio1.eficiencia(Solucions);
                long temps = watch2.ElapsedMilliseconds;
                watch2.Stop();
                using (System.IO.StreamWriter file = new
System.IO.StreamWriter(@"C:/Users/osent/Documents/ETSEIB/Master/TFM/C#/ConsoleApplicat
ionComparar/ConsoleApplicationComparar/bin/Debug/" + nomfitxer, true))
                {
                    file.WriteLine("-     "+temps);
                }

                Console.WriteLine();
                Console.WriteLine();

                inici = fi + 2;
            }
            watch.Stop();
            long temps2 = watch.ElapsedMilliseconds;
            Console.ReadKey();
            System.Environment.Exit(1);
        }
    }
```

D.5. Codi algorisme 2A

Aquest programa llegeix totes les mostres dels exemplars de peces i presenta les solucions de totes elles que obté l'Heurística 2A.

```
class Programa
    {
        public static void Main()
        {

            Definicio ob = new Definicio();
            Arborescent arb = new Arborescent();
            llegir lec = new llegir();
```

```
            string fitxer = "EX100(200)5-3B.txt";
            string nomfitxer = "Exemple3.txt";
            int grandaria_mostra = 100;
            int exemplars = 200;



            string[] lines = lec.llegir2(fitxer);
            int numNivells = lec.numeroNivells(lines);
            int[] maquinesss = lec.llistaMaquines(lines);
            int inici = 5;

            using (System.IO.StreamWriter file = new
System.IO.StreamWriter(@"C:/Users/osent/Documents/ETSEIB/Master/TFM/C#/ConsoleApplicat
ionComparar/ConsoleApplicationComparar/bin/Debug/" + nomfitxer)) { }
            var watch = System.Diagnostics.Stopwatch.StartNew();

            for (int mostra = 0; mostra < exemplars; mostra++)
            {
                int fi = inici + grandaria_mostra;
                Solucio solucio1 = new Solucio();
                List<Solucio> Solucions = new List<Solucio> { };
                int[] maquinaanterior = { 0, 0, 0 };
                var watch2 = System.Diagnostics.Stopwatch.StartNew();
                List<int[]> PecesIni = lec.peces(lines, inici, fi);
                List<List<int[]>> solucioB = ob.HeuristicaInicial(maquinesss,
numNivells, PecesIni, 0);
                ob.escriureFaseB(solucioB);

                int Fmax = arb.Fmaxima(solucioB);

                using (System.IO.StreamWriter file = new
System.IO.StreamWriter(@"C:/Users/osent/Documents/ETSEIB/Master/TFM/C#/ConsoleApplicat
ionComparar/ConsoleApplicationComparar/bin/Debug/" + nomfitxer, true))
                {
                    file.WriteLine(Fmax);
                }

                solucio1.penalitzacio = 0;
                solucio1.sequencia = solucioB;
                solucio1.Fmax = Fmax;
                Solucions.Add(solucio1);


                int[] maquinaorigen1 = arb.maqOrigen(solucioB);
                maquinaanterior = maquinaorigen1;
                Console.WriteLine();
                Console.WriteLine("maquina origen:  " + maquinaorigen1[0]);

                int i = 0;
                while (maquinaorigen1[1] < numNivells - 1)
                {
                    Console.WriteLine();
                    List<List<int[]>> Solucio =
arb.CanviPecaEscollidaMin(maquinaorigen1, solucioB, maquinesss, numNivells);
                    solucioB = arb.CopiarLlista(Solucio);
                    if (solucioB[0][0][0] == 59)
                    {
```

```
                        break;
                    }
                    Fmax = arb.Fmaxima(Solucio);
                    maquinaorigen1 = arb.maqOrigen(Solucio);

                    Solucio solucio2 = new Solucio;
                    solucio2.penalitzacio = i + 1;
                    solucio2.sequencia = Solucio;
                    solucio2.Fmax = Fmax;
                    if (maquinaanterior[1] != maquinaorigen1[1])
                    {
                        solucio2.marcat = true;
                    }
                    //hi posem el backtracking si ens la marca
                    if (solucio2.marcat == true)
                    {
                        List<List<int[]>> solucioC =
arb.CopiarLlista(Solucions[i].sequencia);
                        List<List<int[]>> SolucioDif =
arb.CanviPecaEscollidaP(maquinaanterior, solucioC, maquinesss, numNivells);
                        solucioB = arb.CopiarLlista(SolucioDif);
                        solucioC = arb.CopiarLlista(SolucioDif);
                        List<List<int[]>> solucioD =
arb.CopiarLlista(solucio2.sequencia);
                        int[] maquinaorigen2 = arb.maqOrigen(solucioD);
                        if (solucioB[0][0][0] == 59)
                        {
                            break;
                        }
                        Fmax = arb.Fmaxima(SolucioDif);
                        maquinaorigen1 = arb.maqOrigen(SolucioDif);
                        Solucio solucio3 = new Solucio();
solucio3.penalitzacio = i + 1;
                        solucio3.sequencia = SolucioDif;
                        solucio3.Fmax = Fmax;

                        List<List<int[]>> SolucioPrimera =
arb.CanviPecaEscollidaMin(maquinaorigen1, solucioC, maquinesss, numNivells); //mirem
com seria la seguent iteració a partir de la fase 2
                        List<List<int[]>> SolucioSegona =
arb.CanviPecaEscollidaMin(maquinaorigen2, solucioD, maquinesss, numNivells); //mirem
com seria la seguent iteració a partir de la fase 1
                        int FmaxPrimera = arb.Fmaxima(SolucioPrimera);
                        int FmaxSegona = arb.Fmaxima(SolucioSegona);

                        if (FmaxPrimera < FmaxSegona) //si el segon nivell es millor
canviem
                        {
                            Solucions.Add(solucio3);
                        }
                        else //si el segon nivell no és millor, ho deixem igual
                        {
                            maquinaorigen1 = maquinaorigen2;
                            Solucions.Add(solucio2);
                        }
                    }
                    else
                    {
                        Solucions.Add(solucio2);
                    }
                    maquinaanterior = maquinaorigen1;
```

```
                    using (System.IO.StreamWriter file = new
System.IO.StreamWriter(@"C:/Users/osent/Documents/ETSEIB/Master/TFM/C#/ConsoleApplicat
ionComparar/ConsoleApplicationComparar/bin/Debug/" + nomfitxer, true))
                    {
                        file.WriteLine(Fmax);
                    }
                    Console.WriteLine();
                    Console.WriteLine();
                    Console.WriteLine("Penalització = " + (i + 1) + "             ");
                    ob.escriureFaseB(Solucio);
                    Console.WriteLine();
                    Console.WriteLine("maquina origen:  " + maquinaorigen1[0]);
                    i++;
                }

                float eficiencia1 = solucio1.eficiencia(Solucions);

                long temps = watch2.ElapsedMilliseconds;
                watch2.Stop();

                using (System.IO.StreamWriter file = new
System.IO.StreamWriter(@"C:/Users/osent/Documents/ETSEIB/Master/TFM/C#/ConsoleApplicat
ionComparar/ConsoleApplicationComparar/bin/Debug/" + nomfitxer, true))
                {
                    file.WriteLine("-     "+temps);
                }
                Console.WriteLine();
                Console.WriteLine();

                inici = fi + 2;
            }
            watch.Stop();
            long temps2 = watch.ElapsedMilliseconds;

            Console.ReadKey();
            System.Environment.Exit(1);
        }
    }
```

D.6. Codi algorisme 2B

Aquest programa llegeix totes les mostres dels exemplars de peces i presenta les solucions de totes elles que obté l'Heurística 2B.

```
class Programa
    {
        public static void Main()
        {

            Definicio ob = new Definicio();
            Arborescent arb = new Arborescent();
            llegir lec = new llegir();



            string fitxer = "EX100(200)5-3B.txt";
            string nomfitxer = "Exemple3.txt";
            int grandaria_mostra = 100;
            int exemplars = 200;
```

```
            string[] lines = lec.llegir2(fitxer);
            int numNivells = lec.numeroNivells(lines);
            int[] maquinesss = lec.llistaMaquines(lines);
            int inici = 5;

            using (System.IO.StreamWriter file = new
System.IO.StreamWriter(@"C:/Users/osent/Documents/ETSEIB/Master/TFM/C#/ConsoleApplicat
ionComparar/ConsoleApplicationComparar/bin/Debug/" + nomfitxer)) { }
            var watch = System.Diagnostics.Stopwatch.StartNew();

            for (int mostra = 0; mostra < exemplars; mostra++)
            {
                int fi = inici + grandaria_mostra;
                Solucio solucio1 = new Solucio();
                List<Solucio> Solucions = new List<Solucio> { };
                int[] maquinaanterior = { 0, 0, 0 };
                var watch2 = System.Diagnostics.Stopwatch.StartNew();
                List<int[]> PecesIni = lec.peces(lines, inici, fi);
                List<List<int[]>> solucioB = ob.HeuristicaInicial(maquinesss,
numNivells, PecesIni, 0);
                ob.escriureFaseB(solucioB);

                int Fmax = arb.Fmaxima(solucioB);

                using (System.IO.StreamWriter file = new
System.IO.StreamWriter(@"C:/Users/osent/Documents/ETSEIB/Master/TFM/C#/ConsoleApplicat
ionComparar/ConsoleApplicationComparar/bin/Debug/" + nomfitxer, true))
                {
                    file.WriteLine(Fmax);
                }

                solucio1.penalitzacio = 0;
                solucio1.sequencia = solucioB;
                solucio1.Fmax = Fmax;
                Solucions.Add(solucio1);


                int[] maquinaorigen1 = arb.maqOrigen(solucioB);
                maquinaanterior = maquinaorigen1;
                Console.WriteLine();
                Console.WriteLine("maquina origen:  " + maquinaorigen1[0]);

                int i = 0;
                while (maquinaorigen1[1] < numNivells - 1)
                {
                    Console.WriteLine();
                    List<List<int[]>> Solucio = arb.CanviPecaEscollida(maquinaorigen1,
solucioB, maquinesss, numNivells);
                    solucioB = arb.CopiarLlista(Solucio);
                    if (solucioB[0][0][0] == 59)
                    {
                        break;
                    }
                    Fmax = arb.Fmaxima(Solucio);
                    maquinaorigen1 = arb.maqOrigen(Solucio);

                    Solucio solucio2 = new Solucio;
                    solucio2.penalitzacio = i + 1;
                    solucio2.sequencia = Solucio;
```

```
                    solucio2.Fmax = Fmax;
                    if (maquinaanterior[1] != maquinaorigen1[1])
                    {
                        solucio2.marcat = true;
                    }
                    //hi posem el backtracking si ens la marca
                    if (solucio2.marcat == true)
                    {
                        List<List<int[]>> solucioC =
arb.CopiarLlista(Solucions[i].sequencia);
                        List<List<int[]>> SolucioDif =
arb.CanviPecaEscollidaP(maquinaanterior, solucioC, maquinesss, numNivells);
                        solucioB = arb.CopiarLlista(SolucioDif);
                        solucioC = arb.CopiarLlista(SolucioDif);
                        List<List<int[]>> solucioD =
arb.CopiarLlista(solucio2.sequencia);
                        int[] maquinaorigen2 = arb.maqOrigen(solucioD);
                        if (solucioB[0][0][0] == 59)
                        {
                            break;
                        }
                        Fmax = arb.Fmaxima(SolucioDif);
                        maquinaorigen1 = arb.maqOrigen(SolucioDif);
                        Solucio solucio3 = new Solucio();
solucio3.penalitzacio = i + 1;
                        solucio3.sequencia = SolucioDif;
                        solucio3.Fmax = Fmax;

                        List<List<int[]>> SolucioPrimera =
arb.CanviPecaEscollidaMin(maquinaorigen1, solucioC, maquinesss, numNivells); //mirem
com seria la seguent iteració a partir de la fase 2
                        List<List<int[]>> SolucioSegona =
arb.CanviPecaEscollidaMin(maquinaorigen2, solucioD, maquinesss, numNivells); //mirem
com seria la seguent iteració a partir de la fase 1
                        int FmaxPrimera = arb.Fmaxima(SolucioPrimera);
                        int FmaxSegona = arb.Fmaxima(SolucioSegona);

                        if (FmaxPrimera < FmaxSegona) //si el segon nivell es millor
canviem
                        {
                            Solucions.Add(solucio3);
                        }
                        else //si el segon nivell no és millor, ho deixem igual
                        {
                            maquinaorigen1 = maquinaorigen2;
                            Solucions.Add(solucio2);
                        }
                    }
                    else
                    {
                        Solucions.Add(solucio2);
                    }
                    maquinaanterior = maquinaorigen1;
                    using (System.IO.StreamWriter file = new
System.IO.StreamWriter(@"C:/Users/osent/Documents/ETSEIB/Master/TFM/C#/ConsoleApplicat
ionComparar/ConsoleApplicationComparar/bin/Debug/" + nomfitxer, true))
                    {
                        file.WriteLine(Fmax);
                    }
                    Console.WriteLine();
                    Console.WriteLine();
```

```
                    Console.WriteLine("Penalització = " + (i + 1) + "             ");
                    ob.escriureFaseB(Solucio);
                    Console.WriteLine();
                    Console.WriteLine("maquina origen:  " + maquinaorigen1[0]);
                    i++;
                }

                float eficiencia1 = solucio1.eficiencia(Solucions);

                long temps = watch2.ElapsedMilliseconds;
                watch2.Stop();

                using (System.IO.StreamWriter file = new
System.IO.StreamWriter(@"C:/Users/osent/Documents/ETSEIB/Master/TFM/C#/ConsoleApplicat
ionComparar/ConsoleApplicationComparar/bin/Debug/" + nomfitxer, true))
                {
                    file.WriteLine("-     "+temps);
                }
                Console.WriteLine();
                Console.WriteLine();

                inici = fi + 2;
            }
            watch.Stop();
            long temps2 = watch.ElapsedMilliseconds;

            Console.ReadKey();
            System.Environment.Exit(1);
        }
    }
```