An Update on the Comparison of MIP, CP and Hybrid Approaches for Mixed Resource Allocation and Scheduling DETAILED EXPERIMENTAL RESULTS

Philippe Laborie

IBM, 9 rue de Verdun 94250 Gentilly, France
laborie@fr.ibm.com

Abstract. This is an appendix of the paper *An Update on the Comparison of MIP, CP and Hybrid Approaches for Mixed Resource Allocation and Scheduling* [1] that details the experimental results.

1 Introduction

All experiments were obtained on an IBM blade HS22 with 20GB RAM and an Intel® Xeon® X5570 2.93GHz running GNU/Linux 2.6.32 with a CP Optimizer V12.7.1 search using 4 parallel workers.

2 Comparison with State-of-the-art Techniques on Existing Benchmark

The CP Optimizer model used on Table 1 in [1] is the model given in the article with a few parameter changes as explained in the paper. It is shown on Fig. 1. Detailed results obtained when running this model on all the 335 instances of the benchmark are given on Tables 1-4 for the different families.

3 Results on new benchmark instances without precedence constraints

The cost of the best solutions found by CP Optimizer (using a search parameter to switch off the temporal linear relaxation see model on Fig. 2) with a 1h time-limit on all the 100 new instances is shown on Table 5. Note that the only difference between the two models is the search parameter at line 13. Solutions proved to be optimal are marked with a star '*'.

4 Results on new benchmark instances with precedence constraints

The same variant as in previous section (model on Fig. 2) was used on the new instances with precedence constraints. Detailed results are shown on Table 6.

```
using CP;
tuple ModeData { int p; int c; int f; }
tuple JobData { int r; int d; }
int n = ...; range J = 1..n;
int m = ...; range I = 1..m;
ModeData M[I,J] = ...;
JobData Job[J] = ...;
int C[I] = ...;

execute {
  cp.param.TimeLimit = 3600;
  cp.param.Workers = 4;
  cp.param.FailureDirectedSearchEmphasis = 3.5;
}

dvar interval job[j in J] in Job[j].r..Job[j].d;
dvar interval mode[i in I][j in J] optional size M[i][j].p;

minimize sum(i in I, j in J) (M[i][j].f * presenceOf(mode[i][j]));
subject to {
  forall(j in J) { alternative(job[j], all(i in I) mode[i][j]); }
  forall(i in I) { sum(j in J) pulse(mode[i][j],M[i][j].c) <= C[i]; }
}
```

Fig. 1. Complete CP Optimizer model used for comparison with state-of-the-art approaches on the existing benchmark

```
using CP;
tuple ModeData { int p; int c; int f; }
tuple JobData { int r; int d; }
int n = ...; range J = 1..n;
int m = ...; range I = 1..m;
ModeData M[I,J] = ...;
JobData Job[J] = ...;
int C[I] = ...;

execute {
  cp.param.TimeLimit = 3600;
  cp.param.Workers = 4;
  cp.param.TemporalRelaxation = "Off";
}

dvar interval job[j in J] in Job[j].r..Job[j].d;
dvar interval mode[i in I][j in J] optional size M[i][j].p;

minimize sum(i in I, j in J) (M[i][j].f * presenceOf(mode[i][j]));
subject to {
  forall(j in J) { alternative(job[j], all(i in I) mode[i][j]); }
  forall(i in I) { sum(j in J) pulse(mode[i][j],M[i][j].c) <= C[i]; }
}
```

Fig. 2. Complete CP Optimizer model for new instances

References

1. Laborie, P.: An Update on the Comparison of MIP, CP and Hybrid Approaches for Mixed Resource Allocation and Scheduling. In: Proc. CP-AI-OR 2018 (2018)

Table 1. Optimal costs and computation times (in s) for family 'c'.

		k=1		k=2		k=3		k=4		k=5	
m	n	opt	*time*	opt	*time*	opt	*time*	opt	*time*	opt	*time*
2	10	204	*0.03*	169	*0.02*	206	*0.23*	178	*0.09*	232	*0.10*
2	12	203	*0.08*	223	*0.22*	245	*0.07*	190	*0.06*	208	*0.13*
2	14	227	*0.24*	257	*0.29*	258	*0.14*	262	*0.47*	228	*0.04*
2	16	326	*0.46*	276	*0.79*	289	*0.81*	299	*0.71*	272	*0.45*
2	18	334	*0.89*	353	*1.39*	329	*2.19*	334	*1.28*	358	*2.17*
2	20	408	*3.29*	341	*0.93*	342	*1.29*	360	*1.69*	311	*0.79*
2	22	365	*6.48*	409	*2.22*	373	*2.05*	439	*4.89*	417	*5.18*
2	24	391	*6.64*	472	*14.09*	449	*5.21*	392	*2.39*	431	*6.41*
2	26	459	*25.25*	470	*7.98*	479	*11.47*	454	*9.13*	469	*6.78*
2	28	481	*10.86*	475	*6.87*	519	*16.41*	510	*43.02*	538	*19.14*
2	30	529	*166.16*	565	*45.88*	535	*87.60*	575	*410.60*	583	*17.73*
2	32	614	*37.72*	627	*291.02*	616	*144.07*	585	*375.87*	512	*33.71*
2	34	541	*36.49*	631	*60329*	526	*9.43*	609	*97.08*	611	*507064*
2	36	644	*134.08*	697	*288151*	664	*39.69*	647	*328.66*	678	*586829*
2	38	634	*17023*	716	*74.60*	673	*102.79*	617	*23.81*	627	*31.55*
3	10	272	*0.20*	264	*0.07*	247	*0.08*	274	*0.03*	291	*0.04*
3	12	309	*0.06*	288	*0.21*	232	*0.06*	317	*1.13*	318	*0.39*
3	14	328	*0.56*	358	*0.34*	300	*0.41*	353	*0.34*	294	*0.19*
3	16	378	*0.91*	406	*1.00*	309	*0.52*	355	*0.74*	332	*0.31*
3	18	408	*0.56*	488	*1.21*	354	*1.65*	411	*1.59*	472	*2.42*
3	20	481	*0.99*	443	*1.24*	419	*1.41*	402	*4.12*	405	*1.26*
3	22	512	*2.91*	479	*2.98*	446	*3.24*	473	*2.45*	504	*2.93*
3	24	568	*12.21*	515	*4.96*	558	*2.99*	471	*1.69*	595	*12.42*
3	26	555	*7.96*	493	*5.84*	531	*21.45*	585	*11.29*	564	*8.01*
3	28	573	*20.87*	505	*8.98*	573	*9.55*	583	*17.00*	575	*22.38*
3	30	679	*61.77*	620	*21.55*	617	*33.94*	604	*49.74*	744	*173.00*
3	32	664	*36.43*	600	*21.99*	702	*225.22*	734	*912.59*	666	*93.55*
4	10	229	*0.02*	263	*0.05*	247	*0.05*	301	*0.03*	277	*0.09*
4	12	335	*0.39*	309	*0.17*	372	*0.02*	300	*0.05*	352	*0.12*
4	14	417	*0.58*	364	*0.27*	328	*1.04*	388	*0.53*	407	*0.25*
4	16	406	*0.61*	420	*0.54*	407	*0.37*	447	*0.31*	381	*1.23*
4	18	444	*0.79*	448	*1.35*	509	*3.18*	466	*1.16*	457	*0.72*
4	20	506	*0.67*	444	*1.03*	511	*1.79*	446	*1.12*	472	*0.87*
4	22	558	*2.56*	501	*1.75*	529	*2.75*	532	*1.82*	534	*2.59*
4	24	684	*10.44*	632	*2.70*	591	*3.70*	625	*7.76*	510	*1.83*
4	26	710	*9.77*	618	*24.12*	612	*11.18*	635	*13.33*	624	*7.07*
4	28	681	*7.13*	601	*3.02*	734	*25.68*	637	*10.78*	639	*10.95*
4	30	723	*23.21*	668	*17.18*	726	*56.22*	925	*277.96*	742	*48.11*
4	32	717	*14.84*	757	*60.72*	686	*53.42*	835	*40.92*	716	*71.64*

Table 2. Optimal costs and computation times (in s) for family 'de'.

		k=1		k=2		k=3		k=4		k=5	
m	n	opt	*time*	opt	*time*	opt	*time*	opt	*time*	opt	*time*
3	10	316	*0.03*	inf	*0.00*	inf	*0.00*	284	*0.02*	inf	*0.00*
3	12	338	*0.03*	289	*0.02*	inf	*0.03*	inf	*0.01*	335	*0.03*
3	14	362	*0.01*	319	*0.06*	407	*0.17*	375	*0.04*	397	*0.11*
3	16	372	*0.02*	412	*0.09*	445	*0.13*	459	*0.03*	403	*0.12*
3	18	500	*0.08*	484	*0.08*	502	*0.24*	499	*0.30*	412	*0.03*
3	20	492	*0.18*	465	*0.63*	549	*0.53*	473	*0.30*	462	*0.07*
3	22	555	*0.37*	503	*0.35*	526	*0.14*	520	*0.20*	514	*0.03*
3	24	546	*0.89*	593	*0.46*	589	*0.66*	497	*0.19*	634	*0.25*
3	26	625	*1.02*	572	*1.73*	654	*0.54*	600	*0.99*	706	*0.76*
3	28	653	*2.34*	657	*0.37*	709	*0.39*	785	*8.16*	730	*0.35*

Table 3. Optimal costs and computation times (in s) for family 'df'.

		k=1		k=2		k=3		k=4		k=5	
m	n	opt	*time*	opt	*time*	opt	*time*	opt	*time*	opt	*time*
3	14	366	*0.06*	355	*0.02*	370	*0.03*	377	*0.03*	403	*0.03*
3	16	394	*0.02*	427	*0.03*	412	*0.01*	402	*0.02*	381	*0.03*
3	18	486	*0.04*	472	*0.07*	503	*0.10*	442	*0.04*	444	*0.03*
3	20	499	*0.07*	436	*0.18*	474	*0.12*	433	*0.06*	433	*0.03*
3	22	535	*0.12*	469	*0.23*	539	*0.35*	510	*0.07*	511	*0.19*
3	24	653	*0.53*	560	*0.38*	611	*0.73*	550	*0.03*	581	*0.17*
3	26	626	*0.32*	639	*1.70*	636	*0.07*	593	*0.26*	660	*2.52*
3	28	685	*0.28*	704	*0.65*	641	*0.52*	649	*0.06*	648	*0.30*

Table 4. Optimal costs and computation times (in s) for family 'e'.

		k=1		k=2		k=3		k=4		k=5	
m	n	opt	*time*	opt	*time*	opt	*time*	opt	*time*	opt	*time*
2	10	243	*0.02*	331	*0.14*	280	*0.07*	314	*0.03*	298	*0.09*
2	12	321	*0.05*	313	*0.02*	264	*0.01*	372	*0.89*	367	*0.61*
3	15	400	*0.47*	392	*0.15*	410	*0.35*	365	*0.05*	388	*0.10*
4	20	475	*0.58*	495	*4.42*	456	*0.38*	471	*0.36*	440	*0.72*
5	25	575	*0.45*	553	*0.23*	555	*0.95*	601	*0.82*	560	*1.48*
6	30	627	*1.75*	666	*1.69*	651	*1.04*	658	*3.24*	644	*1.50*
7	35	741	*3.75*	767	*4.96*	728	*6.39*	752	*12.17*	748	*15.00*
8	40	883	*15.86*	843	*41.58*	888	*180.08*	838	*12.87*	869	*31.93*
9	45	1002	*58.92*	1052	*629.09*	1024	*133.01*	1001	*42.37*	1021	*203.74*
10	50	1096	*63.89*	1076	*49.40*	1051	*87.72*	1088	*77.67*	1084	*904.32*

Table 5. Best cost found by CP Optimizer for the new instances without precedence constraints.

n	m	k=1	k=2	k=3	k=4	k=5
20	2	1541*	1896*	2365*	1079*	1593*
30	2	2159	2028	2378	1724	2026*
40	2	2655	2945	2703	2160	2374
50	2	3144	3743	3175	2759	3122
50	5	2611	2907	2818	2739	2685
100	2	6975	7737	5849	6598	6342
100	5	5119	5209	5305	5076	5245
100	10	5232	5338	5177	5104	5179
200	2	12373	14788	11320	13714	11851
200	5	10281	10781	10350	10249	10317
200	10	10810	10650	10575	10598	10255
200	20	10531	10550	10961	10418	10809
500	2	31958	36295	36285	31975	33196
500	5	27549	28034	26874	27298	28023
500	10	28751	28640	27919	28315	28426
500	20	27600	27989	27796	28408	27928
1000	2	68097	75231	70369	70399	67596
1000	5	59869	61358	60305	60615	61281
1000	10	66266	65252	63556	66201	63429
1000	20	65154	65550	65558	62239	65157

Table 6. Best cost found by CP Optimizer for the new instances with precedence constraints.

n	m	k=1	k=2	k=3	k=4	k=5
20	2	1545*	1925*	2365*	1079*	1594*
30	2	2159*	2043*	2378*	1754	2063*
40	2	2669	2952	2716	2197	2385
50	2	3144	3789	3246	2759	3205
50	5	2654	2961	2997	2836	2879
100	2	7021	8321	6146	6994	6474
100	5	5212	5320	5302	5146	5227
100	10	5491	5379	5394	5206	5248
200	2	12962	15289	11600	14810	13173
200	5	10799	10921	10739	10731	10603
200	10	11005	10875	10858	10886	10406
200	20	10817	10946	10941	10651	10829
500	2	33978	41262	39437	38011	37891
500	5	27181	28648	27966	27366	28230
500	10	27794	27353	27316	28145	27154
500	20	27006	28089	27755	27840	27791
1000	2	80475	100250	81371	82621	89117
1000	5	70047	72136	72672	68214	72473
1000	10	68241	64212	70871	74719	71933
1000	20	64506	64653	65426	64308	65880