Mini-task report: SDC with simulated annealing

Ludwig Meysel, Mitja Stachowiak

1 Introduction

The task was to implement a simulated annealing approach (SA) for SDC (system of difference constraints). LPsolve is used to get a schedule for a given set of constraints. The SA-algorithm mutates the order of the constraints to reduce the number of clock cycles of the schedule.

2 Resources and Constraints

The constraints consist of data flow - and resource constraints. The data flow constraints determine, that no operation must start, before all predecessors have obtained a result. The resource constraints prevent, that the same resource is used twice at the same time.

2.1 Resources

Each hardware has a certain amount of resources and resource types. There is a fixed list of operations given in the framework:

Operation name (ar)	delay	weight
MEM	2	9.0
ADD	1	1.0
SUB	1	1.4
MUL	4	2.3
DIV	18	4.3
SH	1	2.0
AND	1	2.0
OR	1	2.0
CMP	1	2.1
OTHER	1	1.0
SLACK	1	0.0

Each resource type can support multiple operations. For this project, the resource(types) are assumed to be overlap-free:

$$\neg\exists R_1, R_2 \in Resources; Op_1, Op_2 \in Operations : Op_1 \in R_1 \wedge Op_1 \in R_2 \wedge Op_2 \in R_1 \wedge Op_2 \notin R_2$$

Each resource can handle one operation within a certain time (delay). Multiple resources of the same type may exist.

2.2 Data Flow Constraints

The data flow constraints are fixed and only need to be computed once.

2.3 Resource Constraints

The data flow constraints are fixed and only need to be computed once.

3 Simulated Annealing

The principal structure of any simulated annealing looks like this:

```
S = RandomConfiguration();
T = InitialTemperature();
while (ExitCriterion()==false) {
  while (InnerLoopCriterion() == false) {
    $S_{new}$ = Generate(S);
    $\Delta$ C = Cost($S_{new}$)-Cost(S);
    r = random(0,1);
    if (r < $e^{-\Delta C/T}$) S = $S_{new}$
  }
  T = updateTemperature();
}
```

The implementation is located in scheduler/SASDC.java:schedule. The parameters are:

- *Random Configuration* ...
- *Initial Temperature* is determined by applying n(nodes) random changes and saving the costs of each change. T is then $20 * standardDeviation(costs)$.
- *Exit Criterion* is the condition, when the simulated annealing should stop. For each temperature, the number of applied changes and the number of accepted changes is counted. When less then 12% of the changes are accepted, the algorithm stops.
- *Update Temperature* decreases T by a factor tu, which depends on the acceptance ratio as well:

acceptance ratio (ar)	temperature factor (tu)
> 96%	0.5
96 .. 80%	0.9
80 .. 15%	0.95
< 15%	0.8

- *Inner Loop Criterion* determines, how many changes are tested for the same temperature. Each change usually moves one node in the ordering of constraint-equations. The larger the number of nodes becomes, the more often each node should be moved, so the number of iterations should depend on the node count. Further more, there is a quality factor $\in [1..10]$ for the algorithm, which can be passed via the third program argument. The formula $n_{inner} = \left\lceil quality * n_{nodes}^{4/3} \right\rceil$ is known to yield a result, thats quality belongs to the given quality.

4 Results

File	Number of Nodes	schedule length ASAP	schedule length ALAP	schedule length SA/SDC	Quality factor of SA	Number of Iterations	Runtime / s
ADPCMn-decode-271-381	27	64,8	56,0	31,9	1	20494	29,39
ADPCMn-decode-271-381	27	64,8	56,0	30,5	5	82216	113,38
ADPCMn-decode-271-381	27	64,8	56,0	30,5	10	144181	185,55
ADPCMn-decode-425-472	12	23,8	23,8	34,2	1	85	0,11
ADPCMn-decode-425-472	12	23,8	23,8	21,8	5	7591	6,37
ADPCMn-decode-425-472	12	23,8	23,8	21,8	10	11001	9,13
ADPCMn-decode-524-553	7	25,4	26,4	25,4	1	15	0,02
ADPCMn-decode-524-553	7	25,4	26,4	25,4	5	68	0,07
ADPCMn-decode-524-553	7	25,4	26,4	25,4	10	135	0,13
ADPCMn-decode-559-599	9	26,1	27,2	28,2	1	20	0,03
ADPCMn-decode-559-599	9	26,1	27,2	25,1	5	4137	3,26
ADPCMn-decode-559-599	9	26,1	27,2	28,2	10	377	0,35
ADPCMn-decode-631-729	23	44,8	40,0	33,3	1	5017	5,81
ADPCMn-decode-631-729	23	44,8	40,0	24,3	5	13121	14,65
ADPCMn-decode-631-729	23	44,8	40,0	22,9	10	108076	120,56
ADPCMn-decode-771-791	5	13,5	13,5	22,5	1	10	0,01
ADPCMn-decode-771-791	5	13,5	13,5	22,5	5	44	0,04
ADPCMn-decode-771-791	5	13,5	13,5	22,5	10	87	0,08
ADPCMn-decode-803-832	8	18,8	18,8	20,2	1	17	0,02
ADPCMn-decode-803-832	8	18,8	18,8	20,2	5	81	0,08
ADPCMn-decode-803-832	8	18,8	18,8	20,2	10	161	0,16
AESrkgcyclic	24	24,4	23,4	37,4	1	421	0,61
AESrkgcyclic	24	24,4	23,4	32,4	5	13881	18,84
AESrkgcyclic	24	24,4	23,4	30,4	10	28414	38,14
BLAKE256Digest-processBlock-160-230	21	57,4	58,4	32,4	1	10905	12,11
BLAKE256Digest-processBlock-160-230	21	57,4	58,4	22,4	5	58291	64,11
BLAKE256Digest-processBlock-160-230	21	57,4	58,4	31,4	10	175161	194,06
BLAKE256Digest-processBlock-189-1577	308	414,9	128,7	110,9	1	6244	241,63
BLAKE256Digest-processBlock-189-1577	308	414,9	128,7	97,9	5	31204	1236,87
ContrastFilter-filter-13-252	47	72,0	67,7	50,8	1	4251	9,49
ContrastFilter-filter-13-252	47	72,0	67,7	37,7	5	24622	54,00
ContrastFilter-filter-13-252	47	72,0	67,7	33,2	10	101821	218,61
ECOH256Digest-AES2RoundsAll-2-666	179	262,3	174,4	58,1	1	175567	2733,60

The table above compares the results of simple ASAP / ALAP-Schedules with the results of the implemented simulated annealing algorithm.

5 Conclusion