lea.schoenberger [☺] tu-dortmund.de
nick.pietrass [☺] tu-dortmund.de

Exercises for
Cyber-Physical Systems Fundamentals
Summer Semester 19

Exercise Sheet 6

(18 Points)

Please note: Solutions to theory assignments must be submitted (individually or in pairs) until 08.07.2019 at 12:00 AM (post box in OH16, ground floor, in front of room E16). Submitting solutions via mail is *not* possible. Discussion: 10.-11.07.2019.

1 Harmonic Task Systems (2 Points)

Consider the following periodic tasks with implicit deadlines:

	τ_1	τ_2	τ_3	τ_4	τ_5	τ_6	τ_7
C_i	0.2	2	2	1.5	1	14	28.8
T_i	2	6	12	24	24	72	288
D_i	2	6	12	24	24	72	288
U_i	0.1	$\frac{1}{3}$	$\frac{1}{6}$	0.0625	0.0417	0.195	0.1

Assume that the tasks are executed on a uniprocessor system.

(a) Determine formally if the rate-monotonic (RM) schedule is feasible.

(b) Determine formally if the earliest deadline first (EDF) schedule is feasible.

2 Real-Time Calculus - Theory (2 Points)

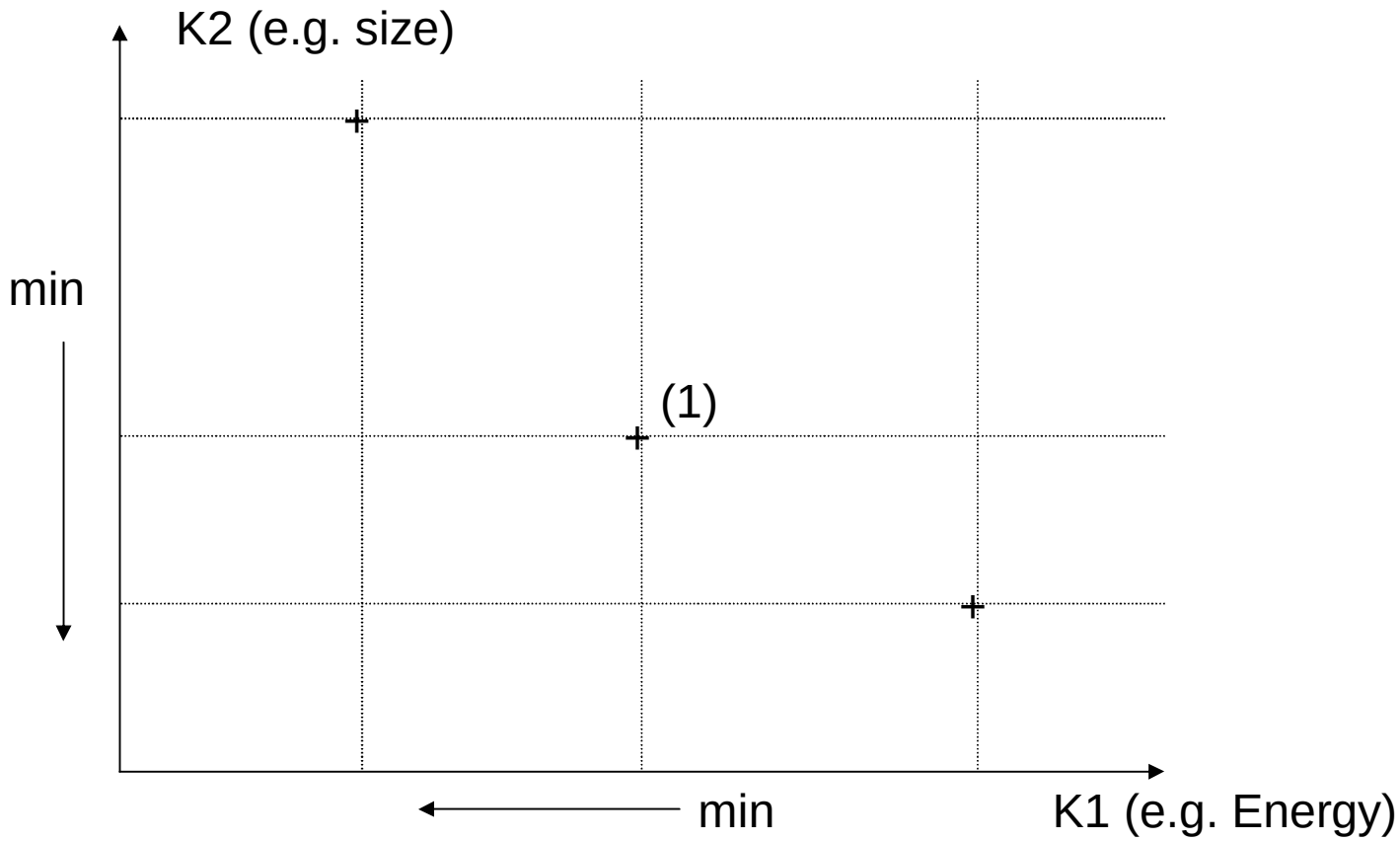
Consider a stream of events, where at the end of each period p an event burst occurs. At the begin of each period, two events with distance d arrive. How do the *arrival curves* for the **maximum** number of events within a time window Δ look like for this stream of events? Draw the curve for the interval $[0..3p]$.

3 TDMA - Theory (2 Points)

Consider a TDMA bus with a transmission rate of b. How do the *service curves* look like for one participant to which the bus is assigned for s time units within each period p?

4 Pareto Optimization - Theory (2 Points)

The following diagram reflects the evaluation of designs with respect to multiple criteria. We assume that we would like to maximize one criteria and minimize the other one. For example, consider an energy harvester to maximize the generated energy with the minimum required space of solar panels or wind wheels. Please indicate the region which is dominated by design (1) (the region in which designs are “inferior” to design (1)). Also, indicate the region in which designs would dominate design (1) (the region in which designs are “superior” to design (1)).



5 Step 3: Scratchpad Allocation and Function Outlining (5 Points)

Execute the command `cd ~/wcet/step3` to open the directory used for this assignment. In the file `test.c`, some preparations have been made to transform the inner loops of the functions `Initialize` and `Sum` into separate functions via function outlining. In this manner, it is possible to swap out the inner functions to the scratchpad memory (SPM) individually.

Perform the function outlining of the inner loops of the aforementioned both functions.

Thereon, compile the program by executing the command **tricore-gcc -o test.elf -g -T ../tc1796.lds test.c** and load it into the `a3tricore` analyzer as explained in the previous exercise sheet. If you start an aiT analysis, some errors will be reported.

Correct the errors in the file `a.ais`, so that the analysis can be performed properly.

How did the WCET change compared to the previous version (complete functions in the SPM)?

Hint: For the previous version, a WCET of 85117 cycles has been computed.

6 Step 4: Integer Linear Programming (5 Points)

aiT performs the WCET analysis by analyzing each basic block of a program separately, constructing an ILP (integer linear program) based on these results and solving it.

In the directory `step4`, more precisely, in the file `example.lp`, an exemplary formulation of an ILP can be found. Open this file with a text editor of your choice and solve it with an ILP solver via `lp_solve example.lp`.

Another example is given in the file `example2.lp`, where the control flow graph of a simple program is modeled. Solve

this problem via `lp_solve example2.lp`.

Explain the result.

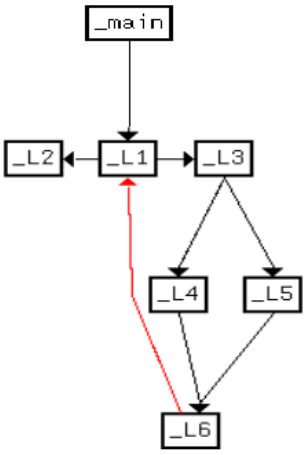
How can we model a certain basic block with execution time of more than one cycle in the ILP?

Here, the following program containing a loop is given:

```
int main()
{
  int i, j = 0;

  _Pragma( "loopbound min
                100 max 100" );
  for ( i = 0; i < 100; i++ ) {
    if ( i < 50 )
      j += i;
    else
      j += ( i * 13 ) % 42;
  }

  return j;
}
```



Block	Cycles
main	21
L1	27
L2	20
L3	2
L4	2
L5	20
L6	13

Abbildung 1: Example program with loop.

The basic structure of this program is already modeled in the file `ipet.lp`.

Add the number of cycles for each basic block.

If you try to solve the ILP via `lp_solve ipet.lp`, the solver will abort with the error message *This problem is unbounded*.

Why is it impossible for the ILP solver to find a solution?

Add the missing constraint for edge h.

If you try again to solve the corrected file with lp_solve, you should obtain a result.
Why does the solver compute a solution in which L4 is never executed?