Supplemental Material

for the Paper **„Randomized Problem-Relaxation Solving for Over-Constrained Schedules"** submitted to KR 2021

Explanations to Remarks (a) -- (e) at the end of Section 2:

Remark (a):

CLAIM: Solution quality wrt. utility will monotonically increase throughout the solving process.

JUSTIFICATION: The best solution is maintained by the framework and only updated if a solution with a better utility is found. Hence, over time, the current best solution can only remain the same or become a better one.

Remark (b):

CLAIM: Our approach is complete, i.e., it will yield a JOP solution given sufficient time and memory, and a full-cycle random number generator.

JUSTIFICATION: Let the (seeded) function that shuffles the job set be injective, i.e., yield a different job order for any two different seeds given as an input. Let the pseudo random number generator have the (full-cycle) property to generate all numbers in its range without repetition. In other words, no number can be generated for the second time until all numbers have been generated for the first time. Using the sequence of numbers generated by such a random number generator as seeds for the function that shuffles the job set, we will obtain a different order of the job set in every iteration. Consequently, since the found JMP solution is uniquely determined by the order of the job set given as an input to the MSMP algorithm, we will finally obtain all JMP solutions in the process. Since every JOP solution is also a JMP solution, the claim follows.

Remark (c):

CLAIM: Each module in our framework is viewed as a black-box and can be realized by different algorithms; this allows our approach to profit from latest research advancements in the JSSP and MSMP fields.

JUSTIFICATION: Since we neither make assumptions about the internals of the JSSP solver, nor about the internals of the MSMP algorithm, other than assuming that the former solves JSSPs and the latter MSMP problems, the claim follows.

Remark (d):

CLAIM: No information exchange is required between different iterations; thus, our approach enables efficient multi-threaded implementations.

JUSTIFICATION: Clearly, multi-threaded applications are more efficient if no communication is necessary between the worker threads.

Remark (e):

CLAIM: Our approach does not require to manually adapt the CP encoding of the given JSSP instance to a JOP encoding.

JUSTIFICATION: This claim follows directly from the description of the solution process using our framework (cf. Fig. 2 and page 3, left column, last-but-one paragraph).

Pseudocode (adaptation of a CP encoding of JSSP to a JOP encoding)

(this illustration supplements the discussion of the encoding principle given in the second paragraph of Sec. 3 in the paper)

```
for each job j:
    var_j = new integer variable with lower = 0 and upper = 1;
    add constraint:
      if var_j == 1 then last operation of j must end before time limit;
endfor

...

maximize (sum of var_j over all jobs j);
```