Prof. Dr. C. Stamm christoph.stamm@fhnw.ch

Solution 6

1 Task Dependency Graph Mapping

a) In Exercise 3.3, you created a task-dependency graph for 14 tasks for an LU decomposition. Now create the same graph in software as an additional example for the optimal task mapper. Run the software and compute a solution for the given LU decomposition task-dependency graph both for $p = 3$ and $p = 4$ processes and compare it with your manual solutions.

p0	A1	B2	C3	G7	I9	M13	N14
p1		D4	E5	H8	K11		
p2			F6	J10	L12		

p0	A1	B2	C3	G7	I9	M13	N14
p1		D4	E5	H8	K11		
p2			F6	J10	L12		
p3							

Since 3 processes are enough for an optimal mapping, the algorithm does not map any task to process P_3 in the mapping for $p = 4$.

b) The script suggests branch-and-bound as a possible improvement to the DFS algorithm. Complete the given solution template with your DFBB implementation and compare the runtime for the same scheduling problems as in task a).

	$p = 3$	$p = 4$
DFS	0.10 s	4.35 s
DFBB	0.0007 s	0.003 s

The provided program also contains a nearly finished implemented parallel version of DFS. What is still missing is the implementation of the work stealing algorithm. It is your task to implement in the method `stealWorkFrom(...)` the empty body of the declared lambda expression. Inside of the lambda expression the `DFSearcher` with the empty *open* list should remove every p^{th} vertex from the *open* list of the given `DFSearcher` *s* and moves these vertices into its own *open* list. To move elements from one list to another, the function `list::splice(...)` is excellent.
Compare the runtime for the same scheduling problems as in task a).

	$p = 3$	$p = 4$
DFS	0.10 s	4.35 s
DFBB	0.0007 s	0.003 s
Parallel DFS	0.1 s	0.002 s

c) For what reason is only every p^{th} element stolen from the open list of the other `DFSearcher` and not every second one?

At the beginning of the parallel execution, a single `DFSearcher` starts with the root vertex. This one `DFSearcher` works alone for some time until its open list contains enough open vertices. Only then the other `DFSearchers` start stealing work from the first `DFSearcher`. Since only one `DFSearcher` has work and $p - 1$ want work, it makes sense that each `DFSearcher` steals only every p^{th} element.