Optimizing Workflow in Cell-Based Slaughtering and Cutting of Pigs

Johan Oppen

Optimizing workflow in cell-based slaughtering and cutting of pigs

Johan Oppen

Møreforsking Molde, 6410 Molde, Norway
johan.oppen@himolde.no

Abstract. In this paper we describe and solve a scheduling problem taken from the slaughterhouse industry. In an on-going research project, a new concept for slaughtering and cutting of pigs is developed. The idea is to replace the traditional production line with a number of *meat factory cells*, where an operator slaughters and rough-cuts the pig, assisted by a robot. In order to minimize non-productive operator time, a solution approach is presented together with computational results.

Keywords: Production Planning · Scheduling · Optimization · Meat Industry · Meat Factory Cell.

1 Introduction

The meat industry, like most other industries, is constantly searching for decreased costs and increased revenues. Cost savings have mainly been achieved through centralization, with fewer, larger and more specialized and automated factories and increased transport distances. This has resulted in very large plants with a slaughter capacity of up to 1 400 pigs per hour in the Netherlands [2], and 5 000 smallstock (lambs/sheep) per day in Bordertown, Australia [1]. Also in Norway, many small slaughterhouses have been closed down and replaced by larger and more automated plants during the last decades. However, long distances and relatively few livestock in Norway makes it impossible to decrease the number and increase the size of slaughterhouses further. The transportation time for livestock from many farms to the closest slaughterhouse is already close to eight hours, which is the longest time allowed for transportation of live animals in Norway. This means that a more efficient meat production must be sought in other ways than by replacing relatively small plants by larger ones with increased capacity and more automation.

Nortura [3], which is the largest meat company in Norway, is currently running a research project called Meat 2.0 to find out if a new way of organizing slaughtering and rough-cutting can give benefits in terms of increased flexibility, improved hygiene and better utilization of edible offals, so-called *plus products*, without losing efficiency compared to today's assembly line-like production system. The basic idea is to replace the traditional production line, where each

worker performs a small operation before the carcass continues to the next station, with a number of *Meat Factory Cells*, where one animal at the time is slaughtered and rough-cut by an operator assisted by a robot.

In the project, parts of the Meat Factory Cell concept is tested on pigs at Nortura's plant in Tønsberg, Norway. The focus in this paper is how the work in the cell is organized, so we will just give a brief overview of the whole process before looking at the operations in the cell and how these can be optimized.

The pigs are first stunned, put to death, emptied for blood, skalded, dehaired and disinfected. Up to this point, the process is the same as in today's production line system. Each pig is then transported to a cell, where an operator, assisted by a robot, slaughters and rough-cuts the pig and places the parts on a rack or trolley. When all the parts are on the rack, the rack is transported via meat control to sorting and further processing of the parts. The equipment in the cell is cleaned, and the next pig is placed in the cell for processing.

The equipment and operations in the cell are still being developed and tested, but it is already clear that both an operator and a machine or robot will be working in the cell, so we have enough information to look at how the work in the cell should be optimized. Each operator will be working in more than one, most likely two, cells in parallel. This is to avoid unnecessary waiting when the robot works in the cell, as the operator must keep a certain distance to a working robot. In addition to choosing which cell to work in at a given time, the operator can also sometimes choose between different operations to carry out, this adds to the complexity of the planning problem. The time needed for the operator to complete all operations to slaughter and rough-cut each pig is assumed to be constant and known, so the goal is to minimize the amount of time the operator spends waiting for the robot and moving between cells.

The problem has many similarities to *scheduling problems*, see, e.g. [5], as we are looking for the best ordering or sequence of operations or tasks. In most scheduling problems in food processing [4], "industry-specific characteristics induce specific and complex scheduling problems" [4, p. 1]. This is also the case here, and we have not been able to find similar problems in the literature. The option for the operator to stay in the current cell and wait for the robot to finish its current task, or move to another cell, seems to be unique. We have tried a few modeling and solution approaches from the operations research literature to handle the problem.

The remainder of the paper is organized as follows. The problem is presented in more detail in Section 2. A heuristic solution method is described in Section 3. Computational testing is described in Section 4, followed by conclusions in Section 5.

2 Problem description

When a pig arrives in the cell, it is placed on a table and kept in place by four grippers, each of them holding one leg. The operator cuts off the pig's head and

the four legs, the parts are lifted onto a rack by the robot. The grippers "hand over" the legs to the robot.

After the head and the legs have been cut off and removed, the pig is turned around on the table, now with the back up. The operator uses a saw to open the carcass by performing one cut on each side of the spine, the robot then lifts the neck/back over to the rack. While the robot is lifting, the operator cuts the back/neck free from the carcass. This means that in this so-called *combined task*, the operator is assisted by the robot, and the robot is active while the operator is working in the cell. During the other robot operations the operator has to keep a certain distance to the robot and can therefore not do operations on the pig.

Next, the operator removes the intestines (heart, lungs, liver, stomach etc), which are also placed on the rack. When all the inner organs are removed, the robot lifts the belly part over to the rack. The table is then cleaned before the next pig arrives.

An overview of operator, robot and combined tasks is given in Table 1. The times needed to perform the different tasks, and to move between cells, are supposed to be known. The times given in Table 1 are preliminary estimates which will become more precise as more testing is performed in the project.

Table 1. Operations in the cell

Task	**Description**	**Predecessors**	**Est. time**
Ot1	Place pig on table		60 sec
Ot2	Cut off head	Ot1	20 sec
Ot3	Cut off legs	Rt1	100 sec
Ot4	Turn pig on table	Rt2	20 sec
Ot5	Saw both sides of back	Ot4	40 sec
Ot6	Remove intestines	Ct1	120 sec
Ot7	Clean table	Rt3	90 sec
Om	Move between cells		15 sec
Rt1	Lift head to rack	Ot2	20 sec
Rt2	Lift legs to rack	Ot3	60 sec
Rt3	Lift belly to rack	Ot6	20 sec
Ct1	Cut loose and lift back/neck	Ot5	30 sec

Operators can work in multiple (normally two) cells at the time, and must choose what to do while the robot works. In general, the operator will wait in the cell if the robot task is short, and go to the neighboring cell and work there if the robot task is long. Depending on the duration of tasks and when the operator moves between cells, it may happen that the robot is still working when the operator returns to a cell, forcing the operator to wait. The time estimates given in Table 1 will not lead to such situations, as the longest robot task (except the task where the operator and the robot works together) does not last longer than the shortest operator task. We nevertheless want to consider this as a

possibility in case time estimates change, and because time estimates may be different for different animal types. If the project shows that the Meat Factory Cell concept should be used in production, both bovine and smallstock may also be slaughtered using this method.

3 Modeling and solution approaches

When studying a new and unknown optimization problem, we have often found it useful to start by writing down a mathematical model and try to find optimal solutions using traditional algorithms. This was also the first approach for the problem presented here. The model was a mixed integer model with indicator variables $\alpha_{otpct'p'c'}$ equal to 1 if operator o performs task t' on pig p' in cell c' immediately after performing task t on pig p in cell c, and 0 otherwise. The model also has variables telling when the operator starts performing the different tasks, and variables for moving and waiting times between operations. The constraints in the model ensure that all tasks are performed in the correct order, and that all time variables are computed correctly. The model is relatively easy to understand and straightforward to write down, but it turns out to be very hard to solve by standard solvers like Gurobi, and not even the smallest instances included in the computational experiments in Section 4 could be solved to optimality in reasonable time. We therefore do not present this model in detail.

3.1 A construction heuristic

Because solving a standard mathematical model turned out to be impossible in reasonable time for realistically sized instances, a heuristic approach seemed natural. We have developed a simple heuristic which finds good solutions in short time for problem instances of reasonable size. The solution method builds schedules for one operator working in two cells in parallel, and works as follows.

- An initially empty vector P of partial schedules is constructed. An empty schedule is created, the first operator task in cell 1 is added, cell 1 is set as the current cell (the cell where the operator is currently staying), and the schedule is added to P.
- Whenever P is not empty, a partial schedule is picked from P, extended in all feasible ways, the resulting schedules are then put back into P. When all tasks in both cells are added to a schedule, the completion time is compared to the best so far, and the schedule with the earliest completion time is kept.
- When a partial schedule is extended, an operator task, a robot task or a combined task is added to the schedule. The previous task in the current cell must have the new task as a successor, and all predecessors of the new task must already be added for the current cell. If the added task is an operator task or a combined task, the current cell does not change.
- If the added task is a robot task, the current cell may change. If the robot task lasts no longer than the time it takes for the operator to move to the

other cell, the operator stays in the cell and thus the current cell is not changed. If the robot task lasts longer than the operator move time, the time it takes before the operator can start working in the current or the other cell is computed and compared.

- If the difference is at most equal to the operator move time, both changing and not changing the current cell is considered by making two copies of the schedule.
- If the difference is larger, the alternative where the operator can start working first is chosen.

4 Computational experiments

In order to find out how well the heuristic performes, both in terms of run time and solution quality, we have conducted computational experiments. In the following subsections, we describe the test instances used and the test results.

4.1 Test instances

We have created 60 test instances, based on five different problem sizes. The smallest instances have three operator tasks, two robot tasks and one combined task, and are small enough to allow us to check that solutions are correct, and even optimal, by inspection. The largest instances have more operator and robot tasks than we consider in the problem presented here, in addition there are up to five successor tasks. Instance 9 represents the real-world problem described in Section 2 with tasks listed in Table 1. For each problem size, we create instances with two different sets of task durations, one where most operator tasks are longer than the longest robot task (instances with odd numbered IDs), and one with shorter operator tasks and longer robot tasks (instances with even numbered IDs). Each instance is then replicated with one and two pigs per cell, and with three different operator move times. An overview of the instances is given in Table 2, and a graphical view of instance 3 is given in Figure 1.

4.2 Results

The heuristic outlined in Section 3 was implemented in C++, and tests are run on an Intel(R) Core(TM) i7-4600U CPU @ 2.10 Ghz 2.70 Ghz with 8.00 GB of RAM. Results are shown in Table 3.

From the results in Table 3, it is evident that the real-world instances (instance 9 and 10) are easy to solve, even if the number of tasks is quite high. Especially instance 7 and 8 are hard to solve, these have both the largest number of tasks and the largest complexity in terms of many possible successors for some of the tasks. This means the number of possible paths through the network of tasks is quite high, and thus finding the best one takes time. Even though the problem studied here normally does not have that many options, we find it useful

Table 2. Problem instances. "Inst" refers to instance numbers, the columns under "Size" give the number of operator, robot and combined tasks, and the maximum number of successors for a task, respecitively. The ranges for task durations are given in the columns under "Duration", and in the last two columns we give the total operator and robot time needed per pig.

Inst	Size				Duration			Time per pig	
	Op tasks	R tasks	C tasks	Succ	Op tasks	R tasks	C tasks	Operator	Robot
1	3	2	1	2	6-16	2-10	12	46	20
2	3	2	1	2	2-10	6-20	12	34	40
3	5	2	1	3	6-16	2-10	14	64	30
4	5	2	1	3	2-10	6-20	14	38	42
5	6	3	1	3	6-16	2-10	14	74	36
6	6	3	1	3	2-10	6-20	14	46	52
7	9	6	1	5	6-16	2-10	14	112	46
8	9	6	1	5	2-10	6-20	14	90	74
9	7	3	1	1	4-24	4-12	6	90	20
10	7	3	1	1	2-16	8-16	6	58	32

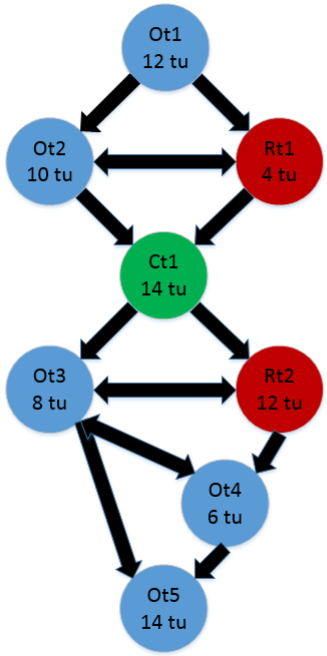

Fig. 1. Instance 3 with five operator tasks (in blue), two robot tasks (in red), and one combined task (in green). Durations for each task are given, and the arrows indicate possible orderings of tasks. In this instance, Ot3 has the highest number of possible successors (three).

Table 3. Test results. Explanation of column headers: "Instance" refer to instances, where the three numbers refer to intance number, the number of pigs processed in each cell and the operator move time, respectively. "Op time" is the total operator time needed to perform the operator and combined tasks. "Obj" is the best solution found in terms of completion time. "Move" and "Wait" are the number of time units the operator spends moving and waiting in the best found solution, and "Time" is the time (in seconds) the algorithm needs to solve the problem. A * in the time column means the algorithm ran out of memory before it completed, and the solution given is the best solution found before this occured.

Instance	Op time	Obj	Move	Wait	Time	Instance	Op time	Obj	Move	Wait	Time
1-1-1	92	97	5	0	0.00	2-1-1	68	81	5	8	0.00
1-1-2	92	102	6	4	0.00	2-1-2	68	84	10	6	0.00
1-1-5	92	111	15	4	0.00	2-1-5	68	93	25	0	0.00
1-2-1	184	193	9	0	0.00	2-2-1	136	155	9	10	0.00
1-2-2	184	202	10	8	0.00	2-2-2	136	160	18	6	0.00
1-2-5	184	217	25	8	0.00	2-2-5	136	181	45	0	0.00
3-1-1	128	133	5	0	0.00	4-1-1	76	91	5	10	0.00
3-1-2	128	138	10	0	0.00	4-1-2	76	94	10	8	0.00
3-1-5	128	151	15	8	0.00	4-1-5	76	103	25	2	0.00
3-2-1	256	265	9	0	0.01	4-2-1	152	171	9	10	0.01
3-2-2	256	274	18	0	0.01	4-2-2	152	178	18	8	0.01
3-2-5	256	297	25	16	0.01	4-2-5	152	199	45	2	0.01
5-1-1	148	155	7	0	0.01	6-1-1	92	103	7	4	0.01
5-1-2	148	162	14	0	0.02	6-1-2	92	108	14	2	0.02
5-1-5	148	181	25	8	0.01	6-1-5	92	127	35	0	0.02
5-2-1	296	309	13	0	8.91	6-2-1	184	201	13	4	8.97
5-2-2	296	322	26	0	14.63	6-2-2	184	212	26	2	18.20
5-2-5	296	357	45	16	6.18	6-2-5	184	249	65	0	25.80
7-1-1	224	237	13	0	36.10	8-1-1	180	203	11	12	21.56
7-1-2	224	250	18	8	30.10	8-1-2	180	212	26	6	49.25
7-1-5	224	275	35	16	16.21	8-1-5	180	245	65	0	108.47
7-2-1	448	475	23	4	*	8-2-1	360	407	21	26	*
7-2-2	448	498	34	16	*	8-2-2	360	424	46	18	*
7-2-5	448	545	65	32	*	8-2-5	360	487	115	12	*
9-1-1	192	199	7	0	0.01	10-1-1	128	137	7	2	0.00
9-1-2	192	206	14	0	0.01	10-1-2	128	142	14	0	0.01
9-1-5	192	223	15	16	0.01	10-1-5	128	163	35	0	0.01
9-2-1	384	397	13	0	0.02	10-2-1	256	273	13	4	0.01
9-2-2	384	410	26	0	0.02	10-2-2	256	282	26	0	0.01
9-2-5	384	441	25	32	0.02	10-2-5	256	321	65	0	0.01

to test some instances with more successors, as there may be other application of this problem with a different structure.

It is not obvious that running instances with two pigs in each cell is needed, as we expect the same pattern of movements between cells and waiting for the robot to be repeated, but we wanted to see how much harder the instances become with two pigs. In addition, it may be beneficial to use different cutting patterns to get more flexibility for how the carcasses are utilized, and thus the ordering and duration of tasks may change from one pig to the next, and between cells.

From Table 3, we also see that the time spent moving and waiting varies with different move times and different task durations. For the odd numbered instances with long operator tasks and short robot tasks (the left half of Table 3), waiting occurs when moving takes more time than waiting for the robot task to complete, and this never happens with short move times. For the even numbered instances with short operator tasks and long robot tasks (the right half of Table 3), waiting may occur also with short move times, as it may happen that waiting is needed when the operator comes back after working in the other cell, because the robot may still be working and thus forces the operator to wait.

5 Conclusions

We have described and solved a scheduling problem in cell-based slaughtering and rough-cutting of pigs, based on a new concept which is currently being tested in a research project. Using standard software to solve a traditional mathematical model is not feasible, so we have developed a simple heuristic to solve the problem. Computational testing shows that this method gives good results in very short time for realistically sized problem instances.

Acknowledgements

This work is funded by the Norwegian Research Council, grant 256266.

References

1. Jbs bordertown plant (2013), http://www.jbssa.com.au/ourfacilities/processingfacilities/Bordertown/default.aspx, accessed 3. Jan 2019
2. F-line marel meat (2019), https://marel.com/meat-processing/systems-and-equipment/slaughter-robotization/pig-slaughter-line-robotization/f-line/1763, accessed 3. Jan 2019
3. Nortura (2019), http://www.nortura.no/, accessed 3. Jan 2019
4. Akkerman, R., Donk, D.P.V.: Analyzing scheduling in the food-processing industry: structure and tasks. Cognition Technology and Work **11**(3), 215–226 (Sep 2009)
5. Nahmias, S.: Production and Operations Analysis. McGraw-Hill, 6 edn. (2009)