Cutting Optimal Sections from Production Foils

Michael Kirchhof, Oliver Meyer, Stefan Mähr, Wolfgang Braunwarth and Claus Weihs

Abstract Rechargeable Lithium-Ion battery cell production is one of the most important processes in the field of electro mobility. The batteries' electrodes are produced in the form of long coated foils which are then cut into pieces of a predefined length called electrode sheets. The production process of the coated foils consists of several sequential process steps and quality parameters are measured frequently along the foil after each sub-process. We aim at determining the maximum number of electrode sheets that can be built from a produced foil of a certain length, with respect to given quality requirements. In a second step, we introduce an algorithm originated from the 0-1 multi-objective knapsack problem that is able to efficiently determine the optimal positions of the sheets based on all observed quality parameters.

Michael Kirchhof · Oliver Meyer · Claus Weihs
Computational Statistics, TU Dortmund University, 44221 Dortmund, Germany
✉ michael.kirchhof@tu-dortmund.de
✉ meyer@statistik.tu-dortmund.de
✉ weihs@statistik.tu-dortmund.de

Stefan Mähr · Wolfgang Braunwarth
Zentrum für Sonnenenergie- und Wasserstoff-Forschung Baden-Württemberg, 89081 Ulm, Germany
✉ stefan.maehr@zsw-bw.de
✉ wolfgang.braunwarth@zsw-bw.de

Archives of Data Science, Series A
(Online First)
KIT Scientific Publishing
Vol. 5, No. 1, 2018

DOI: 10.5445/KSP/1000087327/23
ISSN 2363-9881

1 Introduction

In many industrial applications, products are not individually manufactured, but extracted from larger production items, e.g. pieces cut from a roll of foil. In the age of Industry 4.0, modern producing machines keep track of various quality parameters of these lines. These gathered data offer an opportunity to diminish waste and improve the products' quality by determining how many individual products that comply with certain quality thresholds can be cut from the production line and where the optimal cutting positions are found.

One of the use cases in which the quality of the cut products gives a huge competitive advantage is the production of long coated electrode foils. From coated foils of several hundred meters in length with varying quality, short electrodes are cut and assembled into Rechargeable Lithium-Ion Batteries. The parameters of the coated foil, like the uniformity of the eletrolyte or the residual humidity, influence the electro-chemical characteristics of the finished batteries.

As a member of the battery research cluster ProZell, we focus on the optimization of the production process of such coated foils. We found that the just described problem demands an algorithmic solution that operates efficiently even for fast consecutive measurements and many quality parameters with different significances for the electrodes' energy density.

In this paper, we present two algorithms that jointly compute the optimal cutting positions. The first algorithm determines the maximum number of sheets that can be cut from a given production line, with respect to upper and lower specification limits for the observed quality parameters, by proceeding over the line from the start to the end and picking sheets as soon as possible. As this does not yield the highest quality sheets, a second algorithm originating from the 0-1 multi-objective knapsack problem further reduces the set of possible sheet selections by improving the sheets with respect to all quality parameters.

In the following chapter, the optimization problem and its constraints are defined. Next, the algorithm that identifies the maximum sheet count is presented. Therafter, the second algorithm and its parallels to the knapsack problem are discussed. Also, a runtime study is given to prove the efficiency even in situations with large amounts of data. In the fifth chapter, the algorithms are applied on real production data. Finally, we conclude on our findings and discuss the opportunities and restrictions of the algorithms as well as possibilites for further improvement.

2 Optimization Problem

Along the entire length P of a coated foil, I quality parameters are reported periodically by the production machines. For the i-th quality parameter, the quality $q_i(p)$ of the coated foil at position p, $0 \leq p \leq P$, is defined as the measurement of parameter i that is closest to p. For each parameter, a lower and an upper specification limit $\mathrm{LSL}_i, \mathrm{USL}_i$ are provided. This allows us to define the relative amount of values outside those specification limits, hereafter referred to as outliers, in parameter i for a sheet of length l that ends at position p as:

$$r_i(p) = \frac{1}{l} \int_{p-l}^{p} \mathbb{1}_{\{q_i(x) < \mathrm{LSL}_i \vee q_i(x) > \mathrm{USL}_i\}}(x) dx \ . \tag{1}$$

To take the inaccuracy of the measuring systems into account, a specific percentage of outliers $\alpha_i, 0 \leq \alpha_i \leq 1$, is allowed in the i–th quality parameter of each sheet. If any quality parameter exceeds this limit, the sheet is considered as not in order (n.i.o.), which is expressed by defining the cost $c_i(p)$ of parameter i for a sheet that ends at position p as:

$$c_i(p) = \begin{cases} \infty & , \exists i = 1, \ldots, I : r_i(p) > \alpha_i \\ r_i(p) & , \text{else} \end{cases} \ . \tag{2}$$

The goal of the optimization can be split into two parts: First, the maximum number S of valid sheets that can be cut out of the coated foil has to be determined. Second, the sum of costs in the first quality parameter when choosing S sheets has to be minimized. If several selections of sheets lead to the same sum of costs, the second to the I-th quality parameter have to be minimized in sequential order. This can be summarized as priority based optimization. During the optimization, the cut out sheets are not allowed to overlap. This is expressed by the constraint that if a sheet ending at p is cut out, no more sheets ending at positions between $p - l$ and $p + l$ may be picked.

3 Maximization of Sheet Count

To detect the maximum number of valid sheets in a given coated foil, we use a naïve algorithm. The key idea is to start at the beginning of a foil and set the possible ending p of the first sheet to the required length of the sheets l. Then check whether a sheet ending at that position is valid in all quality parameters. If this is the case, the count of valid sheets is increased by 1 and the ending position is increased by l. If the current sheet is invalid, the position is increased by the maximum difference of allowed and measured outliers over all parameters. This is done because a shorter increase would always result into another invalid sheet. For example if the number of outliers found is 15 with the maximum number of allowed outliers being 10 an increase of 4 would lead to at least 11 outliers in the next sheet. These steps are repeated until the end of the foil is reached. Pseudo code is given in Algorithm 1. Note that in some situations the stepsize can converge to zero, causing the algorithm to stop accidentally. Thus, a minimal increment is implemented in line 9.

Algorithm 1: Naïve Count Maximization.

Require: $r_1(\cdot), \ldots, r_I(\cdot), \alpha_1, \ldots, \alpha_I, P, l, I$

1: Define $sheetCount \leftarrow 0$
2: Define $pos \leftarrow l$
3: **while** $pos < P$ **do**
4: **if** $\forall i = 1, \ldots, I : r_i(p) \leq \alpha_i$ **then**
5: $sheetCount \leftarrow sheetCount + 1$
6: step $\leftarrow l$
7: **else**
8: step $\leftarrow \max_{i=1,\ldots,I} r_i(p) - \alpha_i$
9: step $\leftarrow \max(\text{step}, 1e-16) \cdot l$
10: **end if**
11: $pos \leftarrow pos + \text{step}$
12: **end while**
13: **return** $sheetCount$

4 Optimization of Sheet Quality

Having found the maximum number of valid sheets, the next task is to find the positions of the sheets that minimize the number of outliers. To make this problem solvable with tolerable effort, it is discretized in such a way that it is not allowed anymore to end a sheet at any position p, but only at a finite number of J predefined positions $p_j, j = 1, \ldots, J$, which are structured in a fine grid. From this perspective, each position p_j has the costs $c_i(p_j) \geq 0, i = 1, \ldots, I$, and the aim is to choose S out of the J positions that minimize the overall costs.

This is similar to the 0-1 multi-objective knapsack problem (MOKP) (Lust and Teghem, 2012) in which a subset of J objects with different utility values and a weight has to be put into a knapsack with a given weight limit so that the sum of the chosen utility values is maximized. In the past decades, several approaches to this problem have been published using approximations (Erlebach et al, 2001) or dominance relations to compute a set of possible solutions that each can not be further improved in at least one value parameter (Bazgan et al, 2009). While this can be used to choose a priority-optimum of all non-dominated solutions a posteriori, the runtime complexity for high dimensional value parameters (I) or big numbers of objects (J) makes this approach unsuitable for our given application. In section 4.2, these approaches will be used as benchmark for our proposed method.

As a result, it is beneficial to point out the two main differences between the MOKP and the sheet selection problem of this paper. In contrast to the MOKP, in the sheet optimization the weight of each p_j is 1, and a knapsack of capacity S has to be filled. This offers a big runtime opportunity against using a generalized MOKP solver. On the other hand, the sheet selection requires the further constraint that selecting a sheet that ends at a position p_{j^*} prohibits selecting further sheets that end at positions p_j with a distance of $\left|p_j - p_{j^*}\right| < l$ because of the sheets' lengths.

In Section 4.1, an algorithm is presented that is based on a dynamic programming approach for the knapsack problem and further implements a predecessor-structure to efficiently solve the sheet selection problem for large scale problems. In Section 4.2, a small simulation study is given to show the runtime efficiency of the algorithm.

4.1 Sheet Selection Algorithm

The constraint introduced above is taken into account by creating sets of allowed predecessors $\varphi_{j,s}$. Each of these sets gives the permitted predecessors of the sheet ending at position p_j when this sheet and $s-1$ other sheets have to be chosen. This a priori allows to prohibit predecessors of each position according to the length of the sheets. Moreover, these sets will be used when iterating over the quality parameters to mark which sheet paths are optimal in the quality parameters considered in previous iterations.

In analogy to the single-objective 0-1 knapsack problem (Martello and Toth, 1990, p. 38), the cost $d_{i,s}(p_j)$ in quality parameter i of picking the sheet that ends at position p_j as well as the $s-1$ best possible sheets out of the permitted preceeding sheets is defined recursively as:

$$d_{i,s}(p_j) = \begin{cases} c_i(p_j) + \min\limits_{a \in \varphi_{j,s}} (d_{i,s-1}(p_a)) & \text{if } s > 1 \wedge \varphi_{j,s} \neq \emptyset \\ \infty & \text{if } s > 1 \wedge \varphi_{j,s} = \emptyset \\ c_i(p_j) & \text{else} \end{cases} \tag{3}$$

This formula is computed in Algorithm 2 by using dynamic programming. Iterating over the number of sheets s and the current position p_j, the matrix ownCost[j, s] is filled with the corresponding costs of $d_{i,s}(p_j)$ (lines 6 to 14). At the same time, the sets of valid predecessors valPred[j, s] are updated so that for each sheet only the predecessors with minimal costs are kept as valid predecessors.

After each iteration of the outer loop, only paths of sheets that are optimal with regards to the corresponding quality parameter are left as input for the optimization of the next quality parameter (lines 15 to 23). This ensures the optimization in the given order of the quality parameters. After all quality parameters are considered, a selection of S sheets that is optimal in terms of the optimization goal is given out as optSelection (lines 24 to 30). Note that this section can be modified to return all possibile optimal selections, which might however entail a sharp increase of the running time.

Algorithm 2: Dynamic Sheet Optimization.

Require: valPred$[1, \ldots, J, 1, \ldots, S]$, $c_1(\cdot), \ldots, c_J(\cdot)$, I, J, S

/ Iterate over all quality parameters in their priority order */*
for all quality parameters $i = 1, \ldots, I$ **do**
  Define ownCost$[1, \ldots, J][1, \ldots, S] \leftarrow \infty$
  for all sheet counts $s = 1, \ldots, S$ **do**
    for all positions p_j with indices $j = 1, \ldots, J$ **do**
      if valPred$[j, s] \neq \emptyset$ **then**
        if $s = 1$ **then**
          ownCost$[j, s] \leftarrow c_i(p_j)$
        else
          / Pick the sheet ending at p_j and its cheapest predecessors */*
          minCost $\leftarrow \min_{a \in \text{valPred}[j,s]}$ ownCost$[a, s-1]$
          cheapestPreds $\leftarrow \arg\min_{a \in \text{valPred}[j,s]}$ ownCost$[a, s-1]$
          ownCost$[j, s] \leftarrow c_i(p_j)$ + minCost
          valPred$[j, s] \leftarrow$ cheapestPreds
        end if
      end if
    end for
  end for
  / Delete selections that are not optimal */*
  cheapestSelections $\leftarrow \arg\min_{j=1,\ldots,J}$ ownCost$[j, S]$
  for $j \notin$ cheapestSelections **do**
    valPred$[j, S] \leftarrow \emptyset$
  end for
  / Delete subselections that have no valid successor */*
  for sheet counts $s = S - 1, \ldots, 1$ **do**
    for all position indices $j = 1, \ldots, J$ **do**
      if $j \notin \bigcup$ valPred$[\cdot, s+1]$ **then**
        valPred$[j, s] \leftarrow \emptyset$
      end if
    end for
  end for
end for
/ Return one optimal selection of sheets */*
Declare optSelection$[1, \ldots, S]$
cheapestSelections $\leftarrow \arg\min_{j=1,\ldots,J}$ ownCost$[j, S]$
optSelection$[S] \leftarrow$ one arbitrary element of cheapestSelections
for sheet counts $s = S, \ldots, 2$ **do**
  optSelection$[s-1] \leftarrow$ one arbitrary element of valPred[optSelection$[s]$, s]
end for
return optSelection

A numerical example of the algorithm is given in Figure 1. Here, the task is to choose 4 out of 9 sheets with 2 quality parameters and predefined costs $c_i(p_j)$. Due to the sheets' length there is the constraint that picking one sheet prohibits to pick its direct predecessor. The two tables show the ownCost matrix in each iteration. The lines and circles show which predecessors are still valid after the corresponding iteration. After the first iteration, there are four selections with optimal costs in the first parameter, those are the sheets ending at positions p_j with $j \in \{1, 4, 6, 8\}$, $j \in \{2, 4, 6, 8\}$, $j \in \{1, 5, 7, 9\}$ or $j \in \{2, 5, 7, 9\}$. Out of these possible selections, the sheets ending at positions p_j, $j \in \{2, 5, 7, 9\}$, optimize the costs of the second parameter and are therefore selected in the last iteration. Their total cost is 9 in the first and 10 in the second parameter.

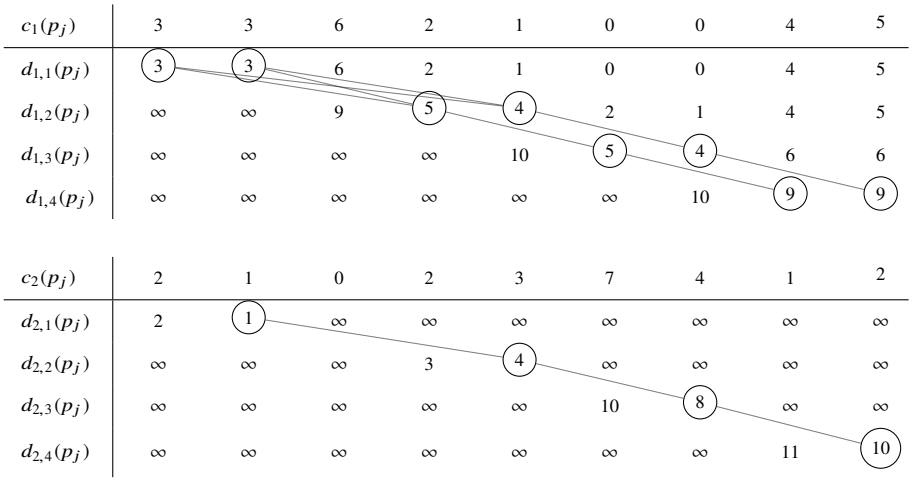

Figure 1: Example of Dynamic Sheet Optimization for $I = 2$ quality parameters to be optimized, $J = 9$ possible sheets and $S = 4$ sheets to be selected. Optimal selections after each iteration are highlighted. The top graphic represents the first quality parameter with several different combinations of 4 sheets leading to a total cost of 9. The second graphic shows the second quality prameter in the same manner with the possible choices of selectable sheets being limited by the results in the first parameter .

4.2 Runtime Study

To evaluate the running time of the optimization algorithm, it is executed in three settings, each with varying sizes of I, J and S:

1. **Random costs:** The costs are drawn independently from a binomial distribution $c_i(p_j) \sim \text{Bin}(n = 20, p = 0.5)$, $j = 1, \ldots, J$, $i = 1, \ldots, I$. The length of each sheet is set to 0, which means there are no constraints in combining the sheets with each other. This does not correspond to the industrial application but delivers runtimes that can be compared to solvers of the MOKP.

2. **Autocorrelated costs:** Here, within each quality parameter i, the costs are autocorrelated. This is achieved by drawing independent binomially distributed variables $X_{i,j} \sim \text{Bin}(n = 20, p = 0.5)$ and defining the cost as the cumulative sum $c_i(p_j) = \sum_{a=1}^{J}(X_{i,a} - 10)$. Moreover, the sheets' length is taken into account by the constraint that each position p_j must not be selected together with the $5\% \cdot J$ positions laying before it. This promises to be an application-oriented setting where the utility values are natural numbers.

3. **Realistic costs:** Measurement series in real world applications can often be well decomposed into different frequencies using, e.g., a Fourier Transform. Hence, this strategy can be applied in reverse order to simulate realistic measurement series. In this setting, the foil's quality itself is simulated by superimposing ten shifted and amplified sinus waves for each quality parameter and adding a random measurement inaccuracy:

$$q_i(p_j) = \sum_{k=1}^{10} \left(\beta_{1,k} + \beta_{2,k} \cdot \sin\left(p_j \cdot \frac{2\pi}{\beta_{4,k}} + \beta_{3,k} \right) \right) + e_j \ ,$$

with $\beta_{1,k} \sim U[-0.25, 0.25]$, $\beta_{2,k} \sim U[0, 0.3]$, $\beta_{3,k} \sim U[-\pi, \pi]$, $\beta_{4,k} \sim U[5, 50]$, $e_j \sim N(0, 0.01)$ drawn randomly. The costs are then given by the relative amount of outliers $r_i(p_j)$ as defined in Chapter 2 with $\alpha_i = 0.2$, where $LSL_i = -1$ and $USL_i = 1, \forall i = 1, \ldots, I$. The task is to extract sheets of length $l = 5$ from a foil of length $P = 100$. The number of sheets S depends on the simulated foils and is therefore computed by Algorithm 1. Trivial cases in which $S = 0$ are rejected and re-simulated.

In the first two cases, a binomial distribution is used for generating the quality parameters, because in any continuous distribution the optimization of the first parameter would almost surely give a single optimal selection of sheets, making the optimization of the remaining quality parameters redundant. In the third case, continuous values can be used as they are truncated at 0 and 0.2. The simulations[1] run in `R` (R Core Team, 2018) in the version 3.5.0. All computations are executed on a single core of an Intel Xeon E5-2630 CPU at 2.30 GHz. Each simulation is repeated 100 times with different randomized costs. Note that the generation of the cost functions as well as the declaration of the inital valid predecessor structure and – if necessary – the determination of S are not part of the algorithm and, therefore, not included in the reported running times.

The results of the three settings are given in Tables 1, 2 and 3 in corresponding order. Note that the worst-case running time of Algorithm 2 is dominated by iterating over I parameters, S sheet counts and J positions and computing a minimum over at most J other positions, yielding a runtime of $O(I \cdot S \cdot J^2)$. The experimental running times in Table 1 provide evidence for this. It can further be seen that the fastest and slowest running times are close to the median in nearly all cases. As mentioned above, these results can be compared to general MOKP solver. Bazgan et al (2009) and Delort and Spanjaard (2010) have run experiments with a similar random sampling (called Type A) with the goal of extracting the exact pareto front, which can be transformed into the priority-optimum desired in our case. When comparing the runtime, our approach performs similar on rather small instances ($J = 100$ and $I = 2$ to 3), but outperforms the MOKP solvers once J or I start growing. This allows to solve problems with a number of positions (J) ten times bigger than in the previous approaches, and to increase the number of considered costs (I) well beyond the previous limit of 3. This shows that our approach utilizes the special constraints in our problem well. In the second setting, there are less a priori valid predecessors due to the length constraint. As can be seen from Table 2, this, however, does not yield faster but comparable running times, except for some minimum running times. In Table 3 the number of sheets S varies depending on the randomly simulated quality functions. This explains the bigger variation in running times. It is notable that the running times stay similar even for a higher

[1] Available under `https://github.com/mkirchhof/optPiecesRuntimeStudy`

number of quality parameters (I). This may be explained by the fact that more quality parameters lead to less regions where all parameters stay inside their specification limits, therefore reducing the number of possible sheets S and thus the running times. In fact, across $I = 3, 5$ and 10 the median S was 14, 10.5 and 5, respectively. In conclusion, the algorithm scales linearly with the number of quality parameters I and the number of sheets to be selected S and quadratically with the number of available positions J.

Table 1: Running time of the Dynamic Sheet Optimization in seconds with random costs. The number of positions J, the number of quality parameters I and the number of sheets to be selected S are varied.

		$I = 3$			$I = 5$			$I = 10$		
	J	Min	Median	Max	Min	Median	Max	Min	Median	Max
$S = 10$	100	0.051	0.059	0.074	0.083	0.085	0.086	0.128	0.151	0.156
	500	0.810	0.827	0.993	1.186	1.202	1.343	2.180	2.210	2.359
	1000	1.782	2.996	3.066	4.452	4.527	4.686	7.643	8.245	9.437
	5000	71.535	71.906	75.431	108.562	109.716	120.060	116.566	206.638	210.431
	10000	281.555	292.278	307.464	436.649	439.539	443.903	797.267	822.193	884.786
$S = 20$	100	0.102	0.123	0.128	0.161	0.178	0.195	0.290	0.327	0.460
	500	1.629	1.660	2.000	2.461	2.487	2.622	4.396	4.431	4.733
	1000	5.964	6.606	7.128	9.376	9.616	10.407	17.056	17.502	19.174
	5000	148.384	150.138	158.015	226.139	229.181	241.768	422.145	428.372	448.539
	10000	601.173	612.102	636.405	512.515	947.360	1001.984	934.023	1669.723	1705.086
$S = 50$	100	0.288	0.329	0.406	0.425	0.470	0.588	0.794	0.825	0.983
	500	4.358	4.521	4.875	6.906	7.027	7.859	11.221	11.421	12.962
	1000	16.124	16.314	17.215	23.661	24.319	25.808	43.556	45.226	48.355
	5000	385.053	394.631	445.054	589.267	596.445	622.367	1056.596	1109.547	1197.112
	10000	868.156	896.929	1636.727	1303.283	1350.215	2555.631	2387.141	2508.293	4503.934

Table 2: Running time of the Dynamic Sheet Optimization in seconds with autocorrelated costs. The number of positions J and the number of quality parameters I are varied, the number of sheets to be selected is fixed at $S = 10$.

	$I = 3$			$I = 5$			$I = 10$		
J	Min	Median	Max	Min	Median	Max	Min	Median	Max
100	0.049	0.058	0.080	0.048	0.089	0.096	0.088	0.141	0.191
500	0.784	0.880	0.976	1.194	1.253	1.428	2.264	2.302	2.483
1000	2.922	3.058	3.217	4.521	4.700	5.424	4.637	8.733	9.650
5000	52.947	71.417	75.561	107.680	108.678	112.456	205.328	208.497	217.378
10000	161.339	287.524	296.573	434.030	456.102	476.220	464.091	845.039	896.169

Table 3: Running time of the Dynamic Sheet Optimization in seconds with realistic costs. The number of positions J and the number of quality parameters I are varied, the number of sheets to be selected S is variable and depends on the simulated quality functions. For $I = 3$ quality parameters, the median of S is 14, for $I = 5$ it is 10.5 and for $I = 10$ it is 5.

	$I = 3$			$I = 5$			$I = 10$		
J	Min	Median	Max	Min	Median	Max	Min	Median	Max
100	0.003	0.101	0.201	0.004	0.126	0.332	0.008	0.061	0.392
500	0.273	1.498	2.764	0.048	2.210	5.614	0.097	1.884	7.603
1000	0.854	6.256	9.190	0.161	7.778	13.188	0.281	5.628	19.225
5000	2.041	136.111	193.631	17.013	141.708	314.461	6.102	130.901	477.014
10000	4.801	292.199	464.287	14.104	390.217	1224.821	16.312	364.728	1275.668

5 Application

After showing that our tool works for simulated data, we present a use case inspired by experiments from the earlier described production of electrode sheets located at the Zentrum für Sonnenenergie- und Wasserstoff-Forschung Baden-Württemberg (ZSW).

The production process of these sheets consists of three sequential production steps. First, an active material in form of a slurry is applied to the foil. This slurry is the center component of the electrode sheets since in the finished battery lithium-ions move between different layers of slurry producing electric current. In the coating step, the applied mass of the slurry per cm^2 defines the quality of the coated foil.

Next, the foil is sent through a calander to compress the applied slurry. Afterwards, the thickness of the coated foil is measured because it is highly correlated with the density and porosity of the material. Since the density and porosity determine the ability of the slurry to carry electronic charge but can only be determined using destructive testing methods, the thickness is commonly used as a surrogate inline quality parameter.

In order to reduce the production time, the foil is produced at twice the width of the sheets used to construct the final batteries. Thus, in the final step the foil has to be cut in half lengthwise. In the batteries themselves, a coil of three different types of foils is used: Anode, cathode and a kind of seperator to ensure those two do not contact each other directly. Since these foils need to overlap each other in a predefined manner in order to prevent short circuits, the width of each of the foils is another important quality parameter.

The following table shows the quality parameters of each of the three production steps, including their corresponding specification limits and the relative number of permitted outliers α.

Table 4: Quality parameters of electrode sheet production.

Production step	Parameter	Unit	Target	LSL	USL	α
Coating	Dry weight	g/m^2	182	180.16	183.84	0.05
Calandering	Thickness	μm	140	136.00	144.00	0.05
Slitting	Width	mm	164	163.50	164.50	0.05

The first three plots in Figure 2 show what the sheet selection results for each production step would look like if all three quality parameters were to be optimized individually. This kind of analysis can be used to determine the productivity of each of the production steps and can help to find the source of waste in the final product. The results of the sheet selection algorithms after all three process steps are shown in the bottom plot in Figure 2. From the 200 m long foil, 25 sheets of length 4.4 m are selected. Altogether there is 110 m (55%) of i.o. and 90 m (45%) of n.i.o. material. The unsatisfying results for the first part of the coil can be explained by the fact that especially the coating process needs some time at the beginning of every production phase to reach the targeted production level. This part of the production process is usually left out of most analyses, however, it was included here to better demonstrate the functionality of the algorithm.

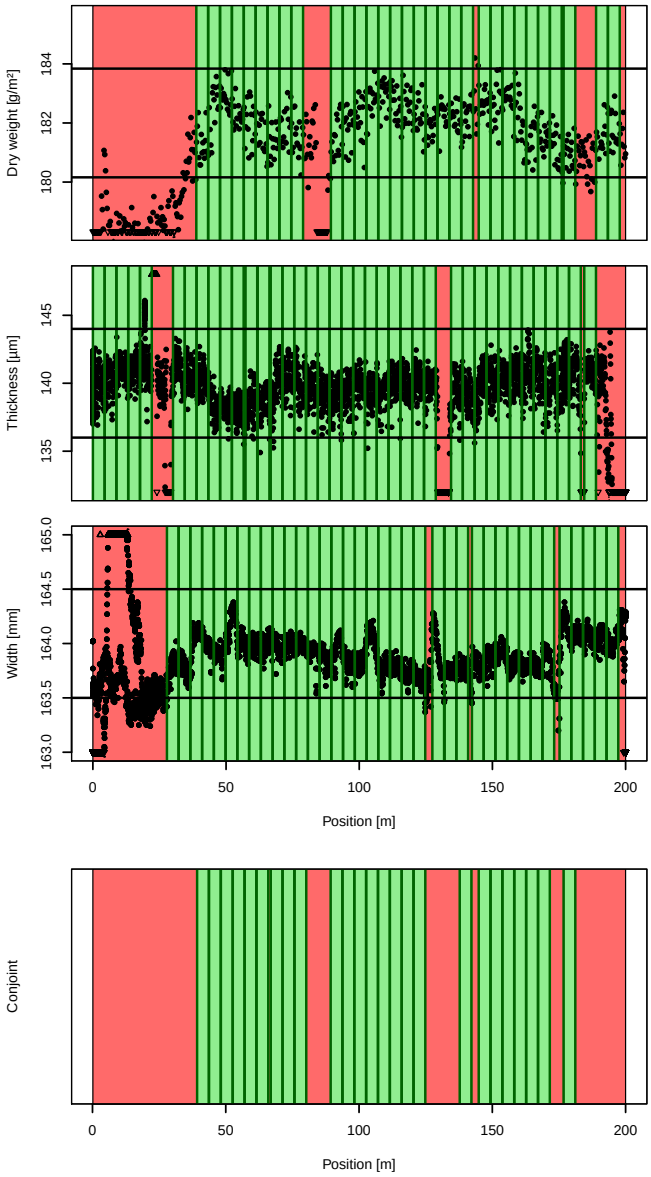

Figure 2: Upper three plots show individual sheet selection results. Bottom plot shows conjoint sheet selection result. Upper and lower specification limits are highlighted as horizontal lines. Red background indicates n.i.o. material, green background indicates the selected sheets. The plots are limited in their y-axis and truncated points are shown as triangles.

6 Conclusion

We developed a combination of two algorithms in order to detect the optimal electrode sheets to be cut from a long production foil regarding multiple quality parameters. The first algorithm uses a naïve approach to determine the maximum number of possible sheets. The second algorithm builds upon the 0-1 multi-objective knapsack problem and is customized to respect length constraints. We showed that the latter algorithm is capable of cutting 50 sheets out of 10000 possible cutting points while considering 10 quality parameters of ordered importance within a reasonable time. Moreover, we applied the algorithm on data from the production of rechargeable Lithium-Ion battery cells. This use case consisted of an analysis of $200m$ of coated foil and its overall quality was subject to three quality parameters.

This approach is applicable in a variety of industrial use cases. The algorithm also allows to include more complex length constraints such as cutting losses. One downside is that the result is only optimal in terms of the discretized cutting positions. This might also lead to situations in which the first algorithm that is able to cut sheets on any position finds more valid sheets than the second. In this case, the density of allowed cutting points has to be increased. Also, the cost functions used in this paper only consider whether measurements are inside the given specification limits. This might be extended to more sophisticated cost functions. The second algorithm is, just like the knapsack problem, independent of the choice of cost functions as long as the sum of costs over all sheets has to be minimized. While a priori ordering the quality parameters by their priority is a simplification compared to computing all non-dominated solutions and selecting a good trade off a posteriori, it is necessary to give this order in an automated production setting. Alternatively, the multiple cost functions could be merged into a one-dimensional quality index. However, the method based on the original quality parameters as was described here has so far been prefered by our industry partners.

In order to introduce these algorithms for industrial usage, we are developing a tool with a graphical user interface for convenient usage.[2] To enable the tool for big datasets, heuristics for increasing the speed of both algorithms can be included depending on the specific optimization problem. Moreover, we strive to generalize the priority optimization algorithm towards constrained multi-objective knapsack problems with weights.

[2] Available under `https://github.com/mkirchhof/optPieces`

Acknowledgements Financial support from the German Federal Ministry of Education and Research (BMBF) under the grant 03XP0076A (Projectcluster: ProZell, Project: QS-Zell) is gratefully acknowledged. We would like to thank the reviewers for their valuable comments and effort.

References

Bazgan C, Hugot H, Vanderpooten D (2009) Solving Efficiently the 0-1 Multi-Objective Knapsack Problem. Computers & Operations Research 36(1):260–279. DOI: 10.1016/j.cor.2007.09.009.

Delort C, Spanjaard O (2010) Using Bound Sets in Multiobjective Optimization: Application to the Biobjective Binary Knapsack Problem. In: Experimental Algorithms, Festa P (ed).

Erlebach T, Kellerer H, Pferschy U (2001) Approximating Multi-objective Knapsack Problems. In: Algorithms and Data Structures, Dehne F, Sack J, Tamassia R (eds), Springer, Berlin, Heidelberg (Germany), pp. 210–221. DOI: 10.1007/3-540-44634-6_20.

Lust T, Teghem J (2012) The Multiobjective Multidimensional Knapsack Problem: A Survey and a New Approach. International Transactions in Operational Research 19(4):495–520. DOI: 10.1111/j.1475-3995.2011.00840.x

Martello S, Toth P (1990) Knapsack Problems: Algorithms and Computer Implementations. John Wiley & Sons, New Jersey (USA).

R Core Team (2018) R: A Language and Environment for Statistical Computing. Vienna, Austria. R Foundation for Statistical Computing, Vienna, Austria. URL: `https://www.R-project.org/`.