Open access • Journal Article • DOI:10.1287/MNSC.44.5.698

Discrete Lotsizing and Scheduling by Batch Sequencing

Carsten Jordan, Andreas Drexl

Institutions: University of Kiel

Published on: 01 May 1998 - Management Science (INFORMS)

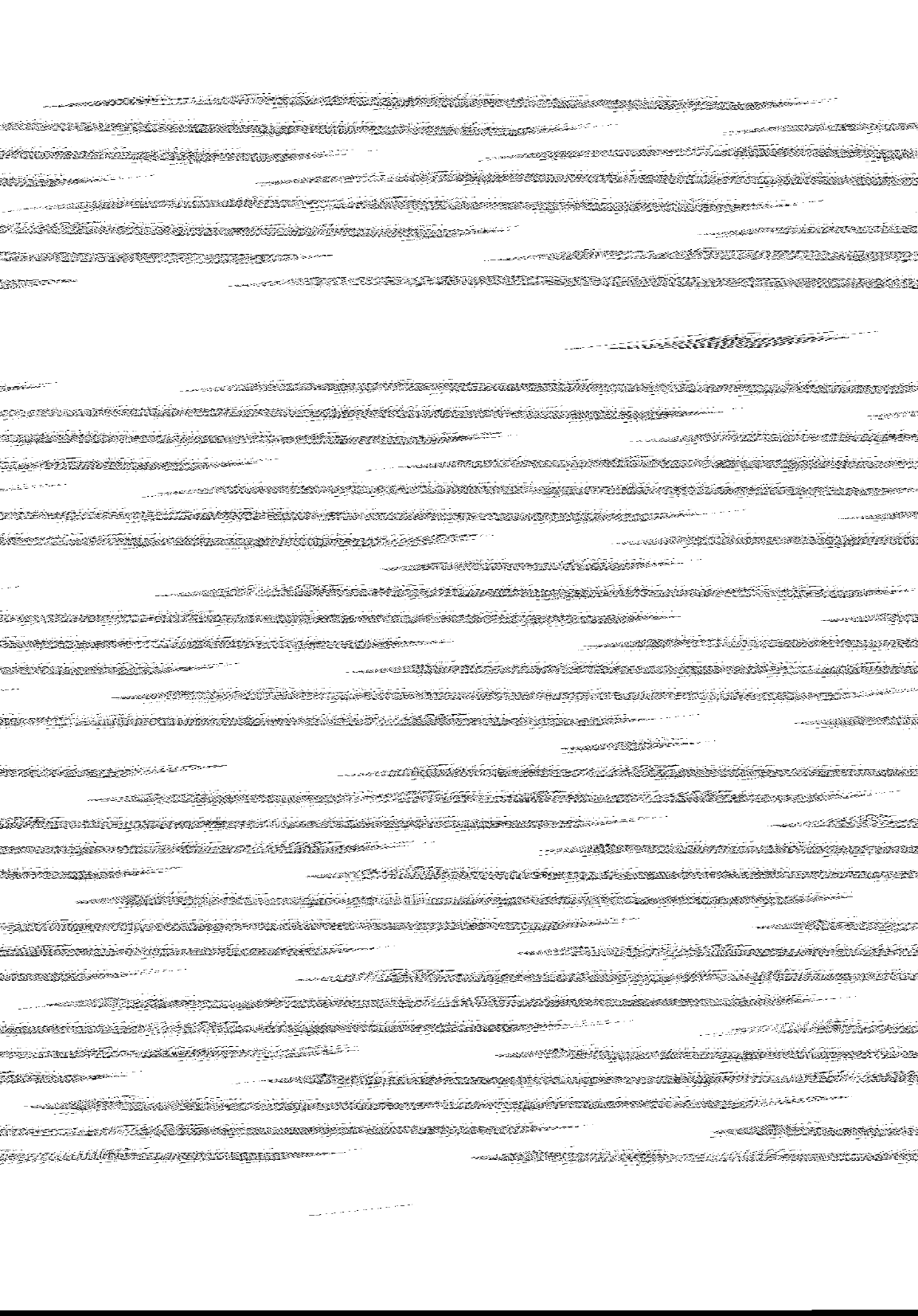
Topics: Single-machine scheduling, Job shop scheduling, Batch processing, Branch and bound and Scheduling (production processes)

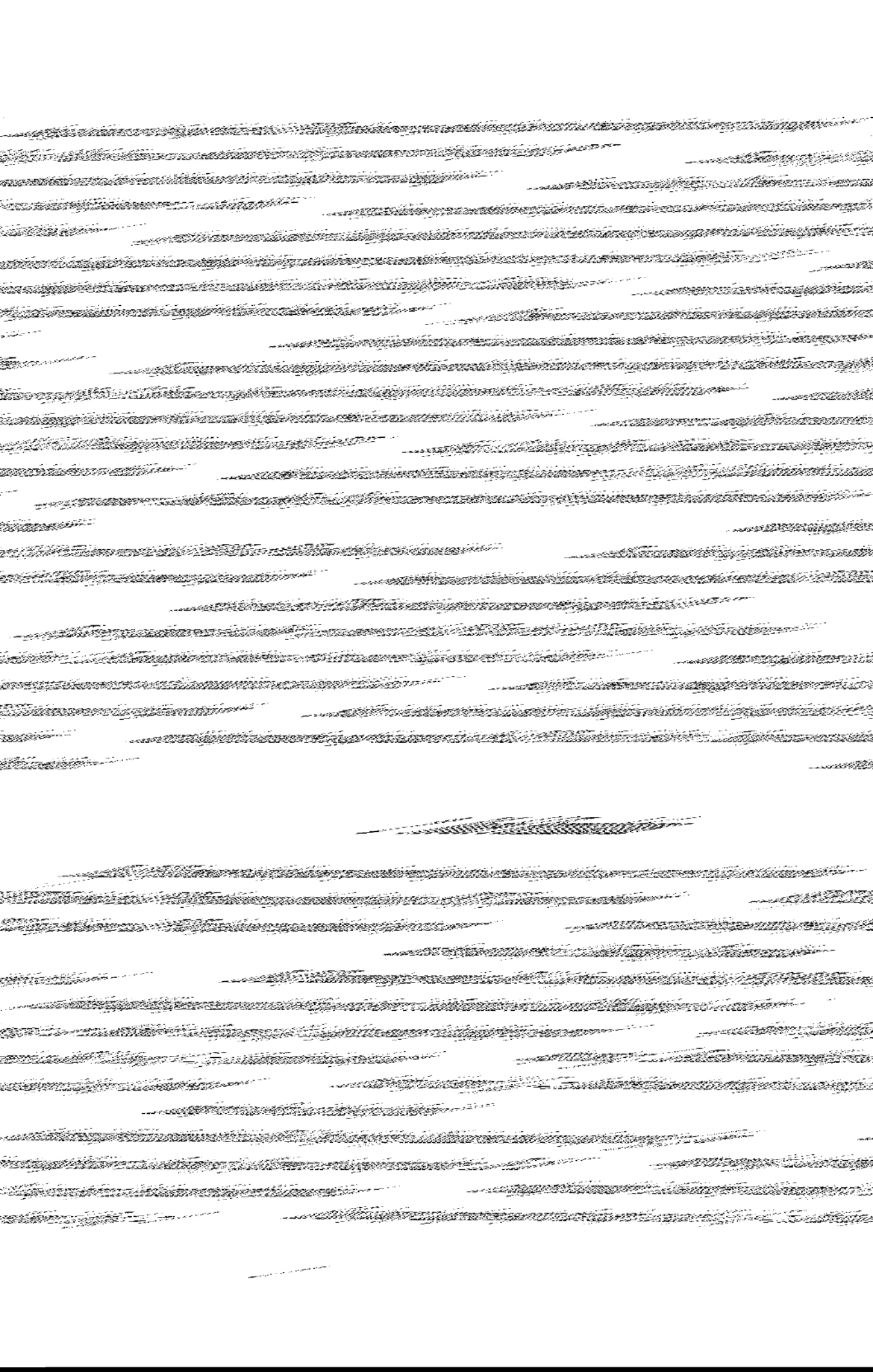
Jordan, Carsten; Drexl, Andreas

Discrete lotsizing and scheduling by batch sequencing

Manuskripte aus den Instituten für Betriebswirtschaftslehre der Universität Kiel, No. 438

Provided in Cooperation with:
Christian-Albrechts-University of Kiel, Institute of Business Administration





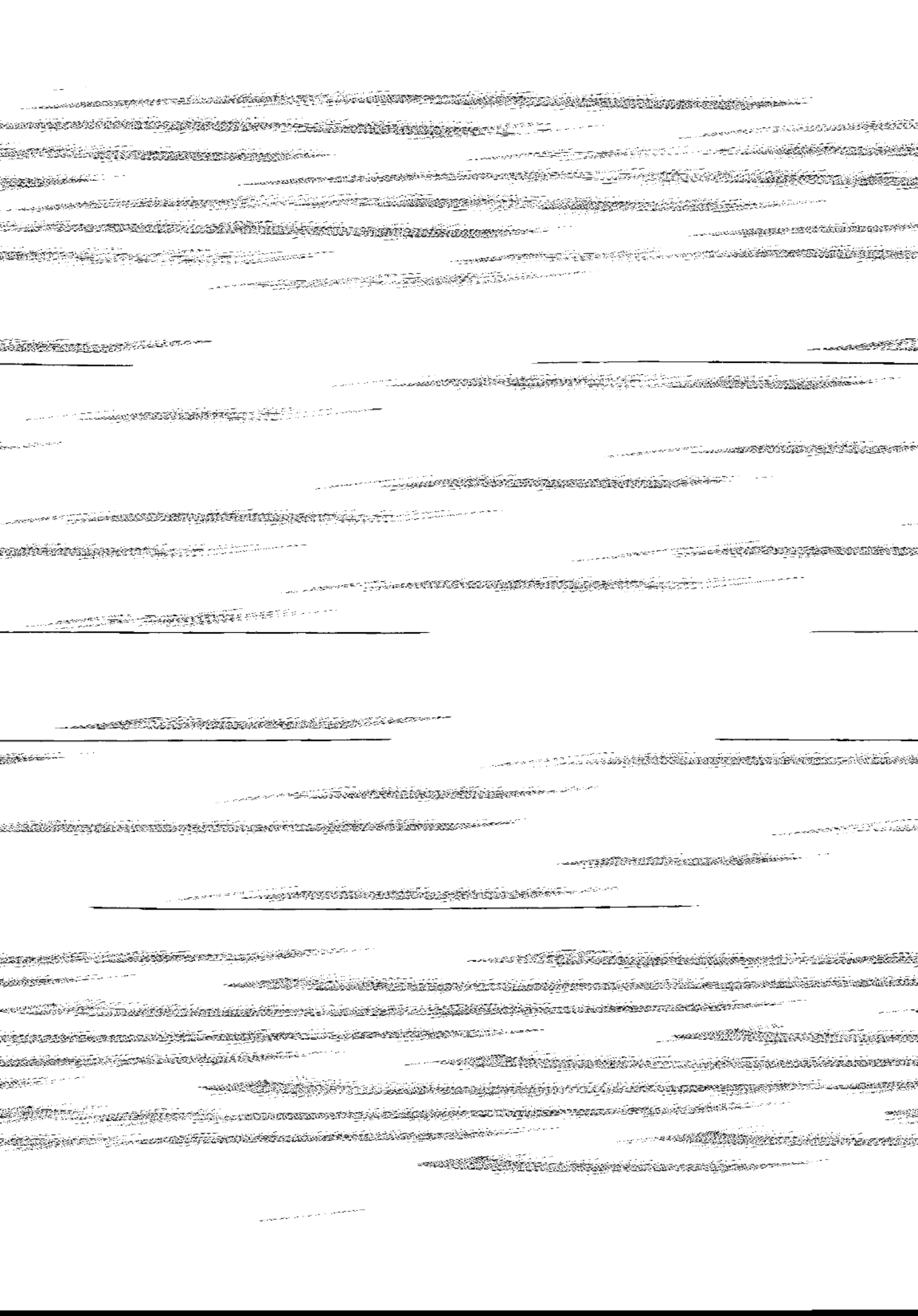
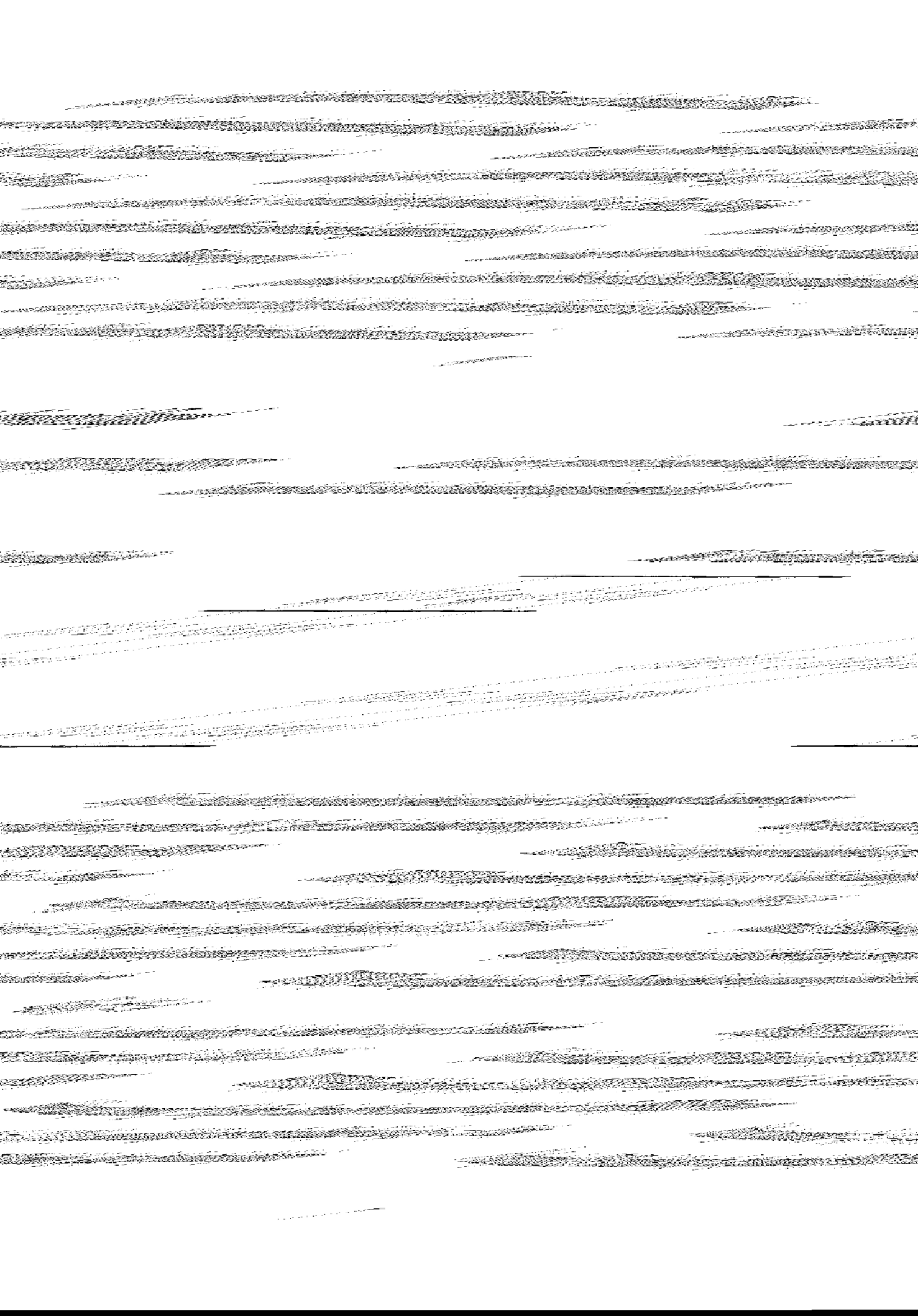


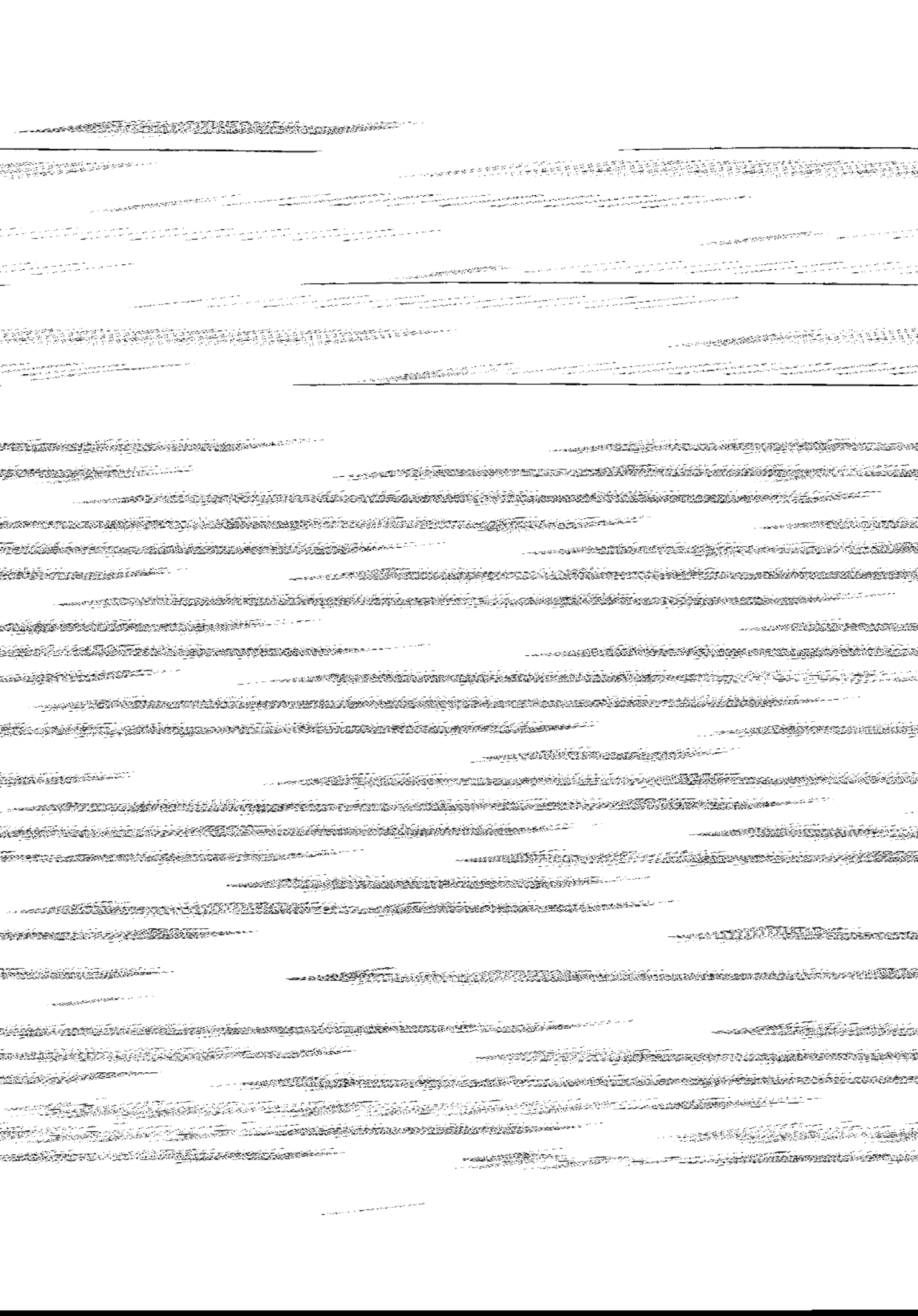
Table 3: Model for the DSCP

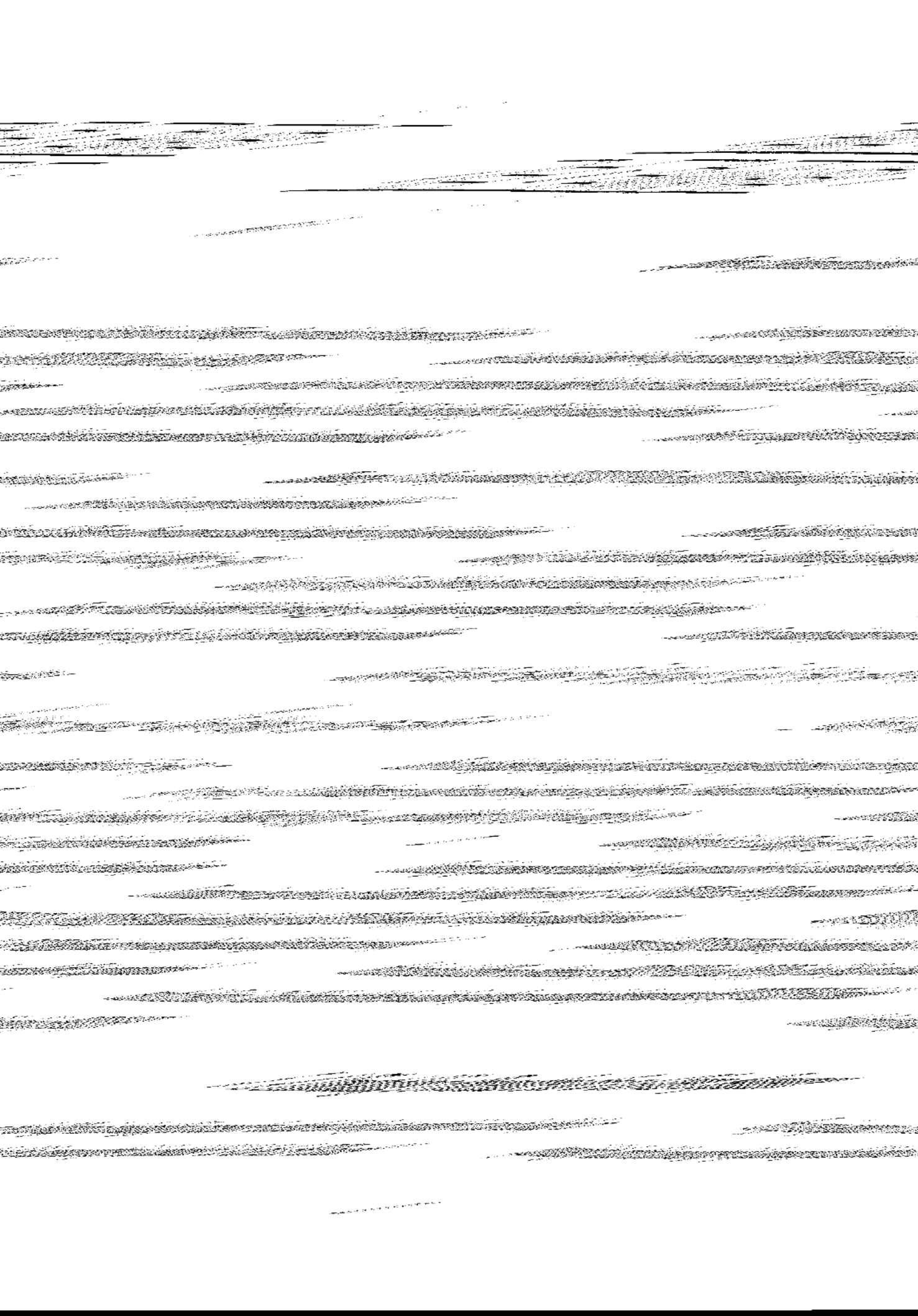
[illegible]

The decision variables are given in Table 3. The sequence π denotes the processing order of jobs [illegible] denote the job at position i together with completion times [illegible] [illegible] schedule π. A conceptual model formulation for the DSCP is presented in Table 3. [illegible] denotes the sum of tardiness and setup costs for a schedule π, which is minimized by the objective (1). The tardiness [illegible] is weighted by [illegible] and setup costs [illegible] are incurred between jobs of different [illegible] families. Each job is to be scheduled between its release and its deadline, while respecting the sequence [illegible] as well as the setup times. This is done by constraints (2)–(4). Constraints (5) and (6) equal [illegible] on the machine [illegible] between two consecutive jobs. We also have a setup time [illegible] from the idle [illegible] to one of them [illegible] dependent setup time [illegible]. Initialization at beginning and end is [illegible] machine rather than a sequence [illegible] respectively [illegible] the schedule are given in (7) and (8) [illegible]

Remark 1 For the ESP and DSCP parameters we assume that [illegible] setup times and setup costs satisfy the triangle inequality [illegible] [illegible]







7 Computational Results

From the analysis in Section 4 we know that we address the same planning problem in BSP and DLSP, and that we find corresponding solutions. Consequently, in this section we compare the performance of algorithms solving the BSP with procedures for solving variants of the DLSP. The comparison is made on the DLSP instances used to test the DLSP procedures; we take the instances provided by the cited authors and solve them as BSP(DLSP) or BSPUT(DLSP) instances (cf. Figure 1). An exception is made for reference [8] where we use randomly generated instances.

The different DLSP variants are summarized in Table 11. For the DLSP, in the first column the reference, in the second the DLSP variant is displayed. The fourth column denotes the proposed algorithm, the third column shows whether computational results for the proposed algorithm are reported for equal or unequal holding costs. Depending on the holding costs, the different DLSP variants are solved as BSP(DLSP) or BSPUT(DLSP) instances. With the exception of reference [18], the DLSP procedures are tested with equal holding costs, so that regenerative schedules are optimal in [4] and [8].

Table 11: *Solving Different DLSP Variants as a BSP*

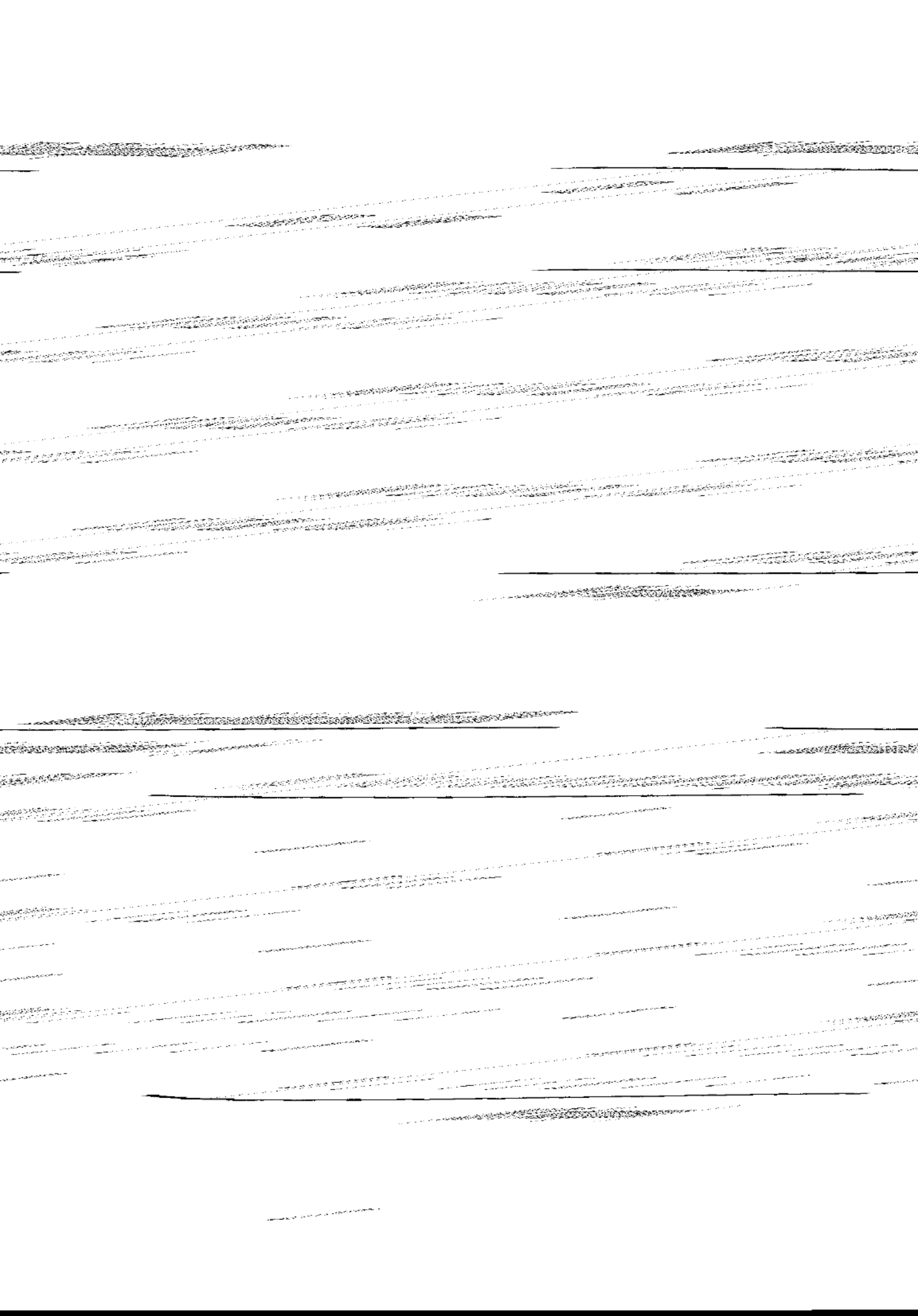
DLSP				BSP	
Author	Variant	Holding Costs	Algorithm	Instances	Properties of Schedules
Cattrysse et al. [4]	SISTSC	$h_i = 1$	DACGP	BSP(DLSP)	EDDWF and regenerative
Fleischmann [8]	SDSC	$h_i = 1$	TSPOROPT	BSP(DLSP)	EDDWF and regenerative
Salomon et al. [18]	SDSTSC	$h_i > 0$	TSPTWA	BSPUT(DLSP)	EDDWF
		$h_i = 0$	TSPTWA	BSP(DLSP)	EDDWF and one block

uence Independent Setup Times and Setup Costs (SISTSC)

et al. [4], a mathematical programming based procedure to solve SISTSC is proposed. al. [4] refer to their procedure as dual ascent and column generation procedure (DACGP). first formulated as a set partitioning problem (SPP) where the columns represent the edule for one item i; the costs of each column can be calculated separately because setups dependent. DACGP then computes a *lower bound* for the SPP by column generation, new generated solving a single item subproblem by a (polynomial) DP recursion. In DACGP le, i.e. an *upper bound*, may be found in the column generation step, or is calculated e algorithm with the columns generated so far. If in neither case a feasible schedule is is made with a simplex based procedure.

7.1 Seq

In Cattrysse
Cattrysse et a
The DLSP is
production sche
are sequence inc
columns can be
a feasible schedu
by an enumerative
found, an attempt



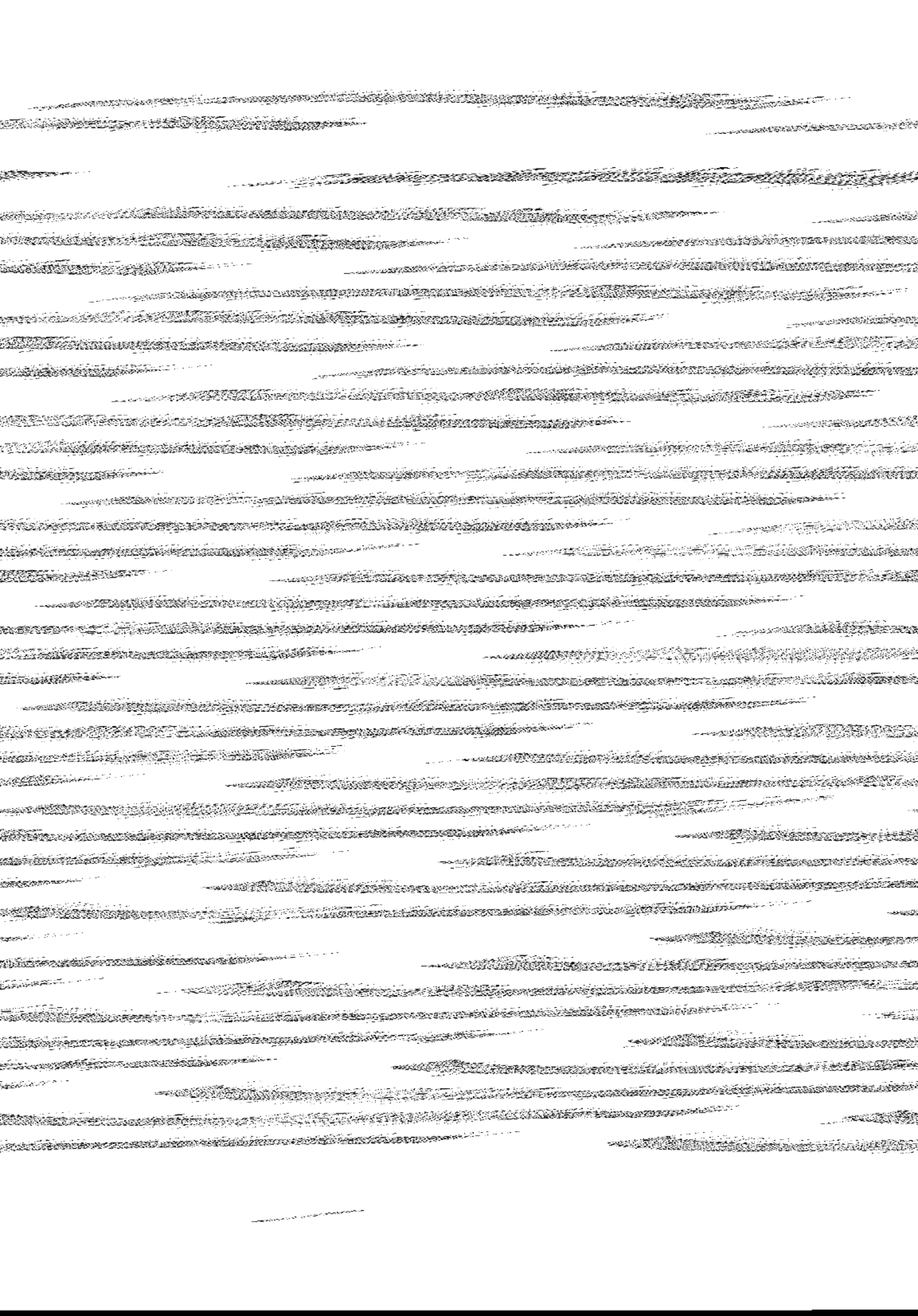


Table 15: Comparison of DLSP and BSP Algorithms for SDSTSC

			TSPTWA		SABSP			
(N,T)	$\#J$	b_j	$\#P$	R_{avg}	$\#P$	$\bar{R}_{avg}$	$\bar{R}^B_{avg}$	ΔZ^B_{max}
(10,48)	25	$=0$	21	<1200	30	3.5	–	–
	25	>0	1	<1200	21	(489.8)	(373.1)	0.0
(3,64)	38	$=0$	30	<1200	30	0.8	–	–
	38	>0	30	<1200	30	5.1	1.4	0.4
(5,64)	38	$=0$	25	<1200	30	28.0	–	–
	38	>0	0		30	140.2	36.3	0.0
(10,68)	38	$=0$	4	<1200	14	(41.6)	–	–
	38	>0	0		2	(408.0)	(498.4)	0.0

(486DX2/66 PC)

8 Summary and Conclusions

In this paper, we examined both the discrete lotsizing and scheduling problem (DLSP), and the batch sequencing problem (BSP).

We presented model formulations for the DLSP and for the BSP. In the DLSP, decisions regarding what is to be done are made in each individual period, while in the BSP, we decide how to schedule jobs.

The DLSP can be solved as a BSP if the DLSP instances are transformed. For each schedule of one model there is a corresponding solution for the other model. We proved the equivalence of both models, meaning that for an optimal schedule of the BSP the corresponding solution of the DLSP is also an optimal schedule, and vice versa.

In order to solve the BSP effectively, we tried to restrict the search to only a subset of all [illegible] possible. schedules. We found out that jobs of one family can be preordered according to their deadlines. Furthermore, for equal holding costs, it is optimal to start production for a family only if there is no inventory of this family.

When solving the BSP with a branch & bound algorithm to optimality, we face the difficulty that already the feasibility problem is difficult. We must maintain feasibility and minimize costs at the same time. Compared with other scheduling models, the objective function is rather difficult for the BSP. A tight lower bound could thus not be developed. We therefore used dominance rules to prune the search tree. Again, the difficult objective function complicates the dominance rules and forces us to distinguish different cases.

In order to evaluate our approach, we tested it against (specialized) procedures for solving variants of the DLSP. Despite the fact that we have no effective lower bound, our approach proved to be more efficient if (i) the number of items is small, and (ii) instances are hard to solve, i.e., capacity utilization is