Variable Neighbourhood Search Solving Sub-problems of a Lagrangian Flexible Scheduling Problem

Alexander Hämmerle[1] and Georg Weichhart[1,2]
[1]*Profactor GmbH, 4407 Steyr-Gleink, Austria*
[2]*Dep. of Business Informatics - Communications Engineering, Johannes Kepler University, 4040 Linz, Austria*

Keywords: Flexible Job Shop Scheduling, Lagrangian Relaxation, Subgradient Search, Variable Neighbourhood Search.

Abstract: New technologies allow the production of goods to be geographically distributed across multiple job shops. When optimising schedules of production jobs in such networks, transportation times between job shops and machines can not be neglected but must be taken into account. We have researched a mathematical formulation and implementation for flexible job shop scheduling problems, minimising total weighted tardiness, and considering transportation times between machines. Based on a time-indexed problem formulation, we apply Lagrangian relaxation, and the scheduling problem is decomposed into independent job-level sub-problems. This results in multiple single job problems to be solved. For this problem, we describe a variable neighbourhood search algorithm, efficiently solving a single flexible job (sub-)problem with many timeslots. The Lagrangian dual problem is solved with a surrogate subgradient search method aggregating the partial solutions. The performance of surrogate subgradient search with VNS is compared with a combination of dynamic programming solving sub-problems, and a standard subgradient search for the overall problem. The algorithms are benchmarked with published problem instances for flexible job shop scheduling. Based on these instances we present novel problem instances for flexible job shop scheduling with transportation times between machines, and lower and upper bounds on total weighted tardiness are calculated for these instances.

1 INTRODUCTION

Manufacturing processes are executed across geographically distributed job shops. Despite this distribution, a globally optimised production schedule is desirable to stay competitive. For implementation of an optimisation algorithm for distributed production networks, we have formulated a flexible job shop problem with transport times to account for the distribution. The jobs have to be scheduled across multiple machines in the production network. Each job is consisting of a defined sequence of operations. The machines assigned to a specific job may be in different shops, requiring transport of the workpiece(s). We regard flexible scheduling problems, i.e. an operation can be processed on alternative machines, where each operation takes a different timespan to be completed. We do not consider transport resources with limited capacities, hence we are not confronted with the additional problem of routing transport resources. Due dates for jobs are specified, and we consider total weighted tardiness as overall objective function.

As the distribution of the production system is one of the distinguishing features of our approach, *distribution* as a general principle, guided the development of our algorithm solving the outlined problem. The scheduling problem has been divided into multiple sub-problems that can be solved in a distributed manner. In a physical modelling approach, a sub-problem corresponds to a physical entity, e.g. a machine, a job or a job shop. With a proper decomposition into sub-problems, a distributed scheduling algorithm allows dynamic scheduling with localised disturbance handling, confined to a small set of sub-problems, and performance improvements through concurrent computing.

We modelled the problem decomposition based on a sound mathematical foundation, and the distributed scheduling algorithm provides not only high-quality upper bounds on total weighted tardiness, but also guaranteed lower bounds.

Lagrangian relaxation is a method satisfying these requirements, with a substantial scientific track record for job shop scheduling problems. Relevant work can be found in (Hoitomt et al., 1993), (Wang et al., 1997), (Chen and Luh, 2003), (Baptiste et al., 2008),

HÃd'mmerle A. and Weichhart G.
Variable Neighbourhood Search Solving Sub-problems of a Lagrangian Flexible Scheduling Problem.
DOI: 10.5220/0006114102340241
In *Proceedings of the 6th International Conference on Operations Research and Enterprise Systems (ICORES 2017)*, pages 234-241
ISBN: 978-989-758-218-9

(Buil et al., 2012), (Chen et al., 1998) and (Kaskavelis and Caramanis, 1998). An analysis of the relevant work shows that predominantly *machine capacity constraints* are relaxed, based on a time-indexed mathematical formulation. The resulting job-level sub-problems (single job-shop scheduling problems) are not $\mathcal{NP}$-hard and they are solved with dynamic programming, with its complexity depending on the required number of timeslots for the scheduling problem instance. The dual problem. through which the sub-problem solutions are brought together, is solved with variants of subgradient search. The effect that the single job-shop scheduling problems are solved concurrently is, that the generated solution is (likely) infeasible, with multiple job-operations scheduled on a single machine. The favourite method for feasibility repair is list scheduling.

Our work relies on the mainstream results of the relevant work. In our approach we relax machine capacity constraints, we use subgradient search to solve the dual problem and we use list scheduling for feasibility repair.

However, when it comes to solving sub-problems, dynamic programming is not efficient enough for problem instances with many timeslots. Such problem instances can easily occur if we regard manufacturing networks with long transportation times between shops, respectively machines. We therefore implemented a new heuristic for solving the sub-problems in more time-efficient manner. For solving the dual problem, we apply a surrogate subgradient search, which allows the sub-problems to be solved approximately. Thus we define our main research contribution:

- Specification of novel problem instances for flexible job shop scheduling with transportation times between machines (FJSSTT), based on published instances for flexible job shop scheduling. These instances are used for benchmarking our algorithms.
- A Variable Neighbourhood Search (VNS) algorithm, efficiently solving job-level sub-problems with many timeslots.
- Calculation of lower and upper bounds on total weighted tardiness for the novel FJSSTT instances.

The remainder of the paper is structured as follows. In section 2 we discuss the benchmark problem instances and their extension with transportation times. The mathematical model including problem formulation and Lagrangian relaxation is described in section 3. Problem solving algorithms are discussed in section 4, and computational results are provided in section 5. Finally we present our conclusions in section 6.

2 FJSSTT PROBLEM INSTANCES

As benchmark instances for flexible job shop scheduling we use the problem instances WT1-WT5, published in (Brandimarte, 1993). Based on these instances, we generated novel FJSSTT problem instances in the following way: (1) For each WT instance the machines are randomly grouped into three job shops. (2) Short/moderate/long transportation times between the job shops were calculated. A lower bound on the makespan for the respective problem instance serves as reference time R. The transportation times are then randomly generated from the following intervals.

- Short transportation times (network type A): $(0.09\,R, 0.11\,R)$
- Moderate transportation times (network type B): $(0.9\,R, 1.1\,R)$
- Long transportation times (network type C): $(9\,R, 11\,R)$

The transportation times between the job shops satisfy the triangle inequality. Within a shop the transportation times are 0. Figure 1 provides an overview of the original instances WT1-WT5 and the generated FJSSTT instances.

Instance	Timeslots	Machines	Jobs	Transportation Times
WT1	300	5	10	-
WT1A	300	5	10	3,4
WT1B	500	5	10	32,34,37
WT1C	3000	5	10	324,326,350
WT2	300	5	20	-
WT2A	300	5	20	3,4
WT2B	1500	5	20	35,36,39
WT2C	4000	5	20	334,362,387
WT3	1500	10	20	-
WT3A	2000	10	20	24,27,29
WT3B	4000	10	20	244,250,263
WT3C	100000	10	20	2500,2512,2861
WT4	3000	10	20	-
WT4A	4000	10	20	25,27
WT4B	5000	10	20	234,242,259
WT4C	50000	10	20	2329,2616,2705
WT5	2000	15	30	-
WT5A	2500	15	30	22,24
WT5B	5000	15	30	208,219,231
WT5C	50000	15	30	2069,2199,2408

Figure 1: Problem instances.

3 MATHEMATICAL MODEL

In this section we describe a mathematical model for the FJSSTT problem with minimisation of total weighted tardiness as objective. Lagrangian relaxation is applied to the machine capacity constraints, and the resulting Lagrangian problem is decomposed into independent job-level sub-problems.

3.1 Problem Formulation

Our problem formulation is based on (Wang et al., 1997). I jobs with individual due dates have to be scheduled on M available machines. We assume immediate availability of jobs. The set of jobs I is $\{0,1,...,I-1\}$ and the set of machines $\mathcal{M}$ is $\{0,1,...,M-1\}$. Job i consists of J_i non-preemptive operations, with $\mathcal{J}_i = \{0,1,...,J_i-1\}$ denoting the set of operations for job i. The operation j of job i is denoted as (i,j). We regard simple, chain-like precedence constraints amongst operations belonging to the same job. The set of alternative machines for operation (i,j) is denoted as $\mathcal{H}_{ij}$, with machine-specific processing times. The scheduling horizon consists of K discrete timeslots, the set of timeslots $\mathcal{K}$ is $\{0,1,...,K-1\}$. The beginning time of an operation is defined as the beginning of the corresponding timeslot, and the completion time as the end of the timeslot.

In the following we introduce further parameters and decision variables used in the mathematical model. Parameters are given with a specific problem instance as input data, whereas the decision variables span the solution space for the scheduling problem.

Parameters

$D_i, i \in I$: Job due dates.

$P_{ijm}, i \in I, j \in \mathcal{J}_i, m \in \mathcal{H}_{ij}$: Processing time of operation (i,j) on machine m.

$R_{mn}, m \in \mathcal{M}, n \in \mathcal{M}$: Transportation time from machine m to machine n.

$W_i, i \in I$: Job tardiness weight.

Variables

$\delta_{ijmk}, i \in I, j \in \mathcal{J}_i, m \in \mathcal{M}, k \in \mathcal{K}$: The binary variable δ_{ijmk} is 1, if operation (i,j) is processed on machine m at timeslot k, and 0 otherwise.

$b_{ij}, i \in I, j \in \mathcal{J}_i$: Beginning time of operation (i,j).

$c_{ij}, i \in I, j \in \mathcal{J}_i$: Completion time of operation (i,j).

$m_{ij} \in \mathcal{H}_{ij}, i \in I, j \in \mathcal{J}_i$: The machine assigned to operation (i,j).

$\lambda_{mk}, m \in \mathcal{M}, k \in \mathcal{K}$: Lagrange multiplier for timeslot k on machine m.

The decision variables δ_{ijmk}, b_{ij} and c_{ij} are not independent, the following relation holds:

$$\delta_{ijmk} = \begin{cases} 1 & \text{if } b_{ij} \leq k \leq c_{ij} \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

The optimisation objective is the minimisation of the weighted sum of job tardiness, the optimisation problem is then

$$\mathcal{Z} = \min_{b_{ij}, m_{ij}} \sum_{i \in I} W_i T_i, \quad (2)$$

with

$$T_i = max(0, C_i - D_i), \quad (3)$$

where C_i is the completion time for job i, i.e. $C_i = c_{i,J_i-1}$.

Constraints Equation (2) has to be solved subject to a number of constraints. The machine capacity constraints are expressed as

$$\sum_{i \in I} \sum_{j \in \mathcal{J}_i} \delta_{ijmk} \leq 1, \forall m \in \mathcal{M}, \forall k \in \mathcal{K}. \quad (4)$$

Equation (4) states that at each timeslot a machine cannot process more than one operation. Processing time constraints define the relation between beginning time and completion time of operations:

$$c_{ij} = b_{ij} + P_{ijm_{ij}} - 1, \forall i \in I, \forall j \in \mathcal{J}_i. \quad (5)$$

The precedence constraints between job operations are

$$b_{ij} \geq c_{i,j-1} + 1 + R_{m_{i,j-1} m_{ij}}, \forall i \in I, \forall j \in \mathcal{J}_i. \quad (6)$$

The term "1" in (5) and (6) occurs due to the definition of operation beginning time and completion time, respectively. The precedence constraints consider transportation times R_{mn} between machines. For operations $(i,j-1)$ and (i,j) equation (6) states that the beginning time of (i,j) cannot be earlier than the arrival time at machine m_{ij}. We assume immediate availability of transport resources to move workpieces corresponding to jobs between machines. The transportation time between machines located in different job shops covers transport between the shops as well as shop-internal logistics activities.

The occurrence of the term $R_{m_{i,j-1} m_{ij}}$ in (6) renders the constraint non-linear. This non-linearity can easily be resolved, in fact the mathematical model can be formulated as a linear integer program, which is outside the scope of this paper.

3.2 Lagrangian Relaxation

For the FJSSTT problem there are two candidate constraint sets for relaxation: precedence constraints and machine capacity constraints. The relaxation of precedence constraints and decomposition into independent machine-level sub-problems is hampered by the structure of the precedence constraints (6), as the term $R_{m_{i,j-1}m_{ij}}$ couples the precedence constraints across machines. Lagrangian relaxation of machine capacity constraints results in the relaxed problem

$$\mathcal{Z}_D(\lambda) = \min_{b_{ij},m_{ij}} \sum_{i\in I} W_i T_i + \sum_{m\in\mathcal{M}} \sum_{k\in\mathcal{K}} \lambda_{mk} \left[\sum_{i\in I} \sum_{j\in\mathcal{J}_i} \delta_{ijmk} - 1 \right], \quad (7)$$

where λ is the vector of Lagrange multipliers. (7) has to be solved subject to constraints (5) and (6). For a given pair of indices m,k the term in brackets is positive if the capacity constraint for timeslot k on machine m is violated. $\mathcal{Z}_D(\lambda)$ can be reformulated as

$$\mathcal{Z}_D(\lambda) = \min_{b_{ij},m_{ij}} \sum_{i\in I} W_i T_i + \sum_{i\in I} \sum_{j\in\mathcal{J}_i} \sum_{k=b_{ij}}^{c_{ij}} \lambda_{m_{ij}k} - \sum_{m\in\mathcal{M}} \sum_{k\in\mathcal{K}} \lambda_{mk}. \quad (8)$$

The structure of $\mathcal{Z}_D(\lambda)$ allows the decomposition into independent job-level sub-problems

$$\mathcal{S}_i = \min_{b_{ij},m_{ij}} W_i T_i + \sum_{j\in\mathcal{J}_i} \sum_{k=b_{ij}}^{c_{ij}} \lambda_{m_{ij}k}. \quad (9)$$

$\mathcal{S}_i$ is a one job scheduling problem and can be characterised as follows, cf. (Chen et al., 1998). A job requires the completion of a set of operations, and each operation can be performed on one of several alternative machines. The job operations must satisfy a set of chain-like precedence constraints (6), considering transportation times between machines. Furthermore processing time constraints (5) have to be satisfied. Each machine has a marginal cost for utilisation at each timeslot within the scheduling horizon under consideration. The scheduling problem is to determine the machine and the completion time of each operation of the job to minimise the sum of job tardiness and the total cost of using the machines to complete the job, where the cost of using machine m at time k is given as λ_{mk}.

With the introduction of sub-problems $\mathcal{S}_i$, the relaxed problem can be reformulated,

$$\mathcal{Z}_D(\lambda) = \sum_{i\in I} \mathcal{S}_i - \sum_{m,k} \lambda_{mk}. \quad (10)$$

The Lagrangian dual problem, optimising the Lagrange multiplier values, is

$$\mathcal{Z}_D = \max_{\lambda} \mathcal{Z}_D(\lambda). \quad (11)$$

It can be shown that $\mathcal{Z}_D(\lambda)$ is concave and piece-wise linear, thus hill-climbing methods like sub-gradient search can be applied to solve the dual problem. The one job scheduling problem is not $\mathcal{NP}$-hard, and it can be exactly solved with dynamic programming, with complexity $O\left(K\sum_j \left|\mathcal{H}_{ij}\right|\right)$. However, we will see in section 5 that the efficiency of dynamic programming is not good enough to cope with FJSSTT instances with many timeslots. Thus a fast heuristic has to be developed to solve the one job scheduling problems. In the following section we will describe such a heuristic approach.

4 PROBLEM SOLVING

In this section we first describe the formulation of the dual problem, which is followed by the sub-problem solving heuristic Variable Neighbourhood Search (VNS).

4.1 Dual Problem

In order to solve the Lagrangian dual problem (11) we apply two variants of the subgradient search (SG): standard SG, and surrogate SG. The standard SG method requires the dual problem (and thus the sub-problems (9)) to be fully optimised, otherwise proper subgradient directions can not be calculated. With the dual problem fully optimised, the dual cost $\mathcal{Z}_D$ are a lower bound on total weighted tardiness. In the surrogate SG method it is sufficient to solve the dual problem approximately, and thus we can apply VNS to solve the sub-problems.

In an SG iteration l the Lagrange multipliers are adjusted for the next iteration according to

$$\lambda_{mk}^{l+1} = \lambda_{mk}^{l} + s_l \gamma_{mk}^{l}, \quad (12)$$

with a positive, scalar step-size s_l and γ_{mk}^{l}, an element of the subgradient vector γ^l,

$$\gamma_{mk}^{l} = \sum_{i\in I} \sum_{j\in\mathcal{J}_i} \delta_{ijmk}^{*} - 1. \quad (13)$$

δ_{ijmk}^{*} denotes the optimal value for δ_{ijmk} in iteration l, resulting from solving $\mathcal{Z}_D(\lambda)$ with Lagrange multipliers λ^l. The element γ_{mk}^{l} can be interpreted as violation

of the machine capacity constraint for machine m and timeslot k in SG iteration l.

In the standard SG method we use a common formula for the calculation of step-sizes:

$$s_l = \alpha_l \frac{\mathcal{Z}^* - \mathcal{Z}_D(\lambda^l)}{\|\gamma^l\|^2}, \tag{14}$$

with a scalar $\alpha_l, 0 < \alpha_l < 2$. $\mathcal{Z}^*$ is an upper bound on $\mathcal{Z}_D$ and is updated if feasibility repair improves the upper bound. The search is initialised with a parameter α_0, and α_l is halved after Γ iterations if no improvement in $\mathcal{Z}_D$ has been achieved.

The step-sizes for the surrogate SG method are calculated according to (Bragin et al., 2014):

$$s_l = \beta_l \frac{s_{l-1} \left\|\tilde{\gamma}^{l-1}\right\|}{\|\tilde{\gamma}^l\|}, \tag{15}$$

with the surrogate subgradient vector $\tilde{\gamma}^l$. The parameter β_l is adjusted according to

$$\beta_l = 1 - \frac{1}{\Omega \cdot l^\rho}, \rho = 1 - \frac{1}{l^r}, \tag{16}$$

with configuration parameters Ω and r. The step-size calculation is initialised with configuration parameter s_0. With the step-size formula (15) (Bragin et al., 2014) prove convergence of the surrogate SG method, and they show that upon convergence the lower bound property of dual cost is preserved.

4.2 Variable Neighbourhood Search

To solve a one job scheduling problem approximately we propose a Variable Neighbourhood Search (VNS) heuristics with the following neighbourhood structures. Let J denote the number of operations of the job under consideration.

Neighbourhood Structure $AM1$**:** An operation j is chosen at random, and a random alternative machine is assigned to j. If this move causes the violation of precedence constraints, it is rejected. $AM1$ is configured with a distance parameter $AM1_{Dist}$, denoting the number of operations for whom alternative machines are assigned cascadingly. Assigning an operation j to a different machine implies that the duration is changed.

Neighbourhood Structure $AM2$**:** An operation j is chosen at random, and a random alternative machine is assigned to j. The new beginning time of j and all successive operations $k > j$ is the respective earliest feasible beginning time considering the precedence constraints, plus some small, random slack. $AM2$ uses a distance parameter $AM2_{Dist}$.

Neighbourhood Structure SL**:** Applying SL to the incumbent solution, a neighbour solution results from a cascade of leftward shifts, starting with a random operation j. Let x_{SL} be the neighbour solution after shifting j, and let σ denote the available slack between operations $j-1, j$. The shift distance SL_{Dist} is a random integer from the interval $(1, \sigma)$. If $\sigma = 0$ the move SL is rejected. Otherwise in the next step of the cascade operation $j+1$ is shifted leftward, with distance SL_{Dist}. Successively the operations $j, j+1, ..., J$ are shifted leftward by SL_{Dist}. After each shift the cost of the resulting neighbour solution are evaluated according to (9). The neighbour solution with minimal cost is the result of applying SL.

Neighbourhood Structure SR**:** Analogous to SL the neighbourhood structure SR defines a cascade of rightward shifts, with a shift distance SR_{Dist}. The cascade starts with a random operation and ends with the first operation.

In the shaking phase of the proposed VNS the neighbourhood structure $AM2$ is used, with increasing values for $AM2_{Dist}$. If no improvement is achieved, the last shaking level is the generation of a new random solution "in the neighbourhood" of the incumbent solution. The term "neighbourhood" reflects the fact that the beginning time of the first operation in the random solution is within a maximal distance L to the beginning time of the first operation in the incumbent solution. However, this last level is a massive shaking of the incumbent solution, and allows to bridge broad valleys in the fitness landscape, guiding the local search to new areas in the search space.

In the local search phase the neighbourhood structures $SL \rightarrow AM1 \rightarrow SR$ are used, in the indicated sequence. Figure *Algorithm 1* shows the pseudocode of the proposed VNS algorithm. The algorithm is initialised in lines 2 - 7. An initial solution is generated at random, and configuration parameters are set. The indicated values for parameters A, B, F, L, G are exemplary. For a randomly generated solution, parameter G denotes the maximal distance between two consecutive operations.

5 COMPUTATIONAL RESULTS

In computational experiments with the problem instances outlined in figure 1 we compared the performance of two algorithms:

1. Algorithm ***StDP***: Standard SG solves the dual problem, dynamic programming is used to solve

Algorithm 1: VNS Pseudocode.

Initialisation
$x \leftarrow$ random solution
$A \leftarrow 50$ ▷ Number of VNS iterations
$B \leftarrow 50$ ▷ Number of local search iterations
$F \leftarrow 3$ ▷ The maximal value of $AM2_{Dist}$
$L \leftarrow 50$
$G \leftarrow 5$
VNS iterations
for $a \leftarrow 1, A$ **do**
  $f \leftarrow 1$
  while $f \leq F+1$ **do** ▷ Shaking loop
    if $f = F+1$ **then**
      $\widehat{x} \leftarrow$ random solution "in the neighbourhood" of x
    else
      $AM2_{Dist} \leftarrow f$
      $\widehat{x} \leftarrow AM2$neighbour solution with incumbent solution$x$
    end if
    for $b \leftarrow 1, B$ **do** ▷ Local search iterations
      for $l \leftarrow 1, 3$ **do**
        if $l = 1$ **then**
          $x' \leftarrow SL$ neighbour solution with incumbent solution $\widehat{x}$
        else if $l = 2$ **then**
          $AM1_{Dist} = 1$
          $x' \leftarrow AM1$ neighbour solution with incumbent solution $\widehat{x}$
        else if $l = 3$ **then**
          $x' \leftarrow SR$ neighbour solution with incumbent solution $\widehat{x}$
        end if
        if cost(x') $\leq$ cost($\widehat{x}$) **then**
          $\widehat{x} \leftarrow x'$
        end if
      end for
    end for
    if cost($\widehat{x}$) $<$ cost(x) **then**
      $x \leftarrow \widehat{x}$
      $f \leftarrow 1$
    else
      $f \leftarrow f+1$
    end if
  end while
end forreturn x

the sub-problems. SG configuration: $\alpha_0 = 2, \Gamma = 20$.

2. Algorithm ***SuVNS***: Surrogate SG solves the dual problem, VNS solves the sub-problems. SG configuration: $s_0 = 0.2, \Omega = 25, r = 0.1$.

Both algorithms use a list scheduling algorithm for feasibility repair. Our implementation supports concurrent computing with respect to solving sub-problems. The experiments were performed on the following computer hardware: Intel® Core™ i7-6700 CPU @ 3.40GHz, 4 cores and 8 GB ram. On the software side, the operating system was Microsoft® Windows™ 10 Pro using Java 1.8 as programming and execution environment.

Figure 2 shows the results of computational experiments with 800 SG iterations, and feasibility repair every 2 iterations. The deterministic algorithm *StDP* was executed once per problem instance. To gather statistical results for the non-deterministic algorithm *SuVNS*, 10 runs were performed with *SuVNS* per problem instance. In figure 2 column UB^* refers to

		Algorithm StDP			Algorithm SuVNS					
Instance	UB*	UB	LB	Runtime	Max UB	UB^M	Min UB	Min LB	Max LB	Runtime
WT1	57	66	43.8	13	79	71	57	43.7	44.0	36
WT1A	138	95	58.2	12	97	90	84	57.5	58.0	35
WT1B	757	573	550.0	29	584	581	574	550.2	550.4	28
WT1C	9869	9030	9004.1	1104	9023	9023	9023	9019.0	9019.1	30
WT2	252	321	128.8	23	350	328	296	125.9	126.8	70
WT2A	911	450	215.6	23	463	441	389	212.0	212.8	67
WT2B	2564	1629	1346.7	538	1695	1660	1628	1336.3	1337.0	59
WT2C	21787	17287	17193.6	5013	17288	17278	17267	17210.1	17211.4	55
WT3	128	281	92.2	1427	276	234	184	85.5	87.1	223
WT3A	4580	2244	1319.6	2920	2370	2282	2185	1310.2	1311.7	443
WT3B	40521	37501	36939.2	15494	37564	37481	37429	36938.6	36940.6	330
WT3C	414403	-	-	-	405531	405477	405402	405182.8	405184.5	194
WT4	112	112	102.2	11956	112	112	112	102.9	105.3	371
WT4A	986	529	449.3	24519	521	503	481	459.4	461.6	545
WT4B	22823	17377	16832.8	38897	17369	17305	17229	16839	16841	424
WT4C	479835	-	-	-	422277	422236	422206	421988	421989	372
WT5	166	166	166.0	15466	169	167	166	153.6	156.7	421
WT5A	3128	887	836.4	27886	905	890	875	845.7	849.8	782
WT5B	28732	17347	16205.0	164000	17289	17204	17100	16218.1	16220.1	600
WT5C	536977	-	-	-	339851	339746	339654	338873.2	338874.2	592

Figure 2: Computational results.

Instance	A	B	G	F	L
WT1	50	100	5	3	50
WT1A	50	100	5	3	50
WT1B	50	100	5	3	50
WT1C	50	100	5	3	50
WT2	50	100	5	3	100
WT2A	50	100	5	3	100
WT2B	50	100	5	3	100
WT2C	50	100	5	3	100
WT3	150	200	5	3	50
WT3A	100	150	5	3	50
WT3B	100	150	5	3	50
WT3C	50	100	5	3	50
WT4	100	100	5	3	50
WT4A	100	150	5	3	50
WT4B	100	150	5	3	50
WT4C	100	150	5	3	50
WT5	50	150	5	3	50
WT5A	100	150	5	3	400
WT5B	100	150	5	3	400
WT5C	100	150	5	3	50

Figure 3: VNS configurations.

the best known value for total weighted tardiness for the respective problem instance. For the instances WT1-5 these values are taken from (Sobeyko and Mönch, 2015), for the FJSSTT instances they are calculated with list scheduling. For algorithm *StDP* lower and upper bound on total weighted tardiness are indicated (columns LB, UB) as well as the runtime in seconds. The columns $\mathrm{Max}\,UB, UB^M, \mathrm{Min}\,UB$ indicate maximum, mean value and minimum for total weighted tardiness calculated with algorithm *SuVNS*. Minimium and maximum for lower bounds are given in columns $\mathrm{Min}\,LB, \mathrm{Max}\,LB$. The VNS algorithm is configured according to figure 3. The configurations were determined in computational experiments with sub-problems, benchmarking VNS with exact solutions calculated with dynamic programming.

Analysing the results for problem instances WT1-5, we note that the duality gap for instances indicates if algorithms *StDP* and *SuVNS* work well on the instance. The duality gaps for WT4 and WT5 are small, and the algorithms provide upper bounds very close or equal to the best known values. For WT2 and WT3 the duality gaps are distinct, and the calculated upper bounds significantly deviate from the best known values. A remarkable result is provided by *StDP* for WT5: the duality gap is 0, proving optimality of the upper bound 166.

The results for the FJSSTT instances show a strong dependence of the *StDP* runtime on the number of timeslots specified for the problem instance, cf. figure 1. For WT3C, WT4C and WT5C we were not able to calculate solutions with *StDP* and reasonable runtime. The algorithm *SuVNS* is clearly advantageous in terms of runtime, and the upper bounds are of good quality, comparing them with *StDP* results.

6 CONCLUSIONS

The computational experiments have shown promising results. However, the list scheduling method employed for feasibility repair is rather weak, and we expect a more sophisticated heuristic to improve the results. Furthermore, it is advisable to extend the computational experiments to more problem instances.

So far we have been concerned with static scheduling problems. In a dynamic scheduling problem, a schedule is executed and disturbances like transportation delays or machine failures hamper the scheduled processing of operations. It would be interesting to apply the proposed, distributed algorithm to dynamic scheduling problems, and to explore the possibilities of localised disturbance handling.

As a next step we want to improve the quality of the feasibility repair mechanism. Also more experiments with MK problems will take place. The algorithm will be extended to allow dynamic rescheduling (e.g. in case of a machine break down).

ACKNOWLEDGEMENTS

This research is funded by the Austrian Ministry for Transport, Innovation and Technology www.bmvit.gv.at through the project NGMPPS-DPC.

REFERENCES

Baptiste, P., Flamini, M., and Sourd, F. (2008). Lagrangian bounds for just-in-time job-shop scheduling. *Computers & Operations Research*, 35(3):906–915.

Bragin, M. A., Luh, P. B., Yan, J. H., Yu, N., and Stern, G. A. (2014). Convergence of the Surrogate Lagrangian Relaxation Method. *Journal of Optimization Theory and Applications*, 164(1):173–201.

Brandimarte, P. (1993). Routing and scheduling in a flexible job shop by tabu search. *Annals of Operations research*, 41(3):157–183.

Buil, R., Piera, M. A., and Luh, P. B. (2012). Improvement of lagrangian relaxation convergence for production scheduling. *Automation Science and Engineering, IEEE Transactions on*, 9(1):137–147.

Chen, H., Chu, C., and Proth, J.-M. (1998). An improvement of the Lagrangean relaxation approach for job shop scheduling: a dynamic programming method. *Robotics and Automation, IEEE Transactions on*, 14(5):786–795.

Chen, H. and Luh, P. B. (2003). An alternative framework to Lagrangian relaxation approach for job shop scheduling. *European Journal of Operational Research*, 149(3):499–512.

Hoitomt, D. J., Luh, P. B., and Pattipati, K. R. (1993). A practical approach to job-shop scheduling problems. *Robotics and Automation, IEEE Transactions on*, 9(1):1–13.

Kaskavelis, C. A. and Caramanis, M. C. (1998). Efficient Lagrangian relaxation algorithms for industry size job-shop scheduling problems. *IIE transactions*, 30(11):1085–1097.

Sobeyko, O. and Mönch, L. (2015). Heuristic Approaches for Scheduling Jobs in Large-scale Flexible Job Shops. *Computers & Operations Research*. accepted for publication.

Wang, J., Luh, P. B., Zhao, X., and Wang, J. (1997). An optimization-based algorithm for job shop scheduling. *Sadhana*, 22(2):241–256.