# A finite capacity production scheduling procedure for a Belgian steel company

**Dieter Debels** [1]
**Mario Vanhoucke** [1,2]

October 2006

2006/418

[1] *Faculty of Economics and Business Administration, Ghent University, Gent, Belgium*
[2] ***Operations & Technology Management Centre, Vlerick Leuven Gent Management School, Gent, Belgium***
mario.vanhoucke@ugent.be

D/2006/7012/63

## ABSTRACT

**Abstract**. We present a finite capacity production scheduling algorithm for an integrated steel company located in Belgium. This multiple-objective optimization model takes various case-specific constraints into account and consists of two steps. A machine assignment step determines the routing of an individual order through the network while a scheduling step makes a detailed timetable for each operation for all orders.

The procedure has been tested on randomly generated data instances that reflect the characteristics of the steel company. We report promising computational results and illustrate the flexibility of the optimization model with respect to the various input parameters.

**Keywords**. Master production scheduling; manufacturing planning and control; scheduling/sequencing.

## 1 Introduction

Due to the often complex nature of production planning, various hierarchical production planning (HPP) approaches have been presented in literature to cope with interactions between product demand, production capacity, real-time data, etc… Each hierarchical level imposes various constraints and objectives to the lower level while the lower level returns feedback about the production progress. There exist different relations of aggregation and desaggregation between the hierarchical levels, based on product unit (product parts into products into product archetypes), production unit (machine into machine group into production step into factory unit), scope (short term, medium term, long term), time unit (minutes, days, weeks, months) and decision level (shop-floor workers, production management, top management) (Venkateswaran et al., 2004). Two essential levels in many HPP systems are the Master Production Schedule (MPS), which determines medium-long term production quantities for the different products, and the Material Requirements Planning (MRP) which translates the resulting master schedule into planned start times for the product components. Early endeavours approach all HPP-systems as infinite loading systems that were insufficiently integrated with the capacity requirements. The MPS was constructed by omitting all capacity constraints and the resulting capacity infeasibilities needed to be straightened in the MRP-schedule by ex post facto capacity planning techniques such as rough cut capacity planning (RCCP) and capacity requirements planning (CRP). However, most studies reveal that these techniques are highly insufficient, and production planners induce a lead time increase while incorporating capacity requirements, leading to inferior schedules with large work-in-process inventories and high costs (Sum and Hill, 1993; Fry et al., 1992).

The need for integrating capacity limitations into planning algorithms raised in the early 1980s when many manufacturing companies started to use MRP as a primary planning tool (Rom et al., 2002). Billington et al. (1983) were the first to propose a finite capacity MRP scheduling problem in a multi-stage environment. The proposed linear programming formulation aims at computing the required production lead times in correspondence with the demands and the available capacity, thereby reducing in-process inventory compared to the usual practice in MRP. However, they did not provide any efficient heuristic procedure, capable to cope with large scale problem instances. Sum and Hill (1993) further defined the research area by also considering setup costs and tardiness costs and by including an order merging/breaking mechanism. They presented a heuristic procedure in which a schedule generation scheme is incorporated, based on the resource-constrained project scheduling problem. Diaz and Laguna (1996) used a topology of multiple parallel work centres for the production of similar products and proposed an LP problem formulation for a finite capacity MRP focusing on minimising multiple cost components. They did not provide a heuristic procedure to solve real-life problems. They stressed the importance of already incorporating resource constraints at the MPS-level and proposed some methods to improve the RCCP. Taal and Wortmann (1997) were the first to embed capacity constraints in a flexible flow shop environment, and presented a priority rule to create a schedule in line with the planning objectives. Finally, Rom et al. (2002) introduced the capacity-constrained MRP system in a job shop environment. Their LP model formulation allows flexibility to change the objective function according to specific planning goals, but does not simultaneously optimise multiple objectives.

In this paper, we present a finite capacity production scheduling algorithm for the integrated steel company Arcelor Gent (formerly known as Sidmar), located on the Ghent-Terneuzen Canal, around 20 km from the centre of the city of Ghent (Belgium). Arcelor Gent yearly produces 5 million tonnes of flat steel strip for the automotive industry and for all kinds of high-quality applications such as domestic appliances, sanitary, heating, construction and furniture and handles every step of the production process, from the supply of raw materials to the coating of steel and the production of laser-welded blanks. With 5,500 employees, they are one of the largest employers in the region. They are part of Arcelor, one of the world´s largest steel companies. For more information, we refer to www.sidmar.be.

Planning and scheduling problems in iron and steel production have not drawn as wide an attention of the operations management researchers as many other industries. However, the iron and steel industry is both capital and energy intensive, which makes the importance of effective planning and scheduling in this industry by no means less than that in other industries (Tang et al., 2001). Moreover, Lee et al. (1996) argue that tools are available to develop efficient algorithms for the extremely difficult scheduling environment of steel and research in this area should be stimulated since the return on investment for software to support improved steel making productivity is very high. Most studies focus on sub-parts of the production process (Harjunkoski et al., 2001) and ignore interactions between these sub-parts, although the objective of steel companies to reduce the lateness and lead-time of the orders increases the need for an integral approach of the problem. In our steel shop production system, we provide a solution approach for the complete production process, taking various constraints and objectives into account. We restrict our procedure to a daily bucket system, where we only assign orders to machines on a daily basis without determining the exact production sequence of the orders on a particular day. Though we already incorporate sequence dependent setup costs at this aggregated level (see section 2.2), the exact sequence of orders *within* a daily bucket needs to be determined by shop floor decisions or process-specific algorithms and is outside the scope of this paper.

The outline of the paper is as follows. Section 2 gives a problem formulation and defines the various constraints and problem objectives into detail. Section 3 presents our solution procedure to solve the problem under study. In section 4 we present extensive computational results and conclusions are given in section 5.

## 2 Problem formulation

### 2.1 Problem parameters and decision variables

A steel shop environment is closely related to a flexible flow shop as it consists of several serial production steps, each consisting of several identical machines in parallel (Lee et al., 1996). The parallel machines may be clustered into machine groups (e.g. they are located at the same place and make use of the same storage facilities). A set of accepted production orders needs to be scheduled. The ordered coils can slightly differ from each other in terms of width, thickness, length, quality, etc. (Okano et al., 2004), resulting in a wide variety of product types.

In the following, we briefly describe the various parameters and the decision variables in order to formulate the production scheduling problem. These parameters will be explained in detail and used throughout the remainder of the manuscript.

***Parameters***

<u>Steel-shop characteristics</u>

| | |
|---|---|
| $nrq$ | Number of production steps (index $q = 1, \ldots, nrq$) |
| $nrn$ | Number of machine groups (index $n = 1, \ldots, nrn$) |
| $nrj$ | Number of machines (index $j = 1, \ldots, nrj$) |
| $S_q^{NQ}$ | Set of machine groups of production step $q$ |
| $S_q^{JQ}$ | Set of machines of production step $q$ |

| | |
|---|---|
| $S_n^{JN}$ | Set of machines of machine group *n* |

Scheduling horizon

| | |
|---|---|
| *nrk* | Number of days (index $k = 1, \ldots, nrk$) |
| *nrm* | Number of weeks (index $m = 1, \ldots, nrm$) |

Order characteristics

| | |
|---|---|
| *nri* | Number of orders (index $i = 1, \ldots, nri$) |

Each order can be characterised by:

| | |
|---|---|
| $v_i$ | Volume of order *i* (in tons) |
| $t_i$ | Due date or delivery date of order *i* |
| $O_i^Q$ | Set of production steps needed to produce order *i* ($O_i^Q \subset \{1, \ldots, nrq\}$) |

The orders can be aggregated in order groups or production flows, as follows:

| | |
|---|---|
| *nrl* | Number of order groups (production flows) for the flow constraints (index $l = 1, \ldots, nrl$) |
| $F_l^O$ | Set of production orders that belong to production flow *l* |
| $f_{lnm}$ | Pre-specified flow quantity (in tons) for order group *l* on machine group *n* during week *m* |

Order routing network

Durations:

| | |
|---|---|
| $p_{ij}$ | Processing time of order *i* on machine *j* (in minutes) |
| | = function of order *i*, machine *j* and volume $v_i$ |
| $d_{jj'}$ | Intermediate duration for order *i* between machine *j* and machine *j'* (in days) |
| | = transportation and minimal cooling down time |
| $c_{jk}$ | Capacity of machine *j* on day *k* (in minutes) |

Costs:

| | |
|---|---|
| $a_{ijj'}$ | Assignment cost of order *i* (in €) if assigned to machines *j* and *j'* |
| | = sum of costs of transportation and cooling down |
| $e_i$ | Earliness unit cost for order *i* (in € per day) |
| $l_i$ | Lateness unit cost for order *i* (in € per day) |
| $u_j$ | Utilization cost of machine *j* (in € per minute deviation of 100% utilization) |
| $b_{ln}^-$ | Lower flow penalty cost (scheduled production $\leq f_{lnm}$) (in € per ton) |
| $b_{ln}^+$ | Upper flow penalty cost (scheduled production exceeds $f_{lnm}$) (in € per ton) |

***Decision variables***

A production schedule consists of an assignment of all operations for each order taking various constraints (section 2.2) and multiple objectives (section 2.3) into account. Hence, every operation of order *i* needs to be executed on a machine *j* on a particular day *k*, resulting in the decision variables as follows:

$x_{ijk}$ = 1, if an operation of order *i* is assigned to machine *j* on day *k*

= 0, otherwise

Note that we construct a production schedule where the daily machine capacity and processing times of the orders are expressed in minutes whereas the intermediate time between machines is expressed in days. Hence, for the production of steel, time spent for operations at the machinery lies far beyond the time needed to support intermediate manipulations such as transporting, heating or cooling down.

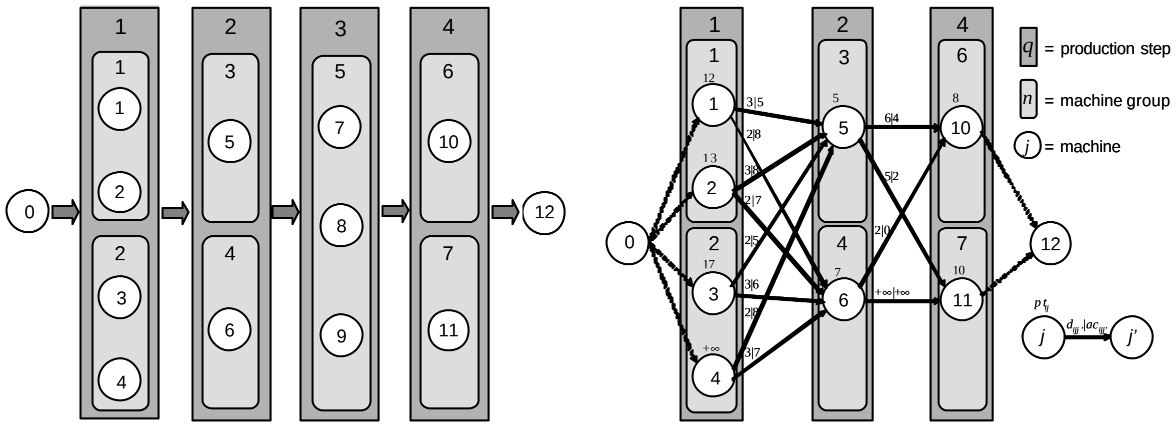

**Figure 1.** An example steel shop (left (a)) and an example order routing network (right (b))

Figure 1(a) displays an example steel shop network with 4 production steps, 7 machine groups and 11 machines. Dummy machine 0 and dummy machine 12 (or in general *nrj* + 1) are used to represent the start and finishing time of an order. Figure 1(b) displays the *order routing network* of an example order for which the routing is limited to production steps 1, 2 and 4. In order to process all operations of the order, the algorithm must select a single path from this network consisting of a sequence of machines. The production steps of a coil of steel consist of casting, hot rolling, pickling, and cold rolling. In order to give specific properties to the coil, extra operations such as annealing, skin passing, galvanizing, coating, recoiling or cutting can be performed. Each production step has many typical production constraints and solving a steel shop scheduling problem involves handling a large number of complicated chemistry-, geometrical- and scheduling rules.

## 2.2 Problem constraints

In this section, we discuss the four different types of technical restrictions (capacity, assignment, precedence and setup constraints) that the steel company incorporates in its production schedule.

**Capacity constraints:** Each machine $j$ has a limited capacity $c_{jk}$ expressed in minutes per day, which may not be exceeded by all assigned orders on that machine at day $k$. In order to avoid that this constraint leads to a structural under-use of the available machine capacity, we add the unused capacity $\Delta c_{jk-1}$ of the previous day to the capacity $c_{jk}$. However, the capacity shift $\Delta c_{jk-1}$ is limited to a threshold value $\tau$ such that a temporal shortage of eligible orders at day $k$ - 1 does not create unrealistic capacity at day $k$. The capacity constraints can be formulated as follows:

$$\sum_{i=1}^{nri} pt_{ij} x_{ijk} \leq c_{jk} + \Delta c_{jk-1} \qquad j = 1, \ldots, nrj \text{ and } k = 1, \ldots, nrk \qquad [1]$$

with

$$\Delta c_{jk} = \min\left( t, c_{jk} + \Delta c_{jk-1} - \sum_{i=1}^{nri} pt_{ij} x_{ijk} \right) \qquad [2]$$

In our production scheduling algorithm, we set $t$ equal to 10 minutes.

**Assignment constraints:** In order to process all operations of an order, the algorithm must select a unique path from the order routing network consisting of a sequence of machines. Therefore, each operation of order $i$ needs to be assigned to one machine $j$ per production step $q \in O_i^Q$, as follows:

$$\sum_{j \in S_q^{JQ}} \sum_{k=1}^{nrk} x_{ijk} = 1 \qquad i = 1, \ldots, nri \text{ and } \forall q \in O_i^Q \qquad [3]$$

**Precedence constraints:** The precedence relations between operations of an order $i$ are shown by the order routing network and define the relations between all machines $j_1 \in S_{q_1}^{JQ}$ and $j_2 \in S_{q_2}^{JQ}$ of two sub-sequent production steps $q_1 \in O_i^Q$ and $q_2 \in O_i^Q$ with a minimal time interval equal to the duration $d_{ij_1j_2}$. Hence, for each couple $(q_1, q_2) \in O_i^Q$ of two subsequent production steps in the routing of an order $i$, the algorithm incorporates the precedence relations as follows:

$$\sum_{k=1}^{nrk} \left( k x_{ij_1k} \right) + d_{ij_1j_2} \leq \sum_{k=1}^{nrk} k x_{ij_2k} \qquad i = 1, \ldots, nri, j_1 \in S_{q_1}^{JQ} \text{ and } j_2 \in S_{q_2}^{JQ} \qquad [4]$$

**Setup constraints:** Our production scheduler assigns orders to machines on a daily basis and does not determine the exact sequence of the individual orders. The setup constraints take sequence-dependent setup costs or transition costs into account by imposing campaigns. Orders with similar characteristics will be grouped in production campaigns, which is a production run with specific start and end times in which coils of a particular type are processed continuously on a process line (Okano et al., 2004). As an example, it is beneficial to start a campaign of thin coils at the cold rolling mill when new rollers are installed, since the thicker coils can be rolled when rollers start to wear out. In our problem formulation, we cluster orders with low mutual setup costs for a particular machine and ensure that only order *i* of this campaign can be scheduled on machine *j* on day *k* and prevents the assignment of all other orders *i'*, as follows:

$$\sum_{\forall i'} x_{i'jk} = 0 \qquad [5]$$

**2.3 Problem objective function**

Previous research studies reveal that the multiple objectives that are used in the steel making industry are often very company-specific. Lee et al. (1996) focus on full capacity use to make the expensive machinery pay, raised the issue of allocating orders efficiently among parallel machines and suggested to group all orders for coils with similar processing properties in order to reduce setup costs. Okano et al. (2004) take a customer satisfaction point of view by minimising the lateness of the orders. Moreover, they consider various technical constraints by production campaigns. Wiers (2002) focuses on stock quantity reduction by means of lead time minimisation. Our production scheduler optimises a multiple objective function by minimising the total cost *TC* as a sum of five different cost functions: assignment cost $C^A$, lateness cost $C^L$, earliness cost $C^E$, utilisation cost $C^U$ and production flow cost $C^F$, i.e. $TC = C^A + C^L + C^E + C^U + C^F$.

**Assignment cost:** The total assignment cost is an immediate result of the selection of paths in the order routing networks. An assignment cost $ac_{ijj'}$ is charged for each arc (*j*, *j'*) of the selected path of an order *i*, as follows:

$$C^A = \sum_{i=1}^{nri} \sum_{j=1}^{nrj} \sum_{j'=j+1}^{nrj} ac_{ijj'} \sum_{k=1}^{nrk} x_{ijk} \sum_{k'=k+1}^{nrk} x_{ij'k'} \qquad [6]$$

**Lateness cost:** The lateness cost penalises the production orders that finish later than the pre-negotiated due date $t_i$, and equals the unit lateness cost $l_i$ times the number of days order *i* is late (or zero, if the order finishes earlier than $t_i$). The algorithm determines the finishing day *k* of order *i* as the assignment of the order on the dummy end machine *nrj* + 1, and hence the lateness cost can be calculated as follows:

$$C^L = \sum_{i=1}^{nri} l_i \max(0, \sum_{k=1}^{nrk} k x_{i\,nrj+1\,k} - t_i) \qquad [7]$$

**Earliness cost:** The earliness cost incorporates the lead-time and stock level minimisation and states that orders should be started no earlier than necessary to finish within the pre-negotiated due date $t_i$. Therefore, the algorithm calculated the latest possible starting time $LST_{i0}$ by means of simple backward calculations, starting from the end dummy machine $nrj + 1$. The earliness cost is equal to the unit earliness cost $e_i$ times the number of days the order $i$ starts earlier than its $LST_{i0}$ (or zero, if the order starts after its $LST_{i0}$).

$$C^E = \sum_{i=1}^{nri} e_i \max(0, LST_{i0} - \sum_{k=1}^{nrk} k x_{i0k}) \qquad [8]$$

**Utilisation cost:** The utilisation cost penalises each time unit (minute) a machine is idle. Hence, the algorithm measures the deviation between the daily machine capacity $c_{jk} + \Delta c_{jk-1}$ and the capacity use $\sum_{i=1}^{nri} p_{ij} x_{ijk}$ of the assigned orders, such that the total utilisation cost can be calculated as follows:

$$C^U = \sum_{j=1}^{nrj} \sum_{k=1}^{nrk} u_j \left( c_{jk} + \Delta c_{jk-1} - \sum_{i=1}^{nri} p_{ij} x_{ijk} \right) \qquad [9]$$

**Production flow cost:** The production of steel requires primary resources (i.e. machines) as well as secondary resources, such as colourings or chemical additives. An efficient stock management of these secondary resources leads to substantial cost reductions. Hence, the steel company clusters orders that use the same secondary resources at particular production steps in order groups or *production flows*. This allows the determination of a pre-specified flow quantity $f_{lnm}$ for order group $l$ on machine group $n$ during week $m$ at the MPS level. Our production scheduler takes these flow constraints into account by penalizing deviations (either below ($b_{ln}^-$) or above ($b_{ln}^+$)) between the scheduled order volumes and the pre-specified flow volumes. Thanks to these production flow quantities, the steel company can order the corresponding secondary resources on a just in time basis, avoiding excessive safety stocks. The production flow cost can be modelled as follows:

$$C^F = \sum_{l=1}^{nrl} \sum_{n=1}^{nrn} \sum_{m=1}^{nrm} \begin{cases} \Delta f_{lnm} b_{ln}^- \text{ if } \Delta f_{lnm} \geq 0 \\ -\Delta f_{lnm} b_{ln}^+ \text{ if } \Delta f_{lnm} < 0 \end{cases} \qquad [10]$$

where $nrl$ has been previously defined as the number of production flows and $f_{lnm}$ the pre-specified flow quantity (in tons) for order group $l$ on machine group $n$ during week $m$. Note that we consider, without loss of generality, weekly buckets to describe the pre-specified production flow volumes. The flow deviation $\Delta F_{lnm}$ for flow $l$ at machine group $n$ during week $m$ can be calculated as

$$\Delta f_{lnm} = f_{lnm} + \Delta f_{lnm\text{-}1} - \left( \sum_{i \in F_l^O} v_i \sum_{j \in S_n^{JN}} \sum_{k \in W_m} x_{ijk} \right) \qquad [11]$$

with $W_m$ the set of days belonging to week $m$.

## 3 Solution approach

In this section, we describe our solution algorithm to solve the production scheduling problem under study. Our solution approach consists of two steps, taking the various constraints and objectives into account, as follows:

Step 1. Machine assignment problem: each order is assigned to one machine for each production step of its routing (assignment constraint), resulting in the total assignment cost $C^A$.
Step 2. Scheduling problem: all operations of each order need to be scheduled on a particular day (given the assigned machines of step 1), taking the three remaining constraints (capacity constraints, precedence constraints, setup constraints) and the multiple objectives (lateness costs, earliness costs, utilisation costs and flow costs) into account.

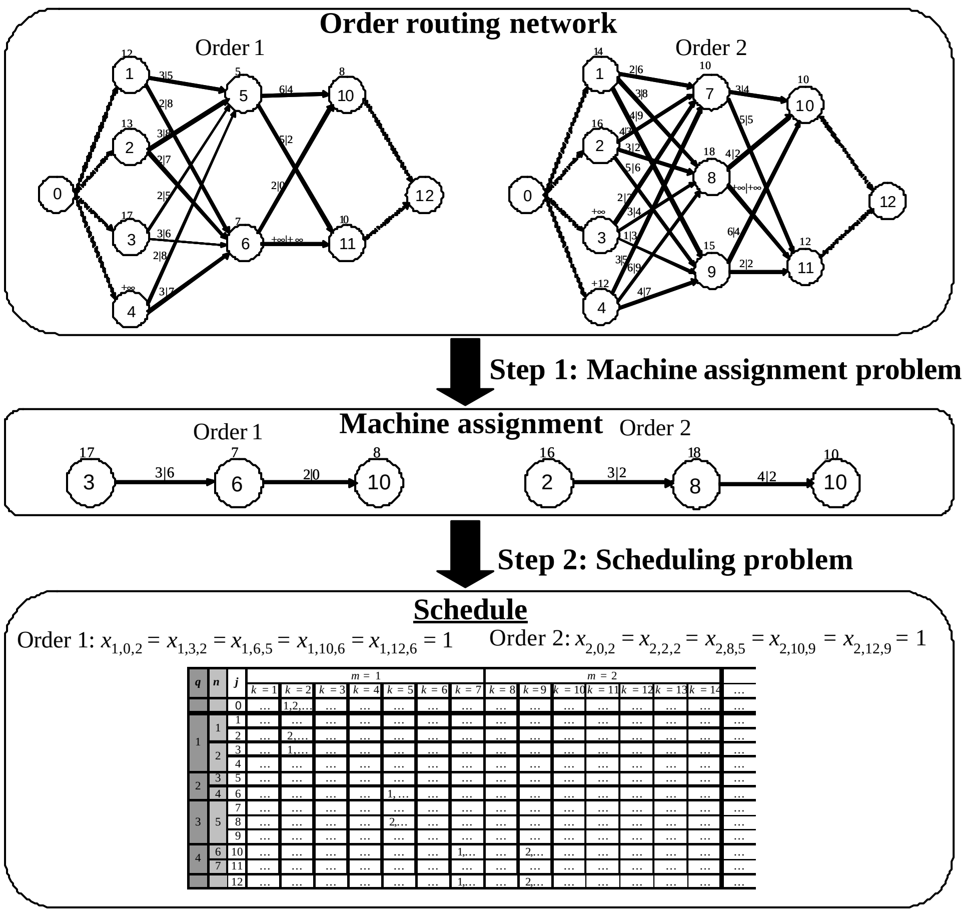
Order routing network
Order 1
Order 2
Step 1: Machine assignment problem
Machine assignment
Order 1
Order 2
Step 2: Scheduling problem
Schedule
Order 1: $x_{1,0,2} = x_{1,3,2} = x_{1,6,5} = x_{1,10,6} = x_{1,12,6} = 1$
Order 2: $x_{2,0,2} = x_{2,2,2} = x_{2,8,5} = x_{2,10,9} = x_{2,12,9} = 1$

**Figure 2.** Our solution approach

Figure 2 displays a fictive example with two orders. In step 1, each order is assigned to a specific machine on its routing (path 3 – 6 – 10 for order 1 and path 2 – 8 – 10 for order 2). These assigned machine paths will be used as an input to solve the scheduling problem of step 2. In figure 2, both orders start on day 2 and have been scheduled as soon as possible. In section 3.1, we discuss the scheduling problem (step 2) in detail. Section 3.2 elaborates on the machine assignment problem (step 1).

## 3.1 Scheduling problem

In order to schedule all orders in time, the algorithm solves a knapsack problem for each machine and for each day of the scheduling horizon. Hence, our *schedule generation scheme* (SGS) iterates over all machines and all days, and can be shown in pseudo-code, as follows:

```
For $k$:=1 to $nrk$
    For $j$:=1 to $nrj$
        Knapsack Problem ($j$,$k$)
```

The knapsack problem determines for each machine and each day the set of *eligible orders* $O_{jk}^E$ that are potential candidates to enter the knapsack (i.e. scheduled on machine $j$ during day $k$). An order $i$ is eligible on machine $j$ on day $k$ if the following constraints are satisfied:

- Assignment constraints: an order can only be scheduled on a machine determined by the machine assignment problem of section 3.2,
- Precedence constraints: an order can only be scheduled if the previous operation of that order has been scheduled earlier, taking the intermediate duration into account,
- Setup constraints: an order can only be scheduled within the production campaign restrictions.

Hence, the knapsack problem boils down to the selection of orders to be scheduled on machine $j$ of day $k$, satisfying the capacity constraints and optimizing the various costs factors of section 2.2.

$$\max\ TCR = \sum_{i \in O_{jk}^E} CR_{ijk}^L + CR_{ijk}^E + CR_{ijk}^U + CR_{ijk}^F \qquad [12]$$

subject to

$$\sum_{i \in O_{jk}^E} pt_{ij} x_{ijk} \leq c_{jk} + \Delta c_{jk-1} \qquad [13]$$

The objective function maximises the total cost reduction $TCR$ when assigning order $i$ to machine $j$ on day $k$. Since the cost of scheduling that order depends on the schedule of all other orders, we need to estimate the total cost reduction $TCR$ when assigning order $i$ to machine $j$ on day $k$, denoted by $CR_{ijk}^L$ (estimated lateness cost reduction), $CR_{ijk}^E$ (estimated earliness cost reduction), $CR_{ijk}^U$ (estimated utilisation cost reduction) and $CR_{ijk}^F$ (estimated production flow cost reduction). The determination of these estimates will be discussed in the remainder of this section. The constraint of equation [13] is equal to the capacity constraint of machine $j$ on day $k$ of equation [1]. The knapsack problem is proven to be an NP-complete problem (Hirschberg and Wong, 1976).

**Estimated lateness cost reduction:** The lateness cost depends on the due date of order $i$ and can only be determined when the last operation of order $i$ has been scheduled. Hence, an estimate for the lateness cost reduction $CR_{ijk}^L$ of assigning order $i$ to machine $j$ on day $k$ is equal to the extra lateness cost we would obtain when postponing it to day $k$ + 1. Hence, we need to rely on an estimate of the probability $P_{ijk}$ as the chance that an order $i$ will increase the lateness by one day if the order is not scheduled on day $k$. Consequently, the estimated lateness cost reduction is equal to $CR_{ijk}^{L} = P_{ijk} l_i$ and is displayed in figure 3.

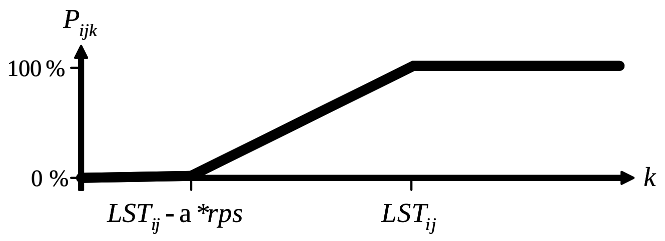

**Figure 3.** An estimate for order lateness

The probability $P_{ijk}$ is assumed to be 100% when the scheduling day $k$ is larger then or equal to the latest start time $LST_{ij}$. This $LST_{ij}$ can be easily determined by means of straightforward backward calculations starting from the order due date $t_i$. Moreover, we assume that this probability is a function of the remaining operations (denoted as the remaining production steps $rps$) of the order and a slack-per-operation parameter a. Therefore, we define a linear function such that the probability increases linearly from 0% to 100% between $LST_{ij}$ – a*$rps$ = $k$ = $LST_{ij}$ . Outside the interval [$LST_{ij}$ – a*$rps$, $LST_{ij}$], the probability equals 0% ($k$ = $LST_{ij}$ – a*$rps$) or 100% ($k \geq LST_{ij}$). In our production scheduling algorithm, we set a equal to 1.

**Estimated earliness cost reduction:** The earliness cost depends on the start of the first operation of order $i$ and is measured as the deviation between the start of the first operation and the latest start time $LST_{i0}$ of this operation. Hence, the estimated earliness cost reduction can be calculated as follows:

$$CR_{ijk}^{E} = \begin{cases} 0 & \text{if } k \geq LST_{i0} \text{ or if } j > 0 \\ -e_i, & \text{otherwise} \end{cases} \qquad [14]$$

**Estimated utilisation cost reduction:** Each order $i$ that enters the knapsack needs to be produced on machine $j$ on day $k$, and hence, increases the utilisation by $pt_{ij}$ minutes (and reduces the utilisation cost by $u_j$ per minute). Hence, the estimated utilisation cost reduction can be calculated as follows:

$$CR_{ijk}^{U} = u_j pt_{ij} \qquad [15]$$

**Estimated production flow cost reduction:** Each order $i$ of production flow $l$ ($i \in F_l^O$) that enters the knapsack to be scheduled at machine $j$ on day $k$ will affect the production flow deviation $\Delta f_{lnm}$ (see equation [11]) of machine group $n$ (with $j \in S_n^{JN}$) during the week $m$ ($k \in W_m$).

We define for each order $i$ a density measure $d_i = \dfrac{CR_{ijk}^L + CR_{ijk}^E + CR_{ijk}^U + b_{ln}^- v_i}{pt_{ij}}$ which measures the total estimated cost reduction per time unit (in minutes) if an order $i$ enters the knapsack at machine $j$ on day $k$. The last term states that the entrance of an eligible order $i$ reduces the production flow cost by $b_{ln}^- v_i$ and hence, assumes that the entrance of eligible order $i$ results in a reduction of the flow deviation (there is a flow 'under-production') (note that the knapsack problem strives for a maximal knapsack density and hence, priority will be given to orders with a high value for the density measure). However, some orders can enter the knapsack resulting in an increase of flow deviation (in case of a flow 'over-production') and hence, an estimate for the production flow deviations needs to be calculated.

We calculate an estimate of the flow deviation equations [11] for each order $i$ (denoted by $\Delta EF_i$) under the assumption that

- The orders $i'$ of the same production flow $l$ (i.e. $i' \in F_l^O$) on machine $j'$ that have been scheduled by the algorithm on previous days $k' < k$ of the same week are already included.
- All eligible orders $i''$ of the same production flow $l$ (i.e. $i$ and $i'' \in F_l^O$) with a higher or equal density value of order $i$ ($d_{i''} \geq d_i$) will be scheduled (i.e. entering the knapsack) prior to scheduling order $i$, as

$$\Delta EF_i = \underbrace{f_{lnm} + \Delta f_{lnm\text{-}1}}_{(a)} - \underbrace{\sum_{i' \in F_l^O} v_{i'} \sum_{j' \in S_n^{JN}} \sum_{\substack{k' \in W_m \\ k' < k}} x_{i'j'k'}}_{(b)} - \underbrace{\sum_{\substack{i'' \in F_l^O \\ d_{i''} \geq d_i}} v_{i''}}_{(c)} \qquad [16]$$

with (a) the pre-specified production flow volumes, (b) the production volume of all scheduled orders $i'$ and (c) the production volume of all orders $i''$ that will probably be scheduled prior to scheduling order $i$ plus the production volume of order $i$.

The value of $\Delta EF_i$ reveals whether or not scheduling order $i$ on machine $j$ on day $k$ will lead to overproduction of production flow $l$. If $\Delta EF_i$ = 0, then scheduling order $i$ will probably reduce the underproduction of flow *l*. If $\Delta EF_i$ = -$v_i$, then scheduling order *i* will probably increase the overproduction of *l*. If -$v_i$ < $\Delta EF_i$ < 0, then scheduling order *i* will probably change the underproduction of *l* to an overproduction. Hence, we rely on this flow deviation estimate to determine the production flow cost reduction $CR^F_{ijk}$ as follows:

$$CR^F_{ijk} = \begin{cases} b^-_{ln} v_i & \text{if } \Delta EF_i \geq 0 \\ -b^+_{ln} v_i & \text{if } \Delta EF_i \leq -v_i \\ b^-_{ln}(v_i + \Delta EF_i) + b^+_{ln} \Delta EF_i & \text{if } -v_i \leq \Delta EF_i < 0 \end{cases} \qquad [17]$$

Figure 4 illustrates the production flow reduction estimate based on an example project of our production scheduler with 9 eligible orders on the third day of a particular week with 3 pre-specified production flows for which $F_1^O$ = {1, 6, 8}, $F_2^O$ = {2, 3, 7} and $F_3^O$ = {4, 5, 9}. The order volumes and density matrix of the orders are assumed to equal to (8, 7, 11, 12, 9, 9, 6, 13, 8) and (1.10, 0.93, 0.90, 0.85, 0.80, 0.76, 0.67, 0.63, 0.57), respectively.

| $l$ | $f_{lnm} + \Delta f_{lnm\text{-}1}$ | $b^-_{ln}$ | $b^+_{ln}$ | $\sum_{i' \in F_l^O} v_{i'} \sum_{j' \in S_n^{JN}} \sum_{k' \in W_m, k' < k} x_{i'j'k'}$ |
|---|---|---|---|---|
| 1 | 40 | 4 | 8 | 10 |
| 2 | 30 | 2 | 6 | 20 |
| 3 | 50 | 8 | 10 | 52 |

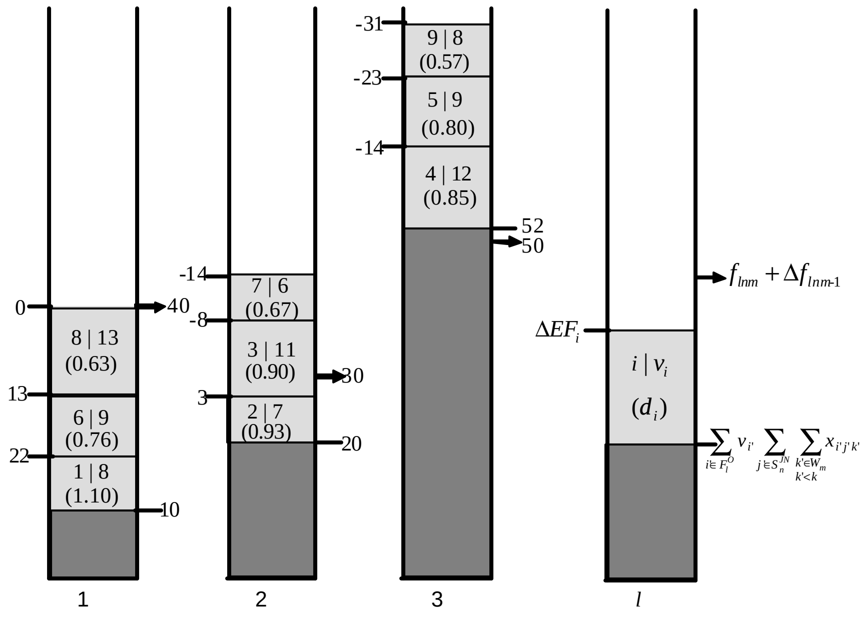

**Figure 4.** A fictive example to illustrate the calculation of the production flow cost reduction

The black areas represent the production volume of all scheduled orders *i'* on previous days of the week for each flow. The density measure serves as a priority estimate for each order to be selected in the knapsack (see figure 4), as follows:

- Flow 1: 1 – 6 – 8: the entrance of all these orders leads to a decrease of the production flow deviation, and hence, the estimate of the flow deviation $\Delta EF_i$ will be zero or positive.
- Flow 2: 2 – 3 – 7: the entrance of order 2 leads to a decrease of the production flow deviation ($\Delta EF_i \geq 0$). The entrance of order 3 will change the underproduction of flow 2 to an overproduction ($-v_i < \Delta EF_i < 0$) while the entrance of order 7 will increase the production flow deviation $\Delta EF_i = -v_i$.
- Flow 3: 4 – 5 – 9: the entrance of all these orders leads to an increase of the production flow deviation, and hence, all estimates $\Delta EF_i = -v_i$.

The above estimates of the flow deviation equations $\Delta EF_i$ will be used to calculate the $CR^F_{ijk}$ values (32, 14, -42, 120, -90, 36, -36, 52, -80) which will be used in the objective function of the knapsack problem.

## 3.2 Machine assignment problem

The basic machine assignment problem randomly selects for each order a single path from the order routing network consisting of a sequence of machines. However, the algorithm is able to control the machine assignment process by estimating cost factors in three various ways, as follows:

**Greedy assignment (based on assignment cost) (GA(A)):** Each order will be assigned to the path with the lowest assignment cost using a shortest path algorithm of Dijkstra (1959).

**Greedy assignment (based on assignment, utilisation and production flow cost) (GA(AUF)):** The orders will be assigned, one after another, to the path with the lowest total cost using the shortest path algorithm of Dijkstra (1959). The total cost is equal to the assignment cost (similar to the GA(A) approach) plus a penalty estimate for the utilisation and production flow cost. The penalty costs of each arc $(j, j')$ are calculated based on the assignments of all previously assigned orders. To obtain utilisation and production flow estimations, we assume that these orders have been scheduled at their latest start time $LST_{ij}$.

- The utilisation penalty cost for arc $(j, j')$ of order $i$ is equal to $u_j$ when the cumulative available capacity of machine $j$ before $LST_{ij}$ has been reserved already completely by the previously assigned orders, and equals zero otherwise.
- The production flow penalty cost for arc $(j, j')$ of order $i$ equals $b^+_{ln}$ when the cumulative pre-specified production flow quantities remaining has been reserved already completely by the previously assigned orders, and equals zero, otherwise.

Consequently, positive penalty factors for an arc $(j, j')$ give the shortest path algorithm an incentive to select another arc of another path in the order routing network for order *i*.

**Local search assignment (LSA):** The local search procedure embeds the SGS of section 3.1 in a local search procedure in order to find high quality schedules. The procedure starts with an initial schedule, constructed by the GA(A) approach and the SGS procedure, and searches for improvements by iteratively changing the machine assignments of a single order *i*. The pseudo-code of the local search procedure can be displayed as follows:

```
Procedure LSA()
  Construct initial schedule
  For $k$ = 1 to $nrk$
    For $i$ = 1 to $nri$
      For $q$ = 1 to $nrq$
        For $j_1$ = first machine to last machine of $S_q^{JQ}$
          If $x_{ij_1k} = 1$
            For $j_2$ = first machine to last machine of $S_q^{JQ}$
              change assignment of order $i$ from $j_1$ to $j_2$
              If "check constraint feasibility" then $\Delta$costs = "phase 1 cost estimate"
                If $\Delta$costs < 0 then costs = SGS()
                  If costs < best found costs then replace best found schedule
                  Else change assignment of $i$ again from $j_2$ to $j_1$.
```

The local search procedure iteratively searches for each day $k$ whether a re-assignment of an order would lead to improvements. Therefore, the algorithm considers all possible re-assignments of an order from machine $j_1$ to machine $j_2$ within a production step $q$, and checks the resulting constraint feasibility and the resulting cost changes, as follows:

- *Check constraint feasibility*: this sub-routine checks whether all constraints are satisfied when re-assigning order *i* from machine $j_1$ to machine $j_2$ of the same production step $q$.
- *New cost estimation*: The machine assignment change will lead to a new cost, that is estimated in two phases.
    - *Phase* 1. *quick and rough estimate*: this cost estimate is a quick and rough estimate to evaluate whether the order re-assignment is a valuable alternative that needs further detailed analysis. The "phase 1 cost estimate" is equal to the change in the assignment cost plus the new utilisation and flow cost. For the latter two, the algorithm simply calculates the cost changes by ignoring the cost effect on the rest of the schedule. If the

rough estimate shows a cost decrease, the algorithm calculates the more detailed cost estimate in phase 2 to decide whether the re-assignment will be executed.

- *Phase* 2. *estimate cost by SGS* (only if phase 1 gives an indication that a re-assignment would be beneficial): The schedule generation scheme schedules the order on the new machine $j_2$ and calculates the resulting cost as described in section 3.1. When the resulting schedule cost is lower than the current schedule cost, the new schedule replaces the previous one and the algorithm continues its search.

Ideally, this process continues until all days have been considered. In order to limit the computational effort, the algorithm will be truncated after 100 generated schedules.

## 4 Experimental results

In this section, we report detailed computational results of different versions of our solution procedure. All procedures have been programmed in Visual C++ 6.0 and tested on an Acer Travelmate 634LC with a Pentium IV 1.8 GHz processor. We rely on a self-generated test set of 50 problem instances explained in section 4.1. Section 4.2 reports detailed computational results for the various machine assignment procedures and the schedule generation scheme. In section 4.3, we illustrate the flexibility of the schedule generation scheme and the use of the various penalty costs that can be modified to create a schedule that fulfils company-specific objectives.

### 4.1 Generation of problem instances

In order to generate problem instances, we have developed an automatic problem generator taking the various problem parameters as pre-specified input values. We vary the number of orders (*nri*) from 1,000; 2,000; 4,000; 8,000 to 16,000 and generate 10 problem instances per setting, resulting in 50 problem instances in total. The number of orders as well as all other parameters have been set based on the investigation of real-life data available at the company. All other parameters are fixed as follows:

<u>Steel-shop characteristics</u>: $nrq = 8$, $nrn = 12$, $nrj = 20$, $S_q^{NQ}$ has been created by random assignments of machine groups for each production step $q$ and $S_n^{JN}$ has been constructed by random assignments of machines for each machine group $n$.

<u>Scheduling horizon</u>: $nrk = 21$ days and $nrm = 3$ weeks

<u>Order characteristics</u>: $v_i$ = rand[20, 40] (in tons), $t_i$ = rand[3, 2*$nrk$] and $O_i^Q$ has been created by random assignments of production steps to each order $i$ (minimum 3 production steps per order). We distinguished $nrl = 10$ order groups or production flows and also for the assignment of orders to order

groups we relied on randomness. In order to generate realistic production flow quantities, the values for $f_{lnm}$ (in tons) have been generated based on a simulated schedule for each problem instance.

Order routing network:

Durations: $pt_{ij}$ = rand[0.9 $v_i$, 1.1 $v_i$] (in minutes), $d_{jj'}$ = rand[0, 7] (in days). Similar to the flow quantities, the daily machine capacities $c_{jk}$ (in minutes) have been generated based on a simulated schedule for each problem instance.

Costs: $a_{ijj'}$ = rand[0, 99] (€), $e_i$ = 4 $v_i$ (€ per day), $l_i$ = rand[0, 3] $v_i$ (€ per day), $u_j$ = 8 (€) and $b^-_{ln}$ = $b^+_{ln}$ = 8 (€).

## 4.2 Computational performance of our solution approach

Table 1 displays the performance of the schedule generation scheme (SGS) by solving the knapsack problem with an exact and a heuristic approach. More precisely, the exact branch-and-bound procedure of Kolesar (1967) is compared with a straightforward greedy search heuristic in which eligible orders are chosen in decreasing order of their density measure. The row with label "Avg. CPU" displays the average CPU time in seconds and the row with label "Avg. *TC*" displays the average total cost. The total costs consist of the individual cost factors of section 2.3 (Avg. $C^A$, Avg. $C^L$, Avg. $C^E$ and Avg. $C^F$), as displayed in the remaining rows. Note that the machine assignment problem has been solved by randomly assigning each order to one machine for each production step of its routing. This machine assignment problem has been repeated ten times.

| ***nri*** | **1000** | | **2000** | | **4000** | | **8000** | | **16000** | |
|---|---|---|---|---|---|---|---|---|---|---|
| | **exact** | **heuristic** | **exact** | **heuristic** | **exact** | **heuristic** | **exact** | **heuristic** | **exact** | **heuristic** |
| **Avg. CPU** | 0.04s | 0.04s | 0.08s | 0.07s | 0.18s | 0.16s | 0.47s | 0.38s | 1.70s | 0.84s |
| **Avg. *TC*** | 448,739 | 453,800 | 802,156 | 809,077 | 1,510,731 | 1,519,521 | 2,928,450 | 2,937,001 | 5,721,389 | 5,730,693 |
| **Avg. $C^A$** | 126,557 | 126,557 | 253,846 | 253,846 | 506,713 | 506,713 | 1,009,435 | 1,009,435 | 2,024,264 | 2,024,264 |
| **Avg. $C^L$** | 23,127 | 22,829 | 30,735 | 30,751 | 53,344 | 53,897 | 90,705 | 90,806 | 166,188 | 166,402 |
| **Avg. $C^E$** | 29,339 | 28,828 | 64,943 | 64,218 | 132,080 | 131,066 | 277,023 | 275,966 | 565,306 | 564,322 |
| **Avg. $C^U$** | 129,451 | 135,275 | 212,094 | 219,668 | 374,385 | 383,357 | 701,847 | 711,758 | 1,368,577 | 1,379,292 |
| **Avg. $C^F$** | 140,264 | 140,309 | 240,536 | 240,593 | 444,208 | 444,487 | 849,440 | 849,035 | 1,597,052 | 1,596,411 |

**Table 1.** Comparison between exact and heuristic knapsack procedure

The table reveals that both the exact and the heuristic solution procedures for the knapsack problems are able to provide solutions within a reasonable time limit. The heuristic approach is able to generate high quality knapsack solutions (see the small Avg. *TC* deviations between the exact and heuristic approach) but the extra CPU time the exact approach needs is relatively small. Hence, in the remainder of this paper, we rely on the exact approach of Kolesar (1967) to solve the knapsack problems.

Table 2 compares the performance of the various machine assignment procedures of section 3.2. The table clearly reveals that more sophisticated assignments such as the greedy assignment GA(AUF) and the local search assignment LSA result in larger CPU times. But the resulting schedule quality of those assignment strategies outperforms the simple GA(AC) and the random assignment. The results can be summarised as follows:

**LSA versus GA(AC):** The LSA approach clearly outperforms the GA(AC) approach at the expense of a much larger CPU time. However, in relative terms, the improvement of LSA compared to the GA(AC) decreases from 8.48% for 1,000 orders to only 0.73% for 16,000 orders. Note that the LSA approach always has a higher assignment cost (the GA(AC) only takes the assignment cost into account) but a lower average total cost.

**GA(AUF) versus GA(AC):** The GA(AUF) approach cle arly outperforms the GA(AC) approach, with a relative limited increase in CPU time. The relative improvement increases from 0.62% (1,000 orders) to 2.09% (16,000 orders). Improvements are most remarkable in the earliness cost, utilisation costs and production flow cost.

**LSA versus GA(AUF):** The LSA approach outperforms the GA(AUF) approach for problem instances with up to 4,000 orders, but performs worse for problem instances with 8,000 and 16,000 orders. Consequently, due to the heavy CPU-time burden, the LSA approach is not able to find high quality solutions (within the 100 generated schedules) that outperform the simple yet time efficient GA(AUF) approach.

| ***nri*** | | **1000** | **2000** | **4000** | **8000** | **16000** |
|---|---|---|---|---|---|---|
| **Avg. CPU** | **Random** | 0.04s | 0.08s | 0.18s | 0.47s | 1.70s |
| | **GA(AC)** | 0.07s | 0.13s | 0.30s | 0.92s | 2.46s |
| | **LSA** | 5.54s | 9.86s | 25.39s | 84.63s | 229.38s |
| | **GA(UAF)** | 0.17s | 0.32s | 0.88s | 2.54s | 6.60s |
| **Avg. *TC*** | **Random** | 448,739 | 802,156 | 1,510,731 | 2,928,450 | 5,721,389 |
| | **GA(AC)** | 362,651 | 645,390 | 1,209,376 | 2,358,950 | 4,604,384 |
| | **LSA** | 331,868 | 605,474 | 1,174,837 | 2,330,723 | 4,570,717 |
| | **GA(UAF)** | 360,407 | 638,359 | 1,192,511 | 2,322,494 | 4,507,826 |
| **Avg.** $C^A$ | **Random** | 126,557 | 253,846 | 506,713 | 1,009,435 | 2,024,264 |
| | **GA(AC)** | 43,110 | 86,780 | 173,116 | 345,530 | 691,843 |
| | **LSA** | 45,013 | 88,945 | 174,456 | 346,356 | 692,463 |
| | **GA(UAF)** | 53,739 | 103,660 | 205,559 | 410,145 | 820,593 |
| **Avg.** $C^L$ | **Random** | 23,127 | 30,735 | 53,344 | 90,705 | 166,188 |
| | **GA(AC)** | 17,837 | 25,501 | 46,462 | 82,581 | 155,232 |
| | **LSA** | 15,440 | 22,276 | 43,245 | 81,326 | 152,738 |
| | **GA(UAF)** | 24,974 | 34,900 | 61,035 | 112,022 | 209,644 |
| **Avg.** $C^E$ | **Random** | 29,339 | 64,943 | 132,080 | 277,023 | 565,306 |
| | **GA(AC)** | 27,164 | 60,428 | 119,746 | 255,733 | 514,755 |
| | **LSA** | 27,211 | 59,645 | 120,009 | 255,524 | 514,987 |
| | **GA(UAF)** | 23,192 | 49,362 | 99,788 | 198,063 | 399,730 |
| **Avg.** $C^U$ | **Random** | 129,451 | 212,094 | 374,385 | 701,847 | 1,368,577 |
| | **GA(AC)** | 118,611 | 193,099 | 340,400 | 634,104 | 1,248,311 |
| | **LSA** | 113,521 | 189,348 | 337,780 | 634,050 | 1,249,648 |
| | **GA(UAF)** | 105,974 | 171,961 | 301,392 | 568,778 | 1,110,039 |
| **Avg.** $C^F$ | **Random** | 140,264 | 240,536 | 444,208 | 849,440 | 1,597,052 |
| | **GA(AC)** | 155,930 | 279,582 | 529,652 | 1,041,002 | 1,994,244 |
| | **LSA** | 130,684 | 245,260 | 499,346 | 1,013,468 | 1,960,880 |
| | **GA(UAF)** | 152,527 | 278,477 | 525,736 | 1,033,487 | 1,967,819 |

**Table 2.** Comparison of different machine assignment approaches

## 4.3 Flexibility of our solution approach

In this section, we analyse the impact of all cost factors on the schedule quality and test the ability to modify the cost input parameters to obtain production schedules satisfying company specific objectives. In our experiment, we carefully change the cost input factors $e_i$, $l_i$, $u_j$, $b_{ln}^-$ and $b_{ln}^+$ and test their influence on the total quality of the schedule. More precisely, we multiply the original cost factor values (see section 4.1) by a factor 0.25, 0.5, 1, 2, 4 or 8 respectively, holding all other cost factors constant, and measure the resulting schedule quality by the following four performance measures:

**Average lateness:** measure the average lateness $\overline{C}^L$ of all orders over the complete scheduling horizon (in days).

**Average earliness:** measures the average earliness $\overline{C}^E$ of all orders over the complete scheduling horizon (in days).

**Average utilisation:** measures the average machine utilisation/capacity ratio as $\overline{C}^U = \sum_{i=1}^{nri}\sum_{j=1}^{nrj}\sum_{k=1}^{nrk}\frac{pt_{ij}x_{ijk}}{c_{jk}+\Delta c_{jk-1}}$.

**Average production flow:** measures the ratio of all production flow deviations and all pre-specified flow quantities as $\overline{C}^F = \frac{\sum_{l=1}^{nrl}\sum_{n=1}^{nrn}\sum_{m=1}^{nrm}|\underset{\sim}{F}_{nm}|}{\sum_{l=1}^{nrl}\sum_{n=1}^{nrn}\sum_{m=1}^{nrm}\underset{\sim}{F}_{nm}}$.

Note that this experiment has also been set up to validate the quality of our cost reduction estimates $CR_{ijk}^L$, $CR_{ijk}^E$, $CR_{ijk}^U$ and $CR_{ijk}^F$, since these estimates will influence the objective function of the knapsack problem and hence, the quality of the constructed schedule. Figure 5 displays the results for the problem instances with 8,000 orders and 10 different random machine assignments. All other problem instances or machine assignment procedures reveal similar results.

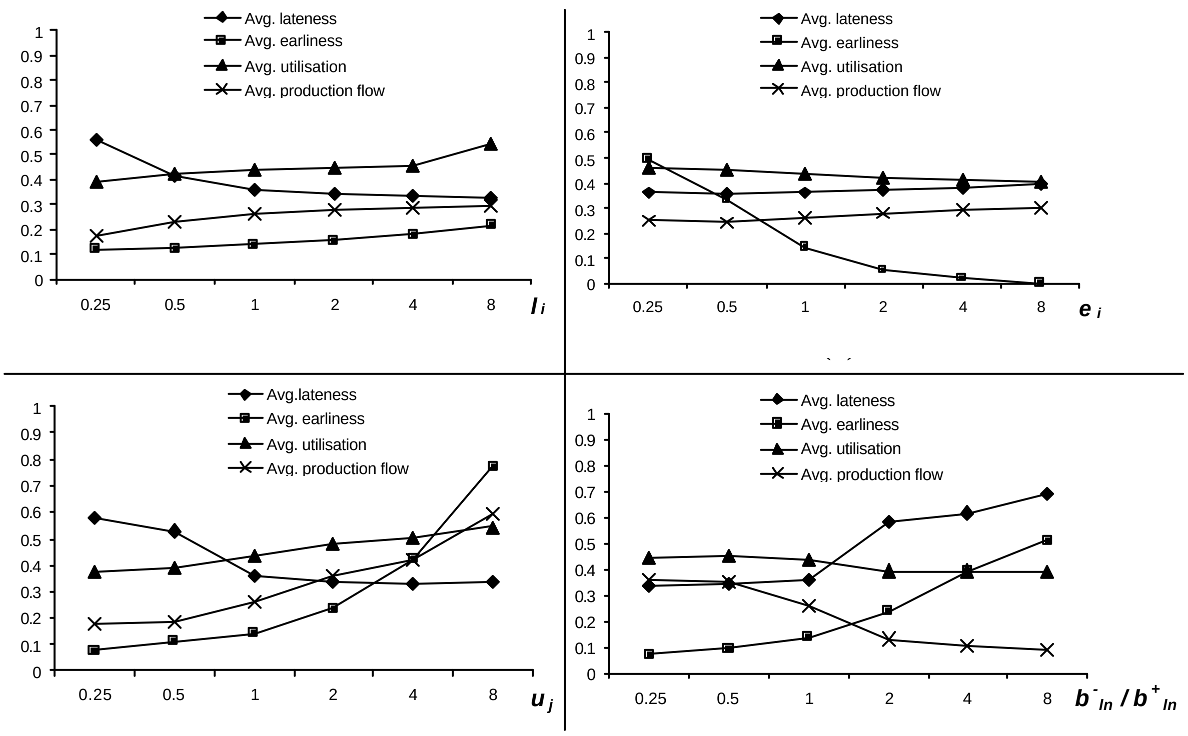

**Figure 5.** Influence of the input cost parameters on the performance measures

Figure 5 clearly shows that the schedule quality, expressed in terms of the four performance measures, clearly depends on the input cost factors. All figures show an improved performance for the corresponding input factor (e.g. figure 5(a) shows an improved average lateness for increasing values for the unit lateness cost $l_i$, figure 5(b) shows an improved average earliness for increasing value for the unit earliness cost $e_i$, etc…). This result illustrates the quality of our cost reduction estimates (see section 3.1) and the ability of the users of the scheduling algorithm to influence and define the schedule quality (the importance of each part of the multiple objective) by modifying the input cost factors. Note that the four cost factors of the multiple objective function not always show a trade-off. Figures 5(a) and 5(c) show that increasing importance of order lateness costs has a beneficial effect on the order lateness as well as on the average machine utilisation, and vice versa. Hence, both objectives are correlated as they stimulate earliest start schedules.

## 5 Conclusions

In this paper, we presented a finite capacity production scheduling algorithm for an integrated steel company located in Belgium. The algorithm takes various case-specific constraints into account and aims at the optimisation of multiple objectives.

The algorithm consists of two solution steps. A machine assignment step assigns each order to a unique machine for each production step. We have tested three different machine assignment methods, each taking various cost factors into account. The second step constructs a schedule where each operation of all orders is assigned to a particular day, given the assigned machines of the previous step. To determine which orders should be selected for scheduling at each machine during each day, we construct knapsack problems that take capacity constraints, precedence constraints and set-up constraints as well as the multiple objectives (lateness costs, earliness costs, utilisation costs and flow costs) into account.

We have tested our algorithm on a randomly generated dataset and have shown that our algorithm is flexible towards the user in terms of input cost parameters. Moreover, we show that a local search machine assignment (step 1) combined with an optimal knapsack solver (step 2) leads to the best performing results.

**Acknowledgements**

We would like to thank Krist Blomme, (member of SYMO (Systems and Models), specialist in line scheduling) and Alain Zegers (member of SYMO, specialist in line scheduling) of Arcelor Gent for drawing our attention to the challenging nature of the production scheduling project and for giving us the permission to use the data of the project. We are also grateful to Frederik Fransoo (member of SYMO, specialist in production scheduling) for providing us information about the data and for the numerous conversations in constructing new fictive data that reflects the real-life characteristics of the production scheduling environment.

## References

Dijkstra, E.W., 1959. A note on two problems in connexion with graphs, Numerische Mathematik., 1, 269-271.

Rahimifard, S. and Newman, S., 2000. A reactive multi-flow approach to the planning and control of flexible machining facilities, International Journal of Computer Integrated Manufacturing, 13 (4), 311-323.

Venkateswaran, J., Son, Y., Jones, A., 2004. Hierarchical production planning using a hybrid system dynamic-discrete event simulation architecture, Proceedings of the 2004 Winter Simulation Conference, 1094-1102.

Sum, C. and Hill, A., 1993. A new framework for manufacturing planning and control systems, Decision sciences, 24 (4), 739-760.

Pandey, P.C, Yenradee, P. and Archariyapruek, S., 2000. A finite capacity material requirements planning system, Production planning & control, 11 (2), 113-121.

Neureuther, B., Polak, G. and Sanders, N., 2004. A hierarchical production plan for a make-to-order steel fabrication plant, Production Planning & Control, 15 (3), 324-335.

Bitran, G. and Hax, A., 1981. Disaggregation and resource allocation using convex knapsack problems with bounded variables, Management Science, 27 (4), 431-441.

Adenso-Diaz, B. and Laguna, M., 1996. Modelling the load levelling problem in master production scheduling for MRP systems, International Journal of Production Research, 34 (2), 483-493.

Billington, P., McClain, J. and Joseph Thomas, L., 1983. Mathematical programming approaches to capacity-constrained MRP Systems: Review, formulation and problem reduction, Management Science, 29 (10), 1126-1141.

Taal, M. and Wortmann, J., 1997. Integrating MRP and finite capacity planning, Production Planning & Control, 8 (3), 245-254.

Rom, W., Icmeli Tukel, O., Muscatello, J., 2002. MRP in a job shop environment using a resource constrained project scheduling model, 30, 275-286.

Fry T.D., Cox J.F. and Blackstone J.H., 1992. An analysis and discussion of the optimised production technology software and its use, Production and Operations Management, 1, 229-242.

Segerstedt A., 1996. A capacity-constrained multi-level inventory and production control problem, International Journal of Production Economics, 45, 440-461.

Schmitt, T.G, Berry W.L. and Vollman T.E., 1984. An analysis of capacity planning procedures for a material requirements planning system, Decision Sciences, 15, 522-541.

Kolesar, J., 1967. A branch and bound algorithm for the knapsack problem. Management Science, 13 (9), 723-735.

Lee, H.S., Murthy, S.S., Haider, S.W., Morse, D.V., 1996. Primary production scheduling at steelmaking industries. IBM Journal of Research Development, 40 (2), 231-252.

Tang, L., Liu, J., Rong, A., Yang, Z., 2001. A review of planning and scheduling systems and methods for integrated steel production, European Journal of Operational Research, 133, 1-20.

Harjunkoski, I., Grossman, I.E., 2001. A decomposition approach for the scheduling of a steel plant production, Computers & Chemical Engineering, 25, 1647-1660.

Wiers, V., 2002. A case study on the integration of APS and ERP in a steel processing plant, Production planning & control, 13 (6), 552-560.

Okano, H., Davenport, 1.J., Trumbo, M., Reddy, C., Yoda, K., Amano, M., 2004. Finishing line scheduling in the steel industry, IBM journal of research and development, 48 (5), 811-830.

Hirschberg, D.S., Wong, C.K., 1976. A polynomial-time algorithm for the knapsack problem with two variables, Journal of the ACM, 23 (1), 147-154.