15th Triennial World Congress, Barcelona, Spain

SCHEDULING POLICY FOR ORDERS BASED ON EVENT-DRIVEN PERTURBATION ANALYSIS

Carsten Thierer, Uwe Kiencke

Institute of Industrial Information Technology
University of Karlsruhe, Hertzstraße 16, Building 06.35
D-76187 Karlsruhe, Germany
E-Mail: Carsten.Thierer@etec.uni-karlsruhe.de

Abstract: In the framework of designing a decentralised environment to decide upon schedule of orders and resource allocation, first an effective policy for one single scheduler has to be found that is suitable for extension towards multi-scheduler interaction. Thus the motivation to have a deterministic and effective approach. In this paper, an appropriate discrete, event-driven model is presented. The scheduling strategy is based on cost-effectiveness. Order delays and dynamic addition of orders to an existing schedule are taken into account. The technique of perturbation analysis is employed to determine an optimised schedule that minimises the cost function introduced.

Keywords: Discrete-event systems, Scheduling algorithms, Planning, Perturbation analysis, Time delay, Optimization problems, Nonlinear systems

1. INTRODUCTION

In this article, focus is laid on scheduling policy for orders of one single scheduler. The central problem to solve is the dynamic placement of one newly acquired order at run time, as this order represents an addition to an existing, static schedule of orders. While minimising the incremental costs involved, the decision remains to be made which already scheduled, future orders are to be delayed, and for which amount of time units. This decision shall as well offer the option to interrupt a currently processed order in favour of the newly arrived one, weighing the costs accordingly. Furthermore, for the application framework addressed, a suitable scheduling model should be extendable to a distributed decision environment that consists of multiple, interacting schedulers. In this context, "distributed" is characterised by a distributed information acquisition, processing and storage, (Kiencke, 1997). The aim of this framework is to realise a collection of communicating schedulers that acquire orders decentralised and cost-effectively for subsequent execution within their respective production sites, (Thierer *et al.*, 2001). Rather than coping with stochastic or heuristic models, instead the motivation for a deterministic approach arises, which advantageously is both simple and effective.

However, the scheduling problem itself inherits both stochastic and dynamic influence. Delays that are not predictable, for instance resource failures, are stochastic in length and instant of appearance. Arrival of a new, important order and the changes imposed on the static schedule show a dynamic behaviour. One elegant solution for such types of discrete event-driven problems, that allows deterministic calculation and optimisation, is the technique of perturbation analysis (Cassandras and Lafortune, 1999). The basic idea is to predict incremental changes to performance, i.e. incremental costs in the context of this paper, that are due to the change in some system parameters. Prediction is merely based on a sample path, i.e. the observation of a given system during nominal behaviour plus subsequent processing of available system knowledge and information. Rather than explicitly modelling the underlying stochastic, instead the

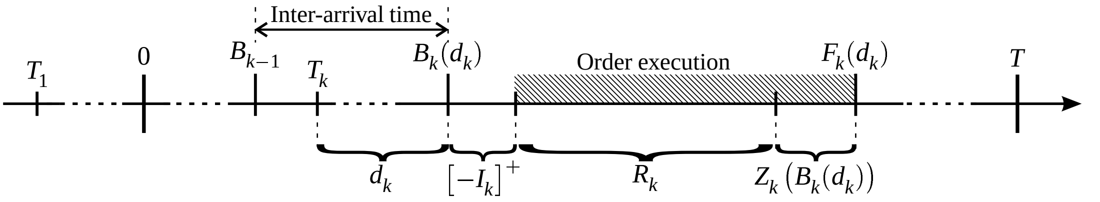

Fig. 1. Order model of single scheduler

parameter-dependent reaction of the system is estimated as deviation between perturbed sample path and observed sample path. Basic concept and initial idea for the model discussed in this paper relates to (Panayiotou and Cassandras, 2001) and stems from an altogether different application area, namely air traffic ground holding policies. In order to be adequately suited to the scheduling problem described, the appropriately adapted order model, required modifications and extensions, as well as options and parameters are presented in the sequel.

2. ORDER MODEL

For the purpose of this paper, one single scheduler is considered.

2.1 Events

The model introduced in this section is addressed as a discrete event system. The events involved are defined as **arrival event** corresponding to the time instant the order is made available to the scheduler and ready for being processed, as **begin event** corresponding to the time instant the execution process of order is scheduled to be started, and as **finish event** corresponding to the time instant the order is completed (thus the subsequently scheduled order may begin).

2.2 Notations and abbreviations

Notations used in this paper ($k \in \mathbb{N}$):

- $B_k(d_k)$ — scheduled begin event of O_k, if a delay of d_k relatively to T_k is assigned
- C_I, C_H — constant cost factors per unit time for interrupting or postponing a scheduled order (C_I) and for putting a newly acquired order on hold (C_H). Restriction (2) applies.
- d_k — time interval that delays O_k relatively to T_k. d_k is system parameter.
- $F_k(d_k)$ — finish event of O_k, if a delay of d_k relatively to T_k is assigned
- I_k — if positive, idle time interval between subsequently scheduled orders O_{k-1} and O_k
- O_k — k-th order in processing queue
- R_k — fixed run time interval of O_k
- T_k — fixed time of arrival event of O_k
- $Z_k(B_k(d_k))$ — additional run time interval of O_k, if $B_k(d_k)$ is provided. Z_k succeeds R_k.

The domain of all expressions comprises the set of real values $\mathbf{R}$, except where stated differently. The order model of one single scheduler, cp. Fig. 1, is derived from the standard queuing system, as it can be found in (Kleinrock, 1975).

As abbreviation used in the latter, define:

$$[x]^+ := \max\{0,x\} \geq 0 \qquad ,\forall x \in \mathbf{R} \qquad (1)$$

2.3 Order arrival T_k:

For a given, overall production time interval $[0,T[$ of length T, a set of $N > 0$ orders $O_1,\ldots,O_N$ is assumed to be a-priori known to the scheduler for future execution. The time of arrival, T_k, of order O_k is acquainted as the instant the order is available to the scheduler for processing. For all N statically fixed orders, T_k is negative in this model, whereas dynamically acquired orders at run time are distinguished by $T_k > 0$, (section 2.5).

2.4 Order delay d_k as system parameter:

Delays $d_k \geq 0$ between order arrival T_k and scheduled start of manufacturing B_k (section 2.5) are only due to congestion at a scheduler's manufacturing site. Possible causes may for instance be dynamic addition of a new order (section 2.5), maintenance or resource failure (stochastic in nature, section 2.7).

2.5 Scheduled start $B_k(d_k)$ and cost factors C_I,C_H:

At the time $T_k \geq 0$ of a new, dynamic order arrival, the scheduler may already be busy with a preceding order. With the option to immediately interrupt a currently processed order at T_k, it remains to be judged whether to wait for completion of the current order is preferable to interruption, depending on the respective costs incurred. Putting the current order on hold will reduce tolerances towards meeting order deadlines and double setting-up times of the manufacturing resources. Note that only immediate interruption is an option of this model. To adequately model this behaviour, a delay d_k for the start of new order O_k with $B_k(d_k) = T_k + d_k \geq 0$ can be assigned. Constant, order independent cost factors per unit time for interrupting

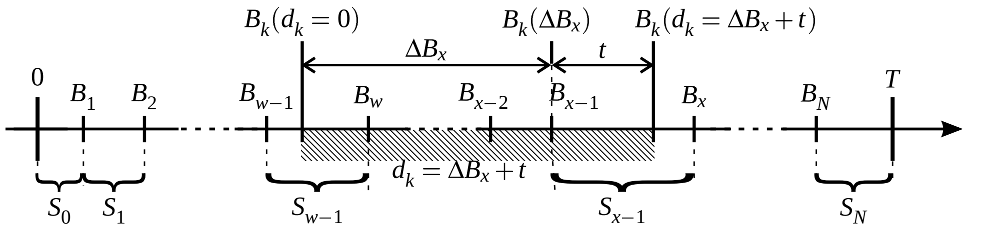

Fig. 2. Dynamically acquired order O_k

or postponing an already scheduled order, C_I, and for putting a newly acquired order on hold, C_H, will be applied:

$$C_I > C_H > 0 \tag{2}$$

Inequality (2) shall hold to ensure that it is not necessarily preferable to interrupt a currently proceeded task at once, depending on the dimension of C_I in relation to C_H.
Extending the concept of $B_k(d_k)$ to statically fixed orders alike, $T_k < 0$, a flexible method of scheduling is introduced, with d_k as system parameter. The absence of delay $d_k = 0$ represents an immediately scheduled order O_k at its arrival time, $B_k(0) = T_k$.

2.6 Fixed run time R_k:

The execution of order O_k takes at least a time interval of run time $R_k > 0$. R_k is fixed, a-priori known at arrival of order T_k and includes worst-case tolerances, based upon former experiences for this kind of order and/or customer.

2.7 Additional run time $Z_k(t)$ and completion $F_k(d_k)$:

As actual run time of an order may vary due to congestions, maintenance and delays d_k, one way to formalise this stochastic perturbation is to introduce additional run time $Z_k(t)$, succeeding R_k, which may be both time dependent (t) and order dependent (k). $Z_k(t)$ is provided by the scheduler's processing site as a feedback function that represents the actual condition of the manufacturing resources. The concept of additional run time is quite general. Allowing $Z_k(t)$ to accept negative values will in fact reduce the fixed, positive minimum run time R_k by its magnitude, $|\,Z_k\,|$, and thus consider earlier completion of order O_k as well, by a subsequently triggered re-scheduling run.
The time instant order O_k is finished, $F_k(d_k)$, corresponds to the end of additional run time, if this order has started with an assigned delay d_k. For calculation of $F_k(d_k)$, see equation (6).

2.8 Idle times I_k as precondition of order model:

Idle time intervals $I_k = B_k - F_{k-1}$ between the processing of two subsequent orders are fixed for all N orders, and, in combination with B_k of known orders, pre-planned to be positive ($I_k \geq 0$) if possible. For formal completeness of I_1, set $F_0 := 0$. To supply pre-planned idle times is a vital precondition to be ensured for the order model, as it allows for run time variations and delays to have less effect and to induce less costs.

2.9 Simplification:

For practical simplicity [1], this paper assumes that:

$$\begin{aligned} & Z_k(t+\Delta t) \approx Z_k(t) \qquad , \Delta t \text{ small} \\ & \Rightarrow \Delta Z_{k,k}(\Delta t) := Z_k(t+\Delta t) - Z_k(t) \approx 0 \end{aligned} \tag{3}$$

3. PERTURBATION ANALYSIS

At any given time, the set of all $N > 0$ fixed orders O_i, $i = 1, \ldots, N$, is assumed to have arrival times T_i as well as already scheduled, corresponding starting times B_i within time horizon $[0, T[$, cp. Fig. 2:

$$0 \leq B_1(d_1) < \ldots < B_N(d_N) < T \tag{4}$$

The choice of B_i shall respect the vital precondition to ensure idle times between (any) two orders if possible (section 2.8). As d_i is a-priori fixed, so will $B_i(d_i) = T_i + d_i$, thus one may simply write B_i to denote $B_i(d_i)$. Similarly, F_i denotes $F_i(d_i)$. The sequence of begin events B_i imposes $N+1$ time slots S_i onto time horizon $[0, T[$:

$$\begin{aligned} S_i := \left[B_i, B_{i+1}\right[\quad &, \; i = 0, 1, \ldots, N; \\ & B_0 := 0, B_{N+1} := T \end{aligned} \tag{5}$$

Each scheduler maintains a list of N a-priori fixed orders O_i, $i = 1, \ldots, N$. Each newly acquired order $O_k := O_{N+1}$ represents a dynamic addition to this static schedule, resulting in incremental costs due to additional delays imposed on the set of subsequent orders. One idea how to minimise the overall, future costs imposed is to assign a cost-effective delay d_k to order O_k. On basis of the provided and observed schedule thus a sample path has been created, allowing the order model to apply perturbation analysis

[1] For a small Δt, a resource failure (or its absence) at time t may still be of similar magnitude for the subsequent order at time $t + \Delta t$, regardless of order index k. However, assumption (3) is not necessary for the model to succeed. For higher accuracy, simply the administrative efforts of tracking all values of $\Delta Z_{l,i}(B_i)$ in (15) is required.

techniques as introduced in (Cassandras and Lafortune, 1999), with d_k as system parameter. The dynamics involved may best be represented in a recursive way, cp. Fig. 1,

$$F_k(d_k) = \max\left\{B_k(d_k), F_{k-1}(d_{k-1})\right\} + R_k + Z_k\left(B_k(d_k)\right) \quad (6)$$

in correspondence with the standard Lindley equation (Kleinrock, 1975). Perturbation analysis would interpret the max-operation in (6) to decompose a sample path into busy and idle periods. With $B_k(d_k)$ as result of the max-operation, the execution of order O_k will be scheduled after the previous order O_{k-1} has been finished: idle period $[F_{k-1}, B_k[$, $[I_k]^+ = I_k$). If the max-operation results in $F_{k-1}(d_{k-1})$, order O_{k-1} is not yet finished, while order O_k is assigned for start already: busy period $[B_k, F_{k-1}[$, $[I_k]^+ = 0$).

With a given static schedule B_i, $i = 1, \dots, N$, next a new arrival $T_k \geq 0$ of an order O_k is dynamically acquired. Note that $T_k < 0$ would imply a static pre-scheduling as a-priori knowledge rather than the need for a dynamic one at run time. The objective is to determine delay d_k such that the additional costs imposed are minimised. T_k shall fall into time slot $S_{w-1} = [B_{w-1}, B_w[$. So will $B_k(0) = T_k$, provided that there is no delay assigned to order O_k yet, $d_k = 0$. In general for $d_k > 0$, however, time slot $S_{x-1} = [B_{x-1}, B_x[\ni B_k(d_k)$, $x \geq w$ will be affected for $d_k > 0$ by:

$$B_k(d_k) = T_k + d_k = B_k(0) + d_k \geq B_k(0) = T_k \quad (7)$$

To simplify coming calculations, it is preferable to divide the yet to be determined delay d_k into two parts,

$$d_k =: \Delta B_x + t \geq 0 \quad (8)$$

with ΔB_x representing the execution delay as if begin event of O_k is scheduled for the same instant as B_{x-1}, and with t the remaining delay relatively to start of time slot S_{x-1}, cp. Fig. 2.

$$\Delta B_x := \left[B_{x-1} - B_k(0)\right]^+ \geq 0 \quad , \forall x \geq w \quad (9)$$

Note that in (9) the max-operation with zero is required for $x = w$. ΔB_x constrains t by the length of the respective slot:

$$\begin{array}{ll} 0 \leq t < B_x - B_{x-1} = \mid S_{x-1} \mid & , \Delta B_x > 0 \\ 0 \leq t < B_x - B_k(0) = B_w - T_k & , \Delta B_x = 0 \end{array} \quad (10)$$

It is important to realise that dynamic acquirement of new order O_k results in a begin event $B_k(d_k) > B_{x-1}$, in compliance with (4). Thus the schedule of finish events F_i, $i \leq x-1$, will not be affected by order O_k, except for F_{w-1} in case of immediate interruption. However, due to propagated delay explicitly caused by $F_k(d_k)$, $B_k(d_k)$ may increase execution times respectively instants of finish events for orders O_i, $i > x-1$ and for order O_{w-1} in case of interruption. Applying sample path technique, the additional delay $\Delta F_i(d_k)$ for order O_i may be expressed as

$$\Delta F_i(d_k) := \widetilde{F}_i(d_k) - F_i \geq 0 \quad , i = 1, \dots, N \quad (11)$$

F_i is interpreted as fixed and scheduled finish event of O_i (nominal sample path) and $\widetilde{F}_i(d_k)$ as expected finish event of O_i that depends on the assigned delay d_k of dynamically added order O_k as parameter (perturbed sample path). It is $\Delta F_i(d_k) = 0$ for $i = 1, \dots, (x-2)$, because prior to arrival of O_k it holds $B_{x-2} < F_{x-2} \leq B_{x-1} < B_k(d_k)$. The incremental, future cost explicitly caused by acceptance of O_k as a function of delay $d_k = \Delta B_x + t$ may now be formulated as a sum:

$$\begin{aligned} C_k(\Delta B_x, t) &= C_k(d_k = \Delta B_x + t) \\ &= C_H \cdot [\Delta B_x + t] + C_I \cdot \left[F_{x-1} - B_k(\Delta B_x + t)\right]^+ \\ &\quad + C_I \cdot \Delta F_{w-1}(t) + C_I \cdot \sum_{\{i > x-1\}} \Delta F_i(t) \end{aligned} \quad (12)$$

The first term corresponds to the cost of putting new order O_k on hold for delay interval d_k, cost factor C_H per time unit. The second term represents the cost for possibly postponing the drafted schedule of order O_k in favour of completion of currently executed order O_{x-1} (cost factor $C_I > C_H$ per time unit). This cost term is positive only if $F_{x-1} > B_k(\Delta B_x + t)$. Similarly, the third term

$$\Delta F_{w-1}(t) = -\left[F_{w-1} - B_k(t)\right]^+ \cdot \frac{t + C_H}{C_I} \quad (13)$$

solely appears if an immediate interruption of O_{w-1} occurred[2], as operation (1) limits non-negative t in accordance to (10). Depending on the ratio $\frac{C_H}{C_I} < 1$ between absolute cost factors C_H and C_I, (13) advocates in favour of ($C_H \to C_I$) respectively against immediate interruption ($C_H \to 0$). The last term in (12) comprises the sum of costs, factor C_I, incurred to all subsequent orders, (11). As all these orders B_i, $i > x-1$, begin after $B_{x-1} < B_k(d_k) < B_i$ has occurred, $\Delta F_i(d_k) = \Delta F_i(t)$ will depend on t only. Similarly, for the third term it holds $\Delta F_{w-1}(d_k) = \Delta F_{w-1}(t)$.

The objective now is to minimise $C_k(d_k) = C_k(\Delta B_x, t)$ by determining delay $d_k = \Delta B_x + t$. As cost function (12) is not differentiable, a way to solve this problem nevertheless is to observe that differentiability problems only relate to cases of event order changes. Rather than minimising d_k, for every possible ΔB_x a continuous t^* restricted by (10) is to be found such that $C_k(\Delta B_x, t^*) \leq C_k(\Delta B_x, t)$, $\forall t$ respecting (10). In a second step, the discrete value ΔB_x^* with minimal

[2] Note that only immediate interruption, i.e. $\Delta B_x = 0$ and $x = w$, is an option of this model. This is because in terms of costs, to postpone O_k is preferable to interrupt a subsequently scheduled order O_x, $x > w$. In the latter case, it holds $d_k \geq \Delta B_x > 0$, i.e. one is already delaying O_k by $d_k > 0$ anyhow, thus there is no point in interrupting a future order.

costs is to be chosen, $C_k(\Delta B_x^*, t^*) \leq C_k(\Delta B_x, t^*), \forall \Delta B_x$ possible. Another advantage of this scheme lies in the ability to easily respect absolute finishing deadline D_k of newly acquired order O_k by simply introducing an upper bounded range for possible intervals ΔB_x:

$$T_k + \Delta B_x \overset{!}{<} D_k - R_k - Z_k(T_k + \Delta B_x) < T \quad (14)$$

3.1 Determination of $\Delta F_i(t)$ and initial $\Delta F_x(t)$ in $C_k(\Delta B_x, t)$:

With x fix and $i > x$, it follows from (11), recursion (6) and $I_i = B_i - F_{i-1}$:

$$\begin{aligned}\Delta F_i(t) &= \max\{B_i, \widetilde{F}_{i-1}(t)\} + R_i + Z_i(B_i + t + R_k) \\ &\quad - \left[\max\{B_i, F_{i-1}\} + R_i + Z_i(B_i)\right] \\ &= \Delta Z_{i,i}(B_i) + \begin{cases} I_i > 0 : \max\{0, \Delta F_{i-1}(t) - I_i\} \\ I_i \leq 0 : \Delta F_{i-1}(t) \end{cases} \\ &= \Delta Z_{i,i}(B_i) + \left[\Delta F_{i-1}(t) - [I_i]^+\right]^+, \forall i > x \quad (15)\end{aligned}$$

(15) describes a recursive expression of how perturbation in a sample path will be propagated from its origin. Although tracking of additional run time Z_i is possible, for simplicity assumption (3) shall be applied, i.e. $\Delta Z_{i,i}(B_i) \approx 0$ and $\Delta t = t + R_k$ small. With x fix, the newly acquired order O_k causes an initial delay ΔF_x for order O_x that is propagated through succeeding orders. Employing idle times as a means to reduce overall delay of all subsequent orders, a distinction of idle time sequences is introduced. For any i with $I_i \leq 0$, it yields $[I_i]^+ = 0$, i.e. no idle time to spare between subsequent orders O_{i-1} and O_i. Thus recursion (15) provides $\Delta F_i(t) = [\Delta F_{i-1}(t)]^+ = \Delta F_{i-1}(t)$, as $\Delta F_j \geq 0$, $\forall j$ by definition (11). Consequently, order O_i will be postponed by the same delay as the previous one. In contrast, $I_i > 0$ will result in idle time $[I_i]^+ = I_i > 0$ to spare, thus recursively deliver $\Delta F_i(t) = [\Delta F_{i-1}(t) - I_i]^+$. Consider the ordered set of subsequent idle times $J := (I_{x+1}, \ldots, I_N)$. Let $(I_{(\mu_2)}, \ldots, I_{(\mu_M)}) \subseteq J$ be the ordered sub-set of J indicating positive idle times $I_{(\mu_j)} > 0$, $j = 2, \ldots, M$ only. Formally set $\mu_{M+1} := N + 1$ and $\mu_1 := x$. In case of absence of interruption, reset $I_x := 0$, see section 3.1. Applying this distinction, any μ_v and μ_{v+1} indicate subsequent positive idle time intervals to spare, $I_{\mu_v} > 0$ and $I_{\mu_{v+1}} > 0$, with $\mu_{v+1} - \mu_v$ describing the number of orders in between, i.e. length of sequence with no spare idle time [3] . One may now write as equivalent for $\sum_{\{i>x-1\}} \Delta F_i(t)$:

$$\sum_{v=1}^{M} (\mu_{v+1} - \mu_v) \cdot \left[\Delta F_x(t) - \sum_{j=1}^{v} I_{(\mu_j)}\right]^+ \quad (16)$$

With $I_x = B_x - F_{x-1}$ (section 2.8), it can be derived [4] when examining all distinct cases that either:

[3] Note that there is no intention to create an idle time for order O_k.
[4] Note that complete case studies and extensive derivations for (17), (18), (19) and (20) would exceed the space allotted.

$$\Delta F_x(t) = \\ = \begin{cases} I_x > 0 : \max\Big\{F_{x-1} + R_k + Z_k(T_k + t) - B_x, \\ \qquad [T_k + \Delta B_x + t + R_k + Z_k(T_k + t) - B_x]^+\Big\} \\ I_x \leq 0 : R_k + Z_k(T_k + t) \end{cases} \quad (17)$$

in case of absence of interruption as assumed in (17), $I_x := 0$, or

$$\begin{aligned}\Delta F_x(t) &= \max\left\{B_x, \widetilde{F}_{x-1}(t)\right\} - \max\left\{B_x, F_{x-1}\right\} \\ &= \left[(\widetilde{F}_{x-1}(t) - F_{x-1}) - [I_x]^+\right]^+ \\ &= \left[(R_k + R_k^*(t)) - [I_x]^+\right]^+ \end{aligned} \quad (18)$$

in case of interruption of order O_{w-1} in favour of O_k, where $x = w$ holds. Here, $R_k^*(t) \geq 0$ is introduced in addition to its fixed run time R_k to model setting-up times and perturbations Z_k caused by interrupting order O_k. Consequently, $\widetilde{F}_{x-1}(t) - F_{x-1} = (R_k + R_k^*(t)) > 0$ holds. Finally, the behaviour of (18) is identical to recursive expression (15), which is why in the case of interruption, one may simply set $\Delta F_x(t) := R_k + R_k^*(t)$ and include I_x unmodified into recursive calculation (16).

3.2 Optimal solution:

Finally, by distinction of all different cases, it can be shown $\forall t$ of any fixed slot S_{x-1} respecting (10) that for every corresponding ΔB_x, with exception of case interruption, $t = t^*(\Delta B_x)$ minimises $C_k(\Delta B_x, t)$, if:

$$\begin{aligned}t^*(\Delta B_x) = &-\min\left\{0, F_{x-1} - B_k(\Delta B_x)\right\} \\ &+ \min\left\{F_{x-1} - B_k(\Delta B_x), B_x - B_k(\Delta B_x)\right\}\end{aligned} \quad (19)$$

The exceptional case of interruption, where $\Delta B_x = 0$ and $x = w$, results in the appearance of additional third term (13) for costs (12) and a local minimum at:

$$t_I^* = \left[\frac{C_I + (F_{w-1} - B_k(0))}{2} - C_H\right]^+ \quad (20)$$

Operator (1) implicitly constrains t to interval $[B_k(0), F_{w-1}[$, where interruption is possible.

3.3 Algorithm to determine cost-optimal solution:

A-priori provide with B_i, (4), a static schedule of $N > 0$ orders, including calculated instants of finish events F_i, fixed run time R_i and idle time I_i. Then, based on the model, the algorithm in Fig. 3 decides on optimal delay d_k^* that minimises the additional, incremental costs $C_k(\Delta B_x, t)$, (12) for a new, dynamically acquired order O_k at the time of order arrival $T_k \geq 0$.

There are three stopping conditions. If $\Delta B_x \cdot C_H \geq C_{Int}$, costs for holding order O_k would exceed the costs that occurred if O_k would have started immediately at arrival T_k, even including optional interruption. In case of $\Delta B_x + T_k \geq D_k - R_k$, optionally assigned deadline D_k of order O_k is exceeded, (14). Stopping condition $x > N+1$ will take care if the last statically scheduled order is effected.

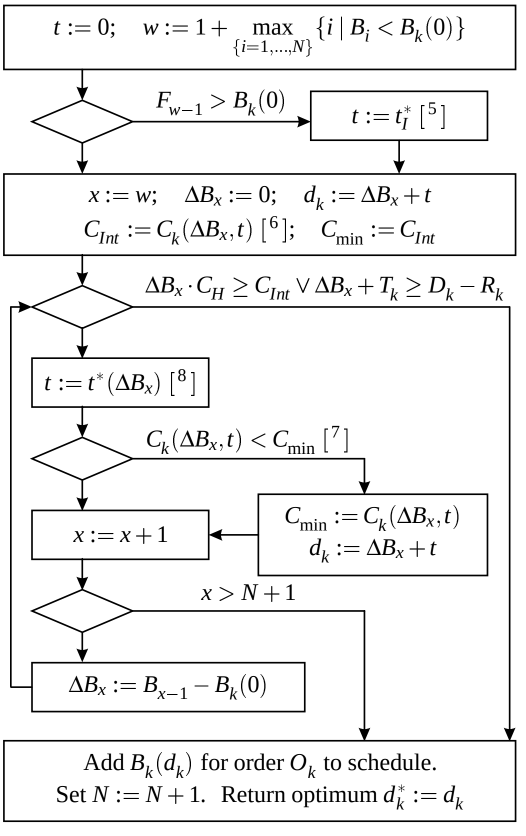

Fig. 3. Algorithm for cost-optimal delay d_k^*

To obtain an initial static schedule for N already given orders O_k consisting of B_i, F_i and I_i, execute:

(1) $B_1 = B_0 = F_0 = I_0 := 0$. $B_{N+1} = T$. $d_1 := B_1 - T_1$. $F_1 := F_1(d_1)$. [9]. ${I_2}^N := {I_1}^O$. $M := 1$.
(2) IF $\{M \geq N\}$ THEN { GOTO (11). }.
(3) CALL algorithm in Fig. 3 WITH PARAMETERS $\{M$ orders instead of N, $O_k := O_{M+1}$, $k := M+1$, $w := 2$, $\Delta B_x := B_1 - T_k\}$ AND OMIT THE LAST STEP.
(4) IF $\{T_k + d_k^* = B_1 = 0\}$ THEN $\{x := 2.\}$. ELSE $\{x := 1 + \max_{\{i=1,\dots,N\}} \{i \mid B_i < T_k + d_k^*.\}$.
(5) IF $\{T_k + d_k^* < F_{x-1}\}$ THEN $\{x := x-1.\}$.
(6) $\{B,F,I\}_i{}^N := \{B,F,I\}_i{}^O$, $i = 0, \dots, x-1$. ${I_x}^N := {I_x}^O$.
(7) IF $\{x \geq M+1\}$ THEN $\{{I_{M+2}}^N := {I_k}^O$. ${F_{M+1}}^N := {B_{M+1}}^N + R_k + Z_k({B_{M+1}}^N)$. ${B_{M+1}}^N := {F_M}^O + {I_{M+1}}^O$. GOTO (11). }.
(8) ${B_x}^N := {B_x}^O$. ${F_x}^N := {B_x}^N + R_k + Z_k({B_x}^N)$. ${I_x}^O := I_k$. $\Delta d_x := {F_x}^N - {B_x}^N + I_k$.
(9) $\{B,F\}_{i+1}{}^N := \{B,F\}_i{}^O + \Delta d_x$, ${I_{i+1}}^N := {I_i}^O$, $i = x, \dots, M$. ${I_{M+2}}^N := {I_{M+1}}^O$.
(10) $B_{M+2} := T$. $M := M+1$. GOTO (2).
(11) $B_{N+1} := T$. $d_i := B_i - T_i$, $i = 1, \dots, N$. END.

[5] Using (20) due to case interruption, where $\Delta B_x = 0$ and $x = w$.
[6] Using (12), (16) and either (17) for $F_{w-1} \leq B_k(0)$ or $\Delta F_x(t) := R_k + R_k^*(t)$ plus (13), for $F_{w-1} > B_k(0)$ (interrupt).
[7] See previous footnote.
[8] Using (19).
[9] Using (6).

Note that X^N refers to the set of actual values for X, derived from the set X^O of the previous iteration. The choice of appropriate, positive idle time values may be set off by a (generous) constant first, and subsequently be updated and refined. Similarly, $Z_k(B_k(d_k^*))$ may be either set off by a (order-specific) safety or maintenance constant, or simply set to the current value of $Z_k(t_0)$ at the instant t_0 a re-scheduling run is triggered. To request a re-scheduling is always recommended for each new order arrival.

4. CONCLUSIONS AND FUTURE WORK

We have introduced a cost functionality that is tolerant to possible order delays, stochastic in nature, and to the addition of a new, dynamically acquired order. An optionally set deadline for a new order can be ensured. Furthermore, the costs incurred by optional interruption of a currently processed order are included. Based on our event-driven model applicable for planning orders of one single scheduler, a deterministic solution on scheduling policy has been delivered. This solution proved to be optimal for minimising the given cost functionality, using perturbation analysis techniques. A clear algorithm summarised the steps for easy calculation of this optimal solution. To establish the initial, static schedule necessary for the model introduced, one method has been demonstrated that relied on repeated employment of the above algorithm.

In a next step, the interaction of multiple schedulers in a distributed decision environment towards optimisation in a global sense can be addressed, based on an extension of the deterministic and scalable scheme introduced in this paper.

REFERENCES

Cassandras, C. G. and S. Lafortune (1999). *Introduction to Discrete Event Systems*. Kluwer. Boston.

Kiencke, U. (1997). *Ereignisdiskrete Systeme, Modellierung und Steuerung verteilter Systeme*. Oldenbourg. Deutschland.

Kleinrock, L. (1975). *Queueing Systems. Volume I: Theory*. John Wiley and Sons. New York.

Panayiotou, C. G. and C. G. Cassandras (2001). A sample path approach for solving the ground-holding policy problem in air traffic control. *IEEE Transactions on Control System Technology* **9**(3), 510–523.

Thierer, C., P. Bort and U. Kiencke (2001). States in distributed systems based upon logical time. *Proceedings of Third IFAC Workshop on Advances in Automotive Control '01, Karlsruhe, Germany, March 28-30*, pp. 171–182.