Bi-level optimization for a dynamic multiobjective problem

Mikko Linnala, Elina Madetoja, Henri Ruotsalainen, Jari Hämäläinen

Bi-level optimization for a dynamic multiobjective problem

Journal:	*Engineering Optimization*
Manuscript ID:	GENO-2010-0256.R2
Manuscript Type:	Review
Date Submitted by the Author:	16-Feb-2011
Complete List of Authors:	Linnala, Mikko; University of Eastern Finland, Department of Applied Physics Madetoja, Elina; University of Eastern Finland, Department of Applied Physics Ruotsalainen, Henri; University of Eastern Finland, Department of Applied Physics Hämäläinen, Jari; Lappeenranta University of Technology, Centre of Computational Engineering and Integrated Design
Keywords:	Bi-level optimization, Dynamic multiobjective optimization, Dynamic process simulation

Note: The following files were submitted by the author for peer review, but cannot be converted to PDF. You must view these files (e.g. movies) online.

main.tex
intro.tex
theory.tex
example.tex
results.tex
ref.bib

Title: Bi-level optimization for a dynamic multiobjective problem

Journal: Engineering Optimization

Corresponding author: Mikko Linnala
Department of Applied Physics,
University of Eastern Finland,
P.O.Box 1627, FI-70211 Kuopio, Finland,
mikko.linnala@uef.fi.

Co-authors: Elina Madetoja
Department of Applied Physics,
University of Eastern Finland,
P.O.Box 1627, FI-70211 Kuopio, Finland,
elina.madetoja@iki.fi.

Henri Ruotsalainen
Department of Applied Physics,
University of Eastern Finland,
P.O.Box 1627, FI-70211 Kuopio, Finland,
henrimatias.ruotsalainen@gmail.com.

Jari Hämäläinen
Centre of Computational Engineering and Integrated Design,
Lappeenranta University of Technology,
P.O.Box 20, FI-53851 Lappeenranta, Finland,
jari.hamalainen@lut.fi.

Engineering Optimization
Vol. 00, No. 00, February 2011, 1–15

DRAFT

Bi-level optimization for a dynamic multiobjective problem

Mikko Linnala[a*], Elina Madetoja[b], Henri Ruotsalainen[c] and Jari Hämäläinen[d]

[a,b,c]*Department of Applied Physics, University of Eastern Finland, P.O.Box 1627, FI-70211 Kuopio, Finland,*
[a]*mikko.linnala@uef.fi*;
[b]*elina.madetoja@iki.fi*;
[c]*henrimatias.ruotsalainen@gmail.com*;
[d]*Centre of Computational Engineering and Integrated Design, Lappeenranta University of Technology, P.O.Box 20, FI-53851 Lappeenranta, Finland, jari.hamalainen@lut.fi.*

(*2nd (final) revision: 16 February 2011*)

In this paper, a bi-level optimization problem covering upper (design) and lower (operation) levels is defined and a solution procedure for bi-level optimization problems is presented. This is devised as a dynamic multiobjective optimization problem, i.e. the values of the control and state variables change over a predefined time horizon and several competing criteria are optimized simultaneously. Moreover, the interaction between the upper and lower levels is analysed. The benefits of bi-level dynamic multiobjective optimization are illustrated in detail by examining an industrial case in which the design of a paper mill (upper level) and the mill operation (lower level) are optimized at the same time. However, the problem definition and the solution procedure are not limited to any specific application but can be exploited in many different industrial areas.

Keywords: Bi-level optimization; dynamic multiobjective optimization; dynamic process simulation

*Corresponding author. Email: mikko.linnala@uef.fi

ISSN: 0305-215X print/ISSN 1029-0273 online

DOI: 10.1080/0305215YYxxxxxxxx
http://www.informaworld.com

1. Introduction

A bi-level optimization makes possible the simultaneous optimization of design and operation. A bi-level optimization problem has two levels where the upper level (e.g. process design) problem is the leader and the lower level (e.g. process operation) problem is the follower. Thus, the upper level solution affects the lower level solution and vice versa. For example, the optimization variables of the upper level are used as constants in the lower lever. If the objective function on one or both of the levels is vector-valued, the problem is called a bi-level multiobjective optimization problem (Eichfelder 2010). Bi-level multiobjective optimization makes it possible to conduct an efficient optimization of complicated practical applications, such as a papermaking process.

Bi-level optimization can be applied in many fields and for different kinds of problems. For example, in chemical engineering, bi-level optimization is exploited to optimize the design and control of processes (Mohideen *et al.* 1996b, Bansal *et al.* 2000a,b), in electricity markets, it is applied to strategic pricing (Fampa *et al.* 2008), and in supply chain problems, production-distribution interactions are studied (Calvete *et al.* 2011). Bi-level optimization methodology was developed by Mohideen *et al.* (1996b) who proposed an algorithm for solving integrated design and operation problems including dynamic mathematical models, uncertainty parameters, timevarying disturbances and robust stability criteria. Subsequently, Bansal *et al.* (2000b) described both simultaneous and sequential solution procedures for bi-level problems. In the sequential procedure, the process design and operation are optimized in turn until optimal design with optimal operation is achieved. Multiobjective bi-level problems have been studied by Fliege and Vicente (2006) as well as Deb and Sinha (2008) and Li *et al.* (2010) solved multiobjective bi-level problems by using evolutionary algorithms. Different cases and useful background to bi-level optimization were reviewed by Colson *et al.* (2005).

Practical optimization problems in industrial processes can usually be considered as being both dynamic and multiobjective. In addition, the solution methods may be based on the process models (Kameswaran and Biegler 2006) in which case they are referred to as model-based problems. As well as bi-level optimization, dynamic multiobjective optimization has been widely applied in chemical engineering. For example, Cervantes *et al.* (2002) have considered optimal control strategies of an industrial low-density polyethylene plant, and Barakat *et al.* (2008) have presented dynamic multiobjective optimization as applied in batch separation processes.

In the paper industry, although dynamic process modelling has been used for a long time (Barber and Scott 2007, Niemenmaa *et al.* 1998) the process optimization has mainly been simply single objective (Höfferl and Steinschorn 2009, Ropponen and Ritala 2010) or multiobjective with steady-state models (Madetoja and Tarvainen 2008, Hämäläinen *et al.* 2010). However, there are examples of dynamic multiobjective approaches being used in single level optimization (Linnala *et al.* 2009, 2010). Bi-level multiobjective optimization has already been exploited so that the design and operation of one sub-process (broke system) are optimized simultaneously (Ropponen *et al.* 2010). In this paper bi-level multiobjective optimization is applied for the broke system similarly to previous studies, but as far as it is known the process model used is more exact, realistic and larger than those used in the previous studies. In this way the results of the optimization are more reliable for implementation in real processes, which is one important goal of this kind of research.

In this study, dynamic and multiobjective properties were taken into account in both levels of a bi-level optimization problem unlike in previous studies of bi-level optimization.

That is because several of the practical optimization problems in the process industry are dynamic and multiobjective, and those problems have been not discussed extensively from a bi-level optimization point of view. Here, definition of the bi-level problem as well as the solution procedure is generalized, i.e. they are not limited to any specific application, algorithm or solver. Since the optimization problem is formulated in the bi-level multiobjective mode, it is possible to handle both long-term and short-term objectives simultaneously and efficiently. In the long-term, one can decrease capital expenditure in the process as well as raw material costs, and in the short-term one can assure optimal operation during different production tasks. In addition to handling different problems, a similar approach can be exploited for the design of both new processes and for the revision of existing processes.

The paper is organized as follows: the next section handles theoretical background and presents a solution procedure for a bi-level optimization problem. Then, there are numerical experiments and the results are presented in Section 3. Finally, conclusions sum up the results in Section 4.

2. Problem setting and solution procedure

Here a bi-level optimization problem in which the upper (design) and lower (operation) levels are taken into account will be defined. There are several papers on this topic (Deb and Sinha 2008, Mohideen *et al.* 1996b). However, certain aspects presented in this paper have not been considered in previously, where the emphasis of the research may have been different. Deb and Sinha (2008) have described an algorithm that uses a multiobjective evolutionary algorithm on both levels of a bi-level optimization problem. Thus, there are some similarities with this solution procedure. For instance, Deb and Sinha emphasize that the upper level control variables do not change in the lower level optimization. The same assumption is also used in this approach. In addition, both levels have separate objective functions and control variables. Moreover, Deb and Sinha highlighted one important issue: Pareto optimal solutions of the lower level optimization problem become feasible solutions to the upper level optimization problem. Pareto optimal solutions are mathematically equally good solutions of a multiobjective optimization problem (Sawaragi *et al.* 1985). However, this approach can use different optimization methods and algorithms whereas their algorithm was developed as a multiobjective evolutionary algorithm.

Similarly to this approach and Deb and Sinha's algorithm, also Mohideen *et al.* (1996b) attempted to find an optimal design on the upper level with feasible and efficient operation on the lower level. They proposed a framework for solving integrated design and control problems that included dynamic mathematical models similar to this approach. Moreover their framework also takes into account uncertainty parameters and time-varying disturbances as well as robust stability criteria. These two last features are not taken into account to the same extent in this approach, though this optimization can also cope with some time-varying disturbances (Linnala *et al.* 2010). The main difference between this approach and the algorithm of Mohideen *et al.* (1996b) is that their solution procedure can only manage a single objective function.

Since there are two important aspects in this approach; multiobjectivity and dynamics, next a dynamic optimization problem and its solution procedure are defined, and then a dynamic multiobjective optimization problem is studied.

2.1. *Dynamic optimization*

A dynamic optimization problem involves a dynamic process model where dynamic variable values change with time (Biegler and Grossmann 2004). Thus, its solution differs from the steady state problem. The dynamic optimization problem can be formulated as follows:

$$\begin{aligned} &\underset{\boldsymbol{u}(t)}{\text{optimize}}\ f(\boldsymbol{x}(t_f),\boldsymbol{u}(t_f),\boldsymbol{p},t_f) \\ &\text{subject to}\ \begin{cases} \boldsymbol{x}\in S_x \\ \boldsymbol{u}\in S_u \\ h\left(\frac{d\boldsymbol{x}(t)}{dt},\boldsymbol{x}(t),\boldsymbol{u}(t),\boldsymbol{\omega}(t),t\right)=0 \quad \text{for all } t\in[t_o,t_f], \end{cases} \end{aligned} \tag{2.1}$$

where f is an objective function, h is a system of the differential and algebraic equation constraints, which can also be divided into two separate systems (Biegler and Grossmann 2004, Cervantes and Biegler 2000), $\boldsymbol{x}$ denotes the differential and algebraic state vectors (these can also be presented separately), $\boldsymbol{u}$ is the vector of optimization variables, also known as a control vector, $\boldsymbol{p}$ is a steady parameter vector, and t_f is a length of time horizon. Sometimes the steady parameter vectors can be ignored, because they are constants from the optimization point of view. Since dynamic optimization has been studied for years, there are several different solution approaches for (2.1). Typically some of the solutions require at least partial discretization of the originally continuous variables (Biegler and Grossmann 2004, Grossmann and Biegler 2004).

In the case when several objectives need to be optimized simultaneously and objectives and/or variables change with time, the optimization problem is called a dynamic multiobjective optimization problem and it can be formulated as follows:

$$\begin{aligned} &\underset{\boldsymbol{u}(t)}{\text{optimize}}\ (f_1(\boldsymbol{x}(t_f),\boldsymbol{u}(t_f),\boldsymbol{p},t_f),\ldots,f_n(\boldsymbol{x}(t_f),\boldsymbol{u}(t_f),\boldsymbol{p},t_f)) \\ &\text{subject to}\ \begin{cases} \boldsymbol{x}\in S_x \\ \boldsymbol{u}\in S_u \\ h\left(\frac{d\boldsymbol{x}(t)}{dt},\boldsymbol{x}(t),\boldsymbol{u}(t),\boldsymbol{p}(t),t\right)=0 \quad \text{for all } t\in[t_o,t_f], \end{cases} \end{aligned} \tag{2.2}$$

where $\boldsymbol{f}=(f_1,\ldots,f_n)^T$ is a vector-valued objective function. In this case, the solution process is different compared to (2.1) because all the objective functions f_i, for all $i=1,\ldots,n$, need to be optimized at the same time. Thus, there does not typically exist a unique solution, but instead, there can be a set of solutions that are mathematically equally good. These solutions are called Pareto optimal or non-dominated or efficient solutions (Sawaragi *et al.* 1985).

2.2. *Bi-level optimization*

Based on the dynamic multiobjective optimization problem (2.2), the problem setting can be extended such that there are two separate levels in the optimization problem; the upper level and the lower level. Assumption is that both levels have multiple objectives

and the upper level includes the lower level optimization problem. In other words, optimization of the upper level requires that the lower level is optimized. The optimization problem considered on the upper level can be given as follows:

$$\begin{aligned} &\underset{\boldsymbol{a}}{\text{optimize}}\{F_1(\boldsymbol{a},\boldsymbol{x}(t_f),\boldsymbol{u}(t_f),\boldsymbol{\omega})\ldots,F_k(\boldsymbol{a},\boldsymbol{x}(t_f),\boldsymbol{u}(t_f),\boldsymbol{\omega})\} \\ &\text{subject to } \begin{cases} \boldsymbol{a}\in S_a \\ (2.4), \end{cases} \end{aligned} \tag{2.3}$$

where F_i, for all $i=1,\ldots,k$, are the upper level objective functions, $\boldsymbol{a}$ is a vector of the optimization variables on the upper level (design variables), $\boldsymbol{x}$ is a vector of the state variables, $\boldsymbol{u}$ is a vector of the optimization variables on the lower level (control variables), $\boldsymbol{\omega}$ is a vector of the operational tasks, t_f is the length of the optimization horizon and S_a is a feasible set of $\boldsymbol{a}$ defined by all the constraints including box constraints as well as linear and nonlinear equality and inequality constraints. In (2.3), the operation tasks ($\boldsymbol{\omega}$) are related to the lower level optimization problem and they typically present some assignment or modification of the system state. One should note that compared to some other formulations in (2.3), steady parameters are ignored.

The problem (2.3) becomes dynamic because the lower level optimization problem includes a system of transient differential and algebraic equations that is a dynamic system. The optimization problem on the lower level can be given as:

$$\begin{aligned} &\underset{\boldsymbol{u}}{\text{optimize}}\{f_1(\boldsymbol{x}(t_f),\boldsymbol{u}(t_f),\boldsymbol{\omega}(\boldsymbol{a},t)),\ldots,f_l(\boldsymbol{x}(t_f),\boldsymbol{u}(t_f),\boldsymbol{\omega}(\boldsymbol{a},t))\} \\ &\text{subject to } \begin{cases} \boldsymbol{x}\in S_x(\boldsymbol{a}) \\ \boldsymbol{u}\in S_u(\boldsymbol{a}) \\ h\left(\frac{d\boldsymbol{x}(t)}{dt},\boldsymbol{x}(t),\boldsymbol{u}(t),\boldsymbol{\omega}(\boldsymbol{a},t),t\right)=0 \quad \text{for all } t\in[t_o,t_f], \end{cases} \end{aligned} \tag{2.4}$$

where f_j for all $j=1\ldots,l$ are the lower level objective functions, $\boldsymbol{x}$, $\boldsymbol{a}$, $\boldsymbol{u}$ and $\boldsymbol{\omega}$ are the same as above, S_x and S_u are the feasible sets of $\boldsymbol{x}$ and $\boldsymbol{u}$, respectively, and h is the system of dynamic differential and algebraic equations, similar to (2.1) and (2.2).

As (2.3) and (2.4) show, in this problem formulation, the current optimization variable values on the upper level affect the lower level problem setting. They appear not only in the operational tasks but also in defining the feasible sets of the state variables and the optimization variables. On the other hand, the solution of the lower level optimization problem (optimal values of the optimization variables) affects the objective functions on the upper level. Thus, the optimization problems on the upper and lower levels have two-way coupling which is presented in Figure 1. Here assumption is that the objective functions and optimization variables are different between the levels, and the optimal variable values are denoted by *.

2.3. *Solution procedure*

Next a solution procedure for the bi-level optimization problem given in (2.3) will be presented. As described, this approach has a few similar features to simultaneous and sequential optimization procedures presented by Mohideen *et al.* (1996a) and the algorithm of Deb and Sinha (2008). In practice, this approach is able to handle dynamic

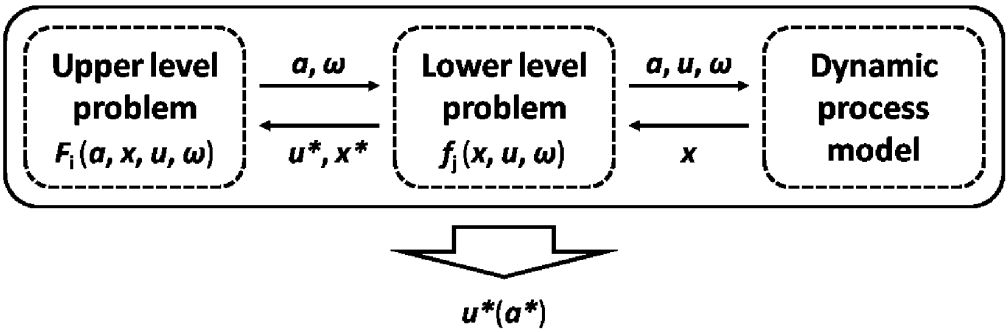


Figure 1. Interactions between the optimization levels and a dynamic process model.

and multiobjective optimization problems on both levels. The main emphasis is given to definition of an abstract solution procedure that can be applied and modified easily for different problems. In the following section, an example of how it is applied for solving a bi-level optimization problem from paper industry will be presented.

Algorithm 2.1 (Bi-level optimization)
Let the bi-level optimization problem be defined as in (2.3) and (2.4). Then the solution procedure for such a problem can be defined as follows.

1. Initialize the optimization method selected for the upper level optimization.
2. Define objective functions, a vector of optimization variables and constraints for the upper level optimization problem (2.3) (including the lower level optimization problem parameterized by the upper level optimization variables) and perform optimization as follows.
   Do until some stopping criterion is fulfilled:
   a) Let vector $\tilde{\boldsymbol{a}}$ contain the current values for the optimization variables on the upper level. Define the corresponding lower level optimization problem (2.4) at $\tilde{\boldsymbol{a}}$.
   b) Find the optimal state for lower level with the selected optimization method, i.e. perform the optimization based on Algorithm 2.2.
   c) Let $\boldsymbol{u}^*$ be the optimal solution of the lower level optimization problem and $\boldsymbol{x}^*$ the corresponding vector of state variables. In the multiobjective case $\boldsymbol{u}^*$ is selected from a set of Pareto optimal solutions using even human decision maker or some predefined information.
   d) Based on $\boldsymbol{u}^*$ and $\boldsymbol{x}^*$, evaluate objective functions $F_1, \ldots, F_k$ on the upper level and provide the objective function values to the optimization method.
   End Do
3. The optimal solutions for the bi-level optimization problem are $F_1, \ldots, F_k$ at $\boldsymbol{a}^*$ with the corresponding optimal lower level optimization solutions $f_1, \ldots, f_l$ at $\boldsymbol{u}^*$.

Algorithm 2.2 (Lower level optimization)
Let a lower level optimization problem be defined as in (2.4). Then the solution procedure is as follows:

1. Initialize the optimization method.
2. Define the objective functions, a vector of the optimization variables and constraints in (2.4) for the given parameters $\tilde{\boldsymbol{a}}$. Start optimization procedure as follows.
   Do until some stopping criterion is fulfilled:

a) Solve the dynamic process model with the current optimization variables $\tilde{\boldsymbol{u}}$.
b) Evaluate objectives $f_1, \ldots, f_l$ at $\tilde{\boldsymbol{u}}$ based on the state variables $\tilde{\boldsymbol{x}}$.

End Do

3. Save the optimal values of the optimization variables $\boldsymbol{u}^*$ and the corresponding state variable values $\boldsymbol{x}^*$ as well as $f_1, \ldots, f_l$ which are needed in Algorithm 2.1.

In this approach, the bi-level optimization problem is multiobjective in both optimization levels. If the objectives are conflicting, a set of mathematically equally good solutions (Pareto optimal solutions) is obtained on both levels. In the work of Deb and Sinha (2008) bi-level Pareto optimality was analysed and their algorithm could handle the Pareto optimal solutions produced on the lower level. However, in many situations only one solution has to be chosen as the final one. This choice can be done by a decision maker who is capable of comparing the Pareto optimal solutions based on her/his expert knowledge of the current problem. Sometimes computational time becomes unacceptably long and that precludes participation of the decision maker. Then some kind of predefined information with classical scalarizing functions can be used to choose the best solution during the bi-level optimization process (Miettinen 1999).

3. Numerical experiment

Here a numerical experiment where a bi-level optimization was applied to the process of papermaking is presented. To be precise, a broke system in which the rejected paper is being collected and re-circulated back into the process as a raw material is considered. Briefly, the aim was to optimize the capital costs and the operating costs simultaneously. The capital costs were described by the broke tower volumes that were minimized, and the operating costs were minimized by maximizing the broke dosage (in order to minimize the need for virgin raw material) and the net production. In order to highlight the special features of this problem, the application will be introduced.

3.1. *Dynamic multiobjective optimization related to papermaking*

In this experiment a dynamic process model of a supercalendered (SC) papermaking line was used. The model consists of a TMP (thermomechanical pulp) mill where the mechanical pulp is produced, an approach system where the raw materials are mixed and diluted, a broke system where the rejected paper is being collected, a paper machine where the paper web is formed and dried, and a reel where the ready paper is packaged for post processing. The dynamic process model was conducted with Apros software (VTT 2010) in co-operation with VTT Technical Research Centre of Finland.

The problem became bi-level one because the design and operation of the broke system had to be considered. The broke system had its own subsystems for a wet broke (paper rejected before drying having approximately 50 % moisture content) and a dry broke (ready paper being rejected). Hence, the broke system consisted of the elements presented in Figure 2.

When the broke is recycled back into the process, its properties differ from those of the virgin raw materials and therefore this affects the paper properties. For example, when the dosage of broke increases, the strength properties of the paper web decrease and furthermore, the risk of production failure, referred to as a break, increases significantly. Should a break occur, all the paper needs to be rejected and fed back to the broke system (that creates a need for larger broke tower volume). This phenomenon was modelled

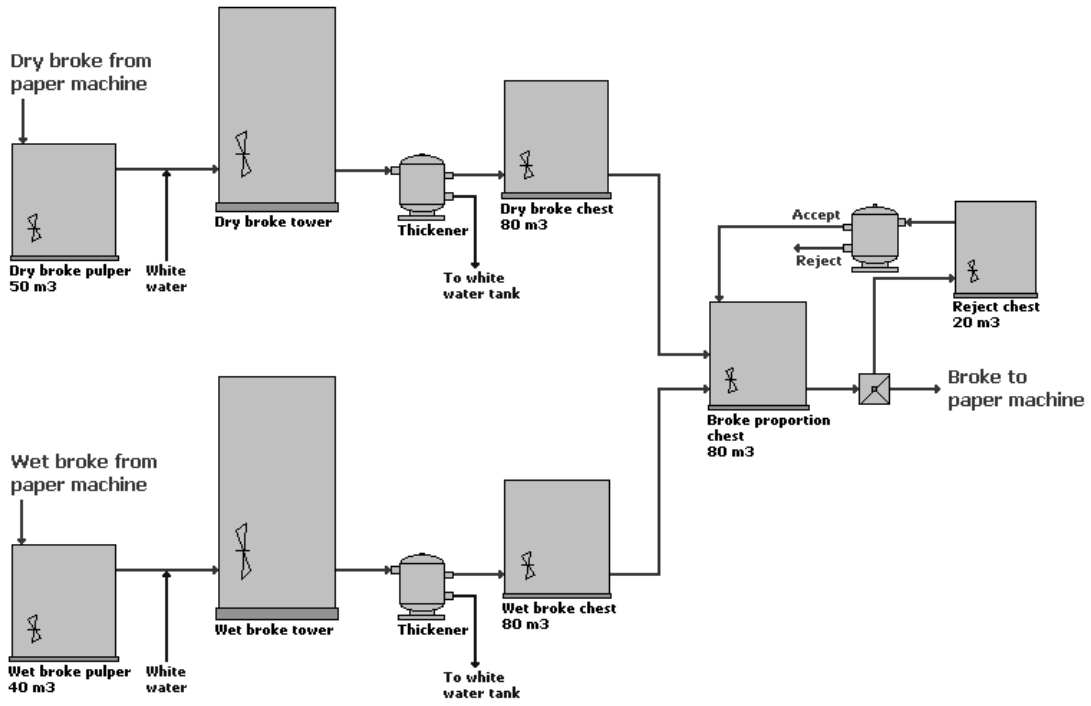


Figure 2. Broke system of the papermaking line modeled.

such that when the broke dosage increased, a larger part of the paper web was rejected, but no actual break occurred. This illustrated the growing risk of break caused by the increased broke dosage over the long-term. On the other hand, broke is a much cheaper raw material than virgin raw materials such as TMP or chemical pulp. Therefore the dosage of broke should be kept as high as possible. Hence, on the lower level (operation of the broke system) there were conflicting objectives: maximize both dosage of broke and net production. On the upper level (process design) investment costs (broke tower volumes) which depend on the operations of the lower level were tried to minimize. Overall, there were multiple objectives at both levels which had very strong interactions with each other.

3.2. *Optimization problem setting*

At the operational level, there were two continuous control variables; dosage of broke and the wet broke proportion of total broke dosaged. Instead, at the design level different tower volumes as control variables were considered and they were defined as integer variables in order to limit the search space. The objective functions optimized on the operational level consisted of production loss (f_1), variation of fill percentage of the dry broke tower (f_2), variation of fill percentage of the wet broke tower (f_3), and broke dosage (f_4). From these functions, f_1, f_2 and f_3 were minimized, and f_4 was maximized. The objective functions on the operational level were defined as follows:

$$f_j = \sum_{t=t_0}^{t_f} |x(t) - x_{target}|, \quad \text{for all } j = 1, \ldots, 4. \tag{3.1}$$

On the design level, the objective functions consisted of the fill percentages of wet broke (F_1) and dry broke (F_2) towers and both functions were minimized. The objective functions were defined as follows:

$$F_i = \max_{t\in[t_0,t_f]} |x(t) - x_{target}|, \quad \text{for all } i = 1, 2. \tag{3.2}$$

In this way, the tower volumes could be minimized, but runnability and stability were able to be maintained with buffer volume needed. This can be seen in the objective functions f_2, f_3, F_1 and F_2 which were based on the same state variables. At the operational level (f_2 and f_3), the process stability was the most important parameter whereas at the design level (F_1 and F_2) maximum values were the most important. Numerical values for the design, control and state variables are presented in Table 1. The upper and lower limits of the design and control variables were defined based on the expert knowledge. The initial values of the variables represented a stable process situation (reference point).

Table 1. Numerical values of the design, control and state variables. Lower and upper limits or step size were not needed for the state variables and target values were not needed for the control variables. Thus, they are marked with '-'.

Operational level	**Control variables**		**State variables**	
	Wet broke prop. [%]	Broke dos.* [%]	Prod. [t/h]	Fill-% [%]
Initial value	60	15	51.39	50
Lower limit	0	0	-	-
Upper limit	100	75	-	-
Target value	-	75	53.69	50
Design level	**Design variables**		**State variables**	
	$V_{\text{wet broke}}$ [m^3]	$V_{\text{dry broke}}$ [m^3]	Fill-% [%]	
Initial value	2000	4000	50	
Lower limit	250	250	-	
Upper limit	6000	6000	-	
Step size	250	250	-	
Target value	-	-	75	

* Broke dosage was both a control and a state variable. Its target value was used only in the objective function formulation.

The optimization problem on the operational level was multiobjective, and thus, a set of Pareto optimal solutions was obtained. Here only one solution was wanted to be brought to the design level. To achieve that goal, a scalarizing function was used to select a single solution. The method of global criteria was applied as a scalarizing function (Deb 2001):

$$S(\boldsymbol{f}, \boldsymbol{z}) = \left(\sum_{m=1}^{l} |f_m(\boldsymbol{u}) - z_m|^p \right)^{1/p}, \tag{3.3}$$

where $\boldsymbol{f} = (f_1, \ldots, f_l)^T$ is a vector of the objective functions, $\boldsymbol{z} = (z_1, \ldots, z_l)$ is the reference point defined close to the ideal point based on the expert knowledge and here p was 2. After the scalarizing, a solution which corresponded to the smallest value of the scalarizing function was brought to the design level.

Since the optimization problem on the operational level was dynamic, a receding horizon prediction principle (model predictive control) was applied (Rawlings 2000). In this example, the same dynamic process model was used for both prediction and simulation. Therefore two different time horizons were defined. The longer time horizon, T_{pred}=3000 s, was used for prediction of the control sequence, and the shorter time horizon, T_{sim}=600 s, was used to simulate one time stage forward with the first controls predicted. This loop was repeated until the end of total time horizon, 4800 s, was achieved ($T_{end} = T_8$). The principle of receding horizon prediction is presented in Figure 3, and an example of this has been presented previously (Linnala *et al.* 2010).

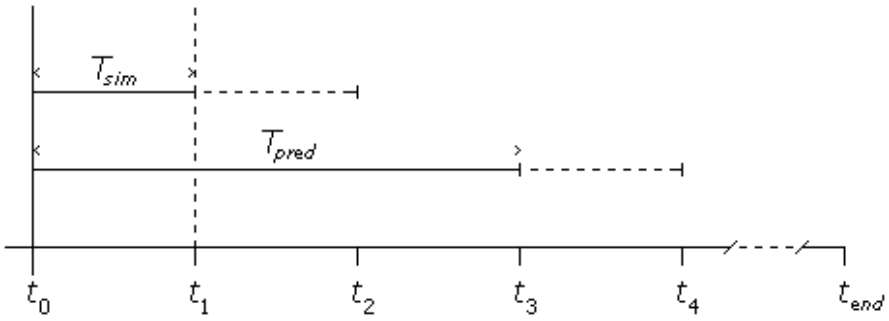


Figure 3. Receding horizon prediction principle.

In this example, the optimization problem on the both levels was solved by using the differential evolution algorithm (DE) implemented in MATLAB. DE belongs to the group of evolution strategies i.e. it simulates natural evolution. The simulation is based on generations and populations as in other evolution algorithms (Price *et al.* 2005, Lampinen 2002, Coello Coello 2000). DE algorithm can be formulated to solve multiobjective problems with linear and nonlinear equality and inequality constraints. The solution of the problem begins with the definition of initial vector population (target vectors) from the permitted search space. After the initialization, differential mutation produces a population of mutation vectors, such that, the scaled difference of two randomly sampled vectors is added to the third vector. The scale factor is used to control the rate at which the population evolves. After a mutation, DE performs a uniform crossover or discrete recombination. Subsequently, trial vectors are built by copying parameter values from two different vectors, i.e. each target vector is crossed with a corresponding mutant vector. The crossover probability is used to control the proportion of parameters copied from the mutant vector. The defined crossover probability is compared to the output of random number generator and if the probability is greater than the random number, then this parameter is copied from the mutant vector. Otherwise, the parameter value is copied from the target vector. Still, at least one parameter value is copied from the mutant vector in order to avoid duplication of the target vector. Finally, DE selects the vectors for the new population. If the trial vector is equally good or better from the objective function point of view, it will replace the target vector in the new generation. In the multiobjective case, the nondominated vector is selected to the new generation. In this way, the search moves toward the optimal solution in a step by step manner (Price *et al.* 2005).

Even if the same solver was used on both levels, some parameter values of DE differed. The mutation rate was 0.3 and the crossover rate was 0.6 on both levels. The number of generations and the population size were five at the design level. At the operational level, the number of generations was five also, but the population size was eight in order

to increase variation in the set of optimal solutions.

3.3. *Results*

After the optimization problem was solved, several different design candidates with the optimal operation were obtained. Since the design optimization problem was multiobjective and the operational level affected the design level, design candidates differed from each other. As an example, the objective function values of two chosen design candidates are presented in Table 2. Here the conflicts between the objectives can be appreciated: At the design level, if F_2 decreased, then F_1 increased and vice versa. At the operational level, conflicts were more complicated because several variables affected several objectives. Nonetheless, it could be considered as a conflict between the dry broke and wet broke line operations as can be seen in Table 2. If f_3 (wet broke tower variations) decreased, f_2 (dry broke tower variations) increased (Candidate 1) and vice versa, if f_2 (dry broke tower variations) decreased, f_3 (wet broke tower variations) increased (Candidate 2).

Table 2. The objective function values of two different solution candidates.

	Candidate 1	Candidate 2
f_1, sum of production loss [t/h]	1149	1200
f_2, sum of dry broke tower liquid level [%]	2587	3377
f_3, sum of wet broke tower liquid level [%]	1672	7489
f_4, sum of broke dosage [%]	408	385
F_1, maximum liquid level of wet broke tower [%]	25.00	24.93
F_2, maximum liquid level of dry broke tower [%]	24.98	25.00

Each optimal process design had its own optimal control sequence defined by the operational optimization over the time horizon $T_1 - T_8$. Table 3 presents the previous two design candidates including the design variable values (tower volumes) and control variable values (broke dosage and wet broke proportion of total broke).

Table 3. Design and control variable values of two design candidates.

Design variables [m^3]		Control variables [%]								
			T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8
Candidate 1, a_1	4000	u_1	15.0	7.6	2.3	18.7	3.9	38.6	8.5	27.5
a_2	1000	u_2	60.0	57.4	23.7	80.8	95.7	95.4	39.0	65.2
Candidate 2, a_1	1000	u_1	0.5	61.0	7.2	7.7	3.9	40.7	25.7	12.4
a_2	2750	u_2	37.5	58.5	38.5	38.4	50.4	42.2	46.3	2.8

a_1: Wet broke tower volume, a_2: Dry broke tower volume.
u_1: Broke dosage, u_2: Wet broke proportion of the total broke.

The optimization results can be visualized with the time series of different state variables used in the objective functions as presented in Figures 4 and 5. Here, five optimal solutions are shown in order to illustrate the variation between the different designs and operations. The dotted lines mark the target levels, i.e. the objective function values are better the closer the state variable values are to the target level. For the sake of clarification, design candidates 1 and 2 presented above are marked with arrows in Figure 4.

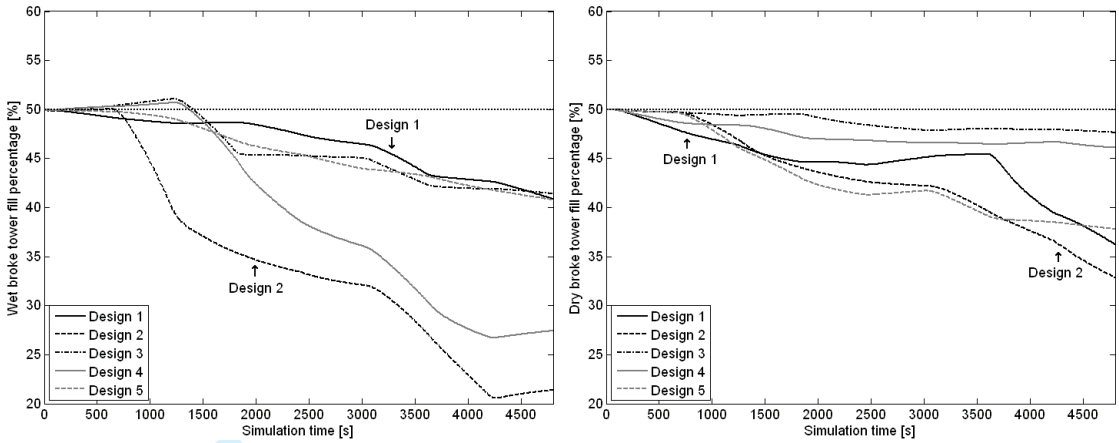


Figure 4. Fill percentages of wet broke (left) and dry broke (right) towers during the simulation. The dotted line represents the target value.

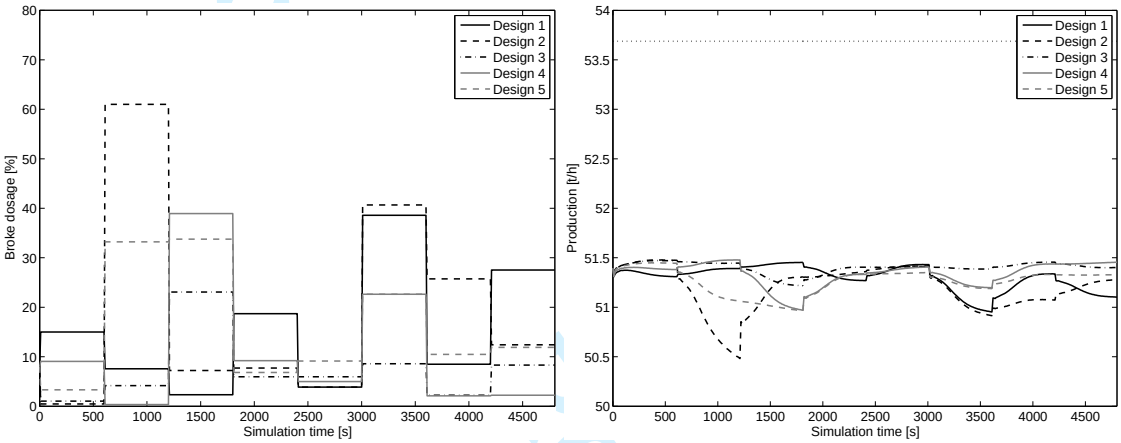


Figure 5. Broke dosage for each of the eight time stages (left) and production (right) during the simulation. The dotted line represents the target value.

Figure 6 presents variation in basis weight and filler content which are very important paper quality parameters. These parameters were not included in the objective functions but produce interesting additional information for the decision maker because the quality parameters should stay within the target interval (marked with dotted lines).

As the results show, there was variation between the solutions both in design and in operation. At the operational level, variations in the broke tower liquid levels (Figure 4) differed rather markedly between the design candidates. For example, designs 2 and 4 show that too strong operations lead to loss of the process stability. The same phenomenon can be seen in Figure 5 in which too high a broke dosage leads to production losses (design 2). In addition, some solutions, such as designs 2 and 4, could be discounted in practice based on the expert knowledge because the paper quality would not stay at the target interval, as can be seen in Figure 6. Since the paper quality was not included in the objective functions, all the solutions presented are equally good from the optimization point of view. Therefore, a decision maker is the best individual to compare the various solution candidates and choose the best one to be realized.

Above, only one control sequence for each design is presented. Nonetheless, depending on the method used at the operational level, different control sequences are also possible. Here, the scalarizing function selected the best controls for eight time stages, one after

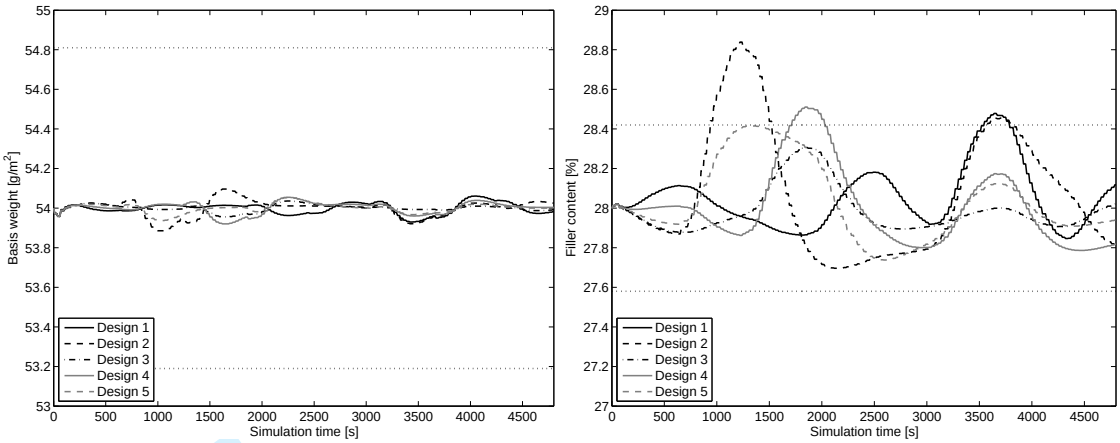


Figure 6. Basis weight (left) and filler content (right) during the simulation. The dotted lines represent the target intervals.

another. When the first controls were implemented, the next control possibilities were fixed, because the next stage depended on the previous stage. Note that if the scalarizing function or the value of reference point was changed, the optimal control sequence could be different. This is the challenge of dynamics in a bi-level case which has not been considered previously in the literature. Overall, this numerical experiment shows that dynamic and multiobjective bi-level optimization can be applied in practical optimization problems, but it requires expert knowledge to understand the relationships between different optimization levels, objectives and variables.

4. Conclusions

In this paper a real life optimization problem being dynamic, multiobjective and bi-level by nature was discussed. All these special features set requirements for the solution process as well as the problem formulation. Thus, the generalized formulation for a dynamic multiobjective bi-level optimization problem was defined, and a solution procedure for such problems was presented. Subsequently, the approach was illustrated with a real life example in which the process design and operation were optimized simultaneously in place of theoretical test case examination. The results show that this approach was successful and it could be implemented in practical optimization problems. Moreover, this complicated problem cannot be solved efficiently as a single level optimization problem. Here, the bi-level optimization problem was solved by using differential evolution algorithm, but other techniques can be implemented as well. It would be interesting to apply this approach with different optimization problems and algorithms. Furthermore, this generalized approach could be implemented in some industrial, even more complicated and challenging, optimization problems in order to demonstrate the real advantages of this approach.

Acknowledgements

The authors wish to thank M.Sc. Jouni Savolainen from VTT Technical Research Centre of Finland for their co-operation, and Finnish Forestcluster Ltd. and Finnish Funding Agency for Technology and Innovation (TEKES) for financial support (EffTech and

EffNet programs).

References

Bansal, V., *et al.*, 2000a. Simultaneous design and control optimisation under uncertainty. *Computers & Chemical Engineering*, 24 (3), 261–266.

Bansal, V., *et al.*, 2000b. The interactions of design and control: double-effect distillation. *Journal of Process Control*, 10 (4), 219–227.

Barakat, T., Fraga, E., and Sörensen, E., 2008. Multi-objective optimisation of batch separation processes. *Chemical Engineering and Processing: Process Intensification*, 47 (12), 2303–2314.

Barber, V. and Scott, G., 2007. Dynamic modeling of a paper machine, part I: programming and software development. *Tappi Journal*, 6 (1), 11–17.

Biegler, L. and Grossmann, I., 2004. Retrospective on optimization. *Computers & Chemical Engineering*, 28, 1169–1192.

Calvete, H., Gale, C., and Oliveros, M., 2011. Bilevel model for production-distribution planning solved using ant colony optimization. *Computers & Operations Research*, 38 (1), 320–327.

Cervantes, A. and Biegler, L., 2000. A stable elemental decomposition for dynamic process optimization. *Journal of Computational and Applied Mathematics*, 120 (12), 41–57.

Cervantes, A., *et al.*, 2002. Large-scale dynamic optimization for grade transitions in a low density polyethylene plant. *Computers and Chemical Engineering*, 26, 227–237.

Coello Coello, C., 2000. An updated survey of GA-based multiobjective optimization techniques. *ACM Computing Surveys*, 32 (2), 109–143.

Colson, B., Marcotte, B., and Savard, G., 2005. Bilevel optimization: A survey. *4OR: A Quarterly Journal of Operations Research*, 3 (2), 87–107.

Deb, K., 2001. *Multi-Objective Optimization using Evolutionary Algorithms*. Chichester: John Wiley & Sons, Ltd.

Deb, K. and Sinha, A., 2008. Solving bilevel multi-objective optimization problems using evolutionary algorithms. *KanGAL Report Number 2008005*.

Eichfelder, G., 2010. Multiobjective bilevel optimization. *Mathematical Programming*, 123 (2), 419–449.

Fampa, M., *et al.*, 2008. Bilevel optimization applied to strategic pricing in competitive electricity markets. *Computational Optimization and Applications*, 39 (2), 121–142.

Fliege, J. and Vicente, L., 2006. Multicriteria approach to bilevel optimization. *Journal of Optimization Theory and Applications*, 131 (2), 209–225.

Grossmann, I. and Biegler, L., 2004. Part II. Future perspective on optimization. *Computers & Chemical Engineering*, 28, 1193–1218.

Hämäläinen, J., Madetoja, E., and Ruotsalainen, H., 2010. Simulation-based optimization and decision support for conflicting objectives in papermaking. *Nordic Pulp and Paper Research Journal*, 25 (3), 405–410.

Höfferl, F. and Steinschorn, D., 2009. A dynamic programming extension to the steady state refinery-LP. *European Journal of Operational Research*, 197 (2), 465–474.

Kameswaran, S. and Biegler, L., 2006. Simultaneous dynamic optimization strategies: Recent advances and challenges. *Computers & Chemical Engineering*, 30 (10-12), 1560–75.

Lampinen, J., 2002. *In*: *Multi-Constrained Nonlinear Optimization by the Differential*

Evolution Algorithm., 305–316, London, England.

Li, M., Lin, D., and Wang, S., 2010. Solving a type of biobjective bilevel programming problem using NSGA-II. *Computers and Mathematics with Applications*, 59 (2), 706–715.

Linnala, M., *et al.*, 2009. Dynamic multiobjective optimization in papermaking process simulation. *In*: E. Madetoja, H. Niskanen and J. Hämäläinen, eds. *Proceedings of Papermaking Research Symposium 2009 (PRS2009)*, Kuopio, Finland.

Linnala, M., *et al.*, 2010. Dynamic simulation and optimization of an SC papermaking line – illustrated with case studies. *Nordic Pulp and Paper Research Journal*, 25 (2), 213–220.

Madetoja, E. and Tarvainen, P., 2008. Multiobjective process line optimization under uncertainty applied to papermaking. *Structural and Multidisciplinary Optimization*, 35, 461–472.

Miettinen, K., 1999. *Nonlinear Multiobjective Optimization*. Boston: Kluwer Academic Publishers.

Mohideen, M., Perkins, J., and Pistikopoulos, E., 1996a. A framework for process design and control. *In*: *Proceedings of UKACC International Conference on CONTROL96*, Exeter, United Kingdom.

Mohideen, M., Perkins, J., and Pistikopoulos, E., 1996b. Optimal design of dynamic systems under uncertainty. *AlChE Journal*, 42 (8), 2251–2272.

Niemenmaa, A., *et al.*, 1998. A multi-purpose tool for dynamic simulation of paper mills. *Simulation Practice and Theory*, 6 (3), 297–304.

Price, K., Storn, R., and Lampinen, J., 2005. *Differential Evolution - A Practical Approach to Global Optimization*. Berlin: Springer.

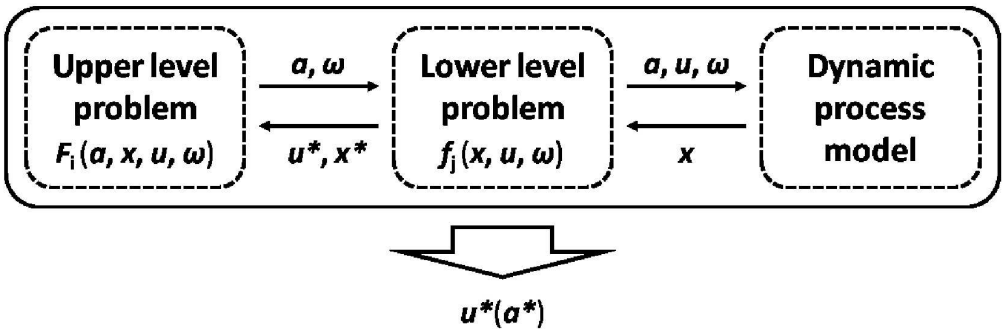
Rawlings, J., 2000. Tutorial Overview of Model Predictive Control. *Control Systems Magazine, IEEE*, 20 (3), 38–52.

Ropponen, A. and Ritala, R., 2010. Optimizing the design and operation of measurements for control: Dynamic optimization approach. *Measurement*, 43 (1), 9–20.

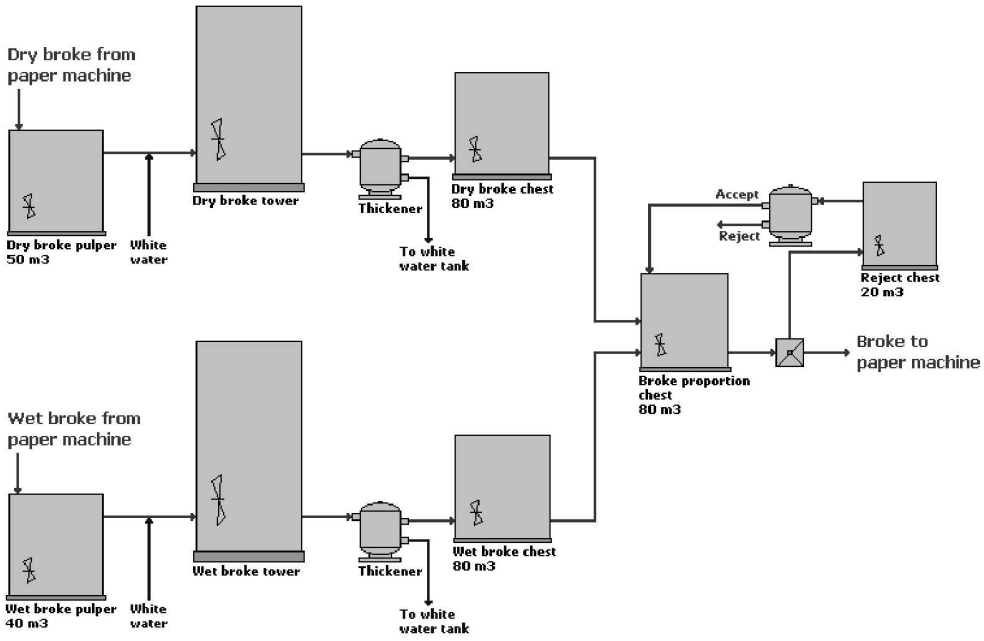
Ropponen, A., Ritala, R., and Pistikopoulos, E., 2010. Broke management optimization in design of paper production systems. *In*: S. Pierucci and G. Buzzi Ferraris, eds. *20th European Symposium on Computer Aided Process Engineering, ESCAPE20*, Naples, Italy.

Sawaragi, Y., Nakayama, H., and Tanino, T., 1985. *Theory of Multiobjective Optimization*. Orlando: Academic Press, Inc.

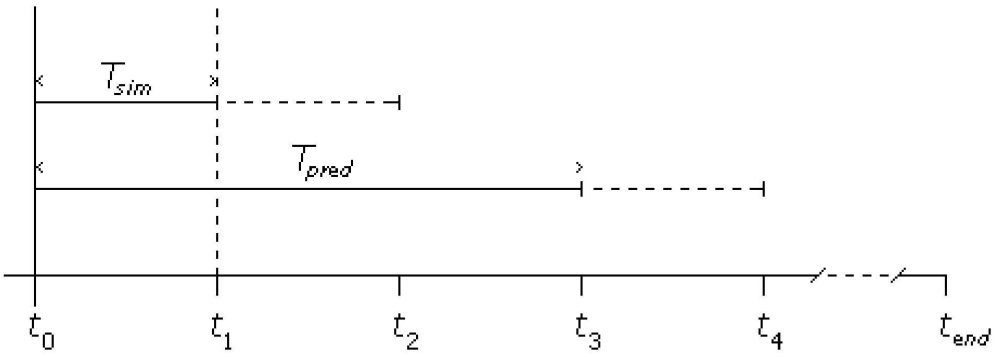
VTT, Chapter title. *Apros Overview [online]*, Technical Research Centre of Finland Available from: http://www.apros.fi [Accessed 27 October 2010], 2010. .



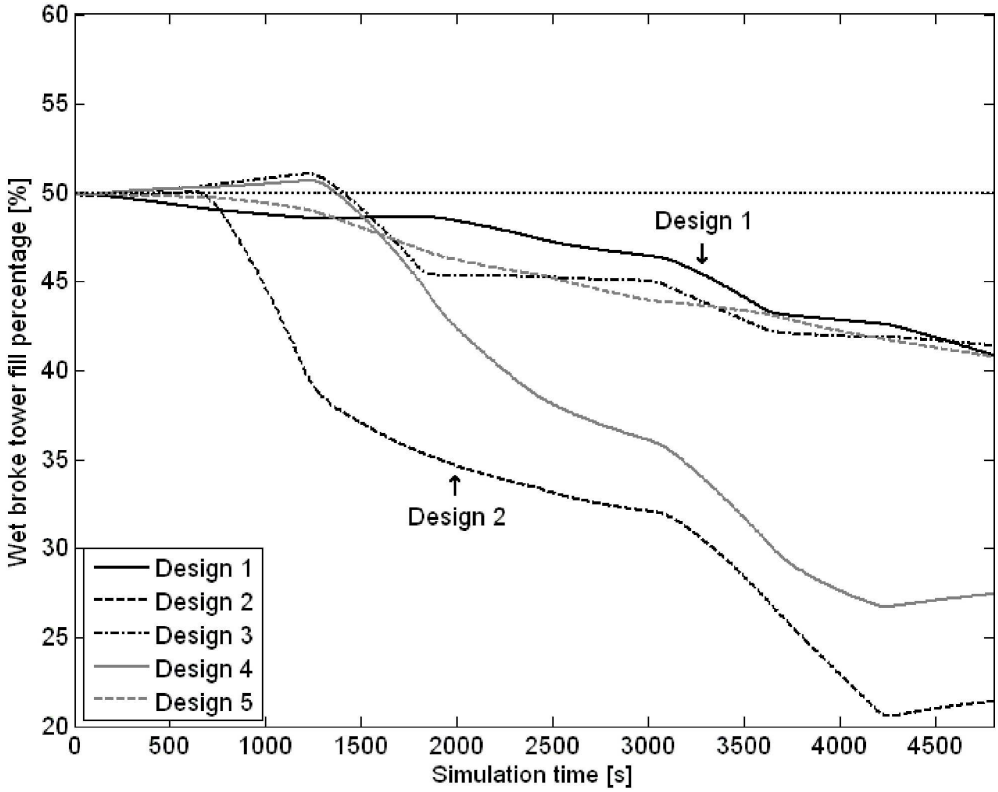
Interactions between the optimization levels and a dynamic process model.
419x137mm (600 x 600 DPI)



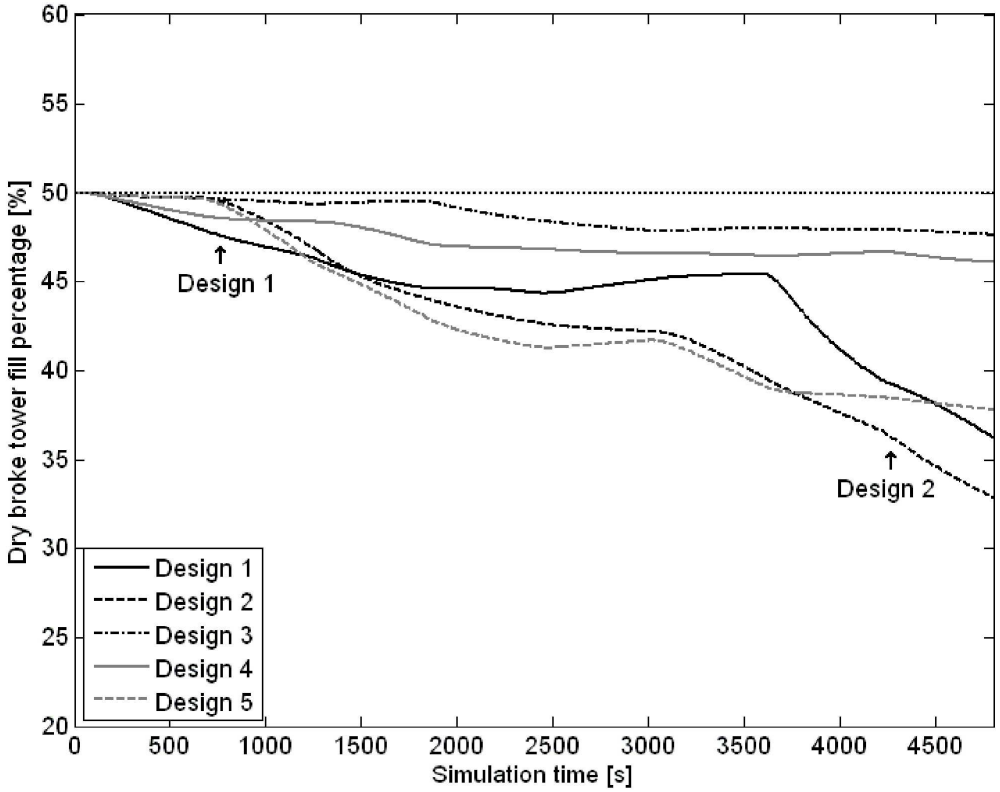
Broke system of the papermaking line modeled.
271x176mm (600 x 600 DPI)



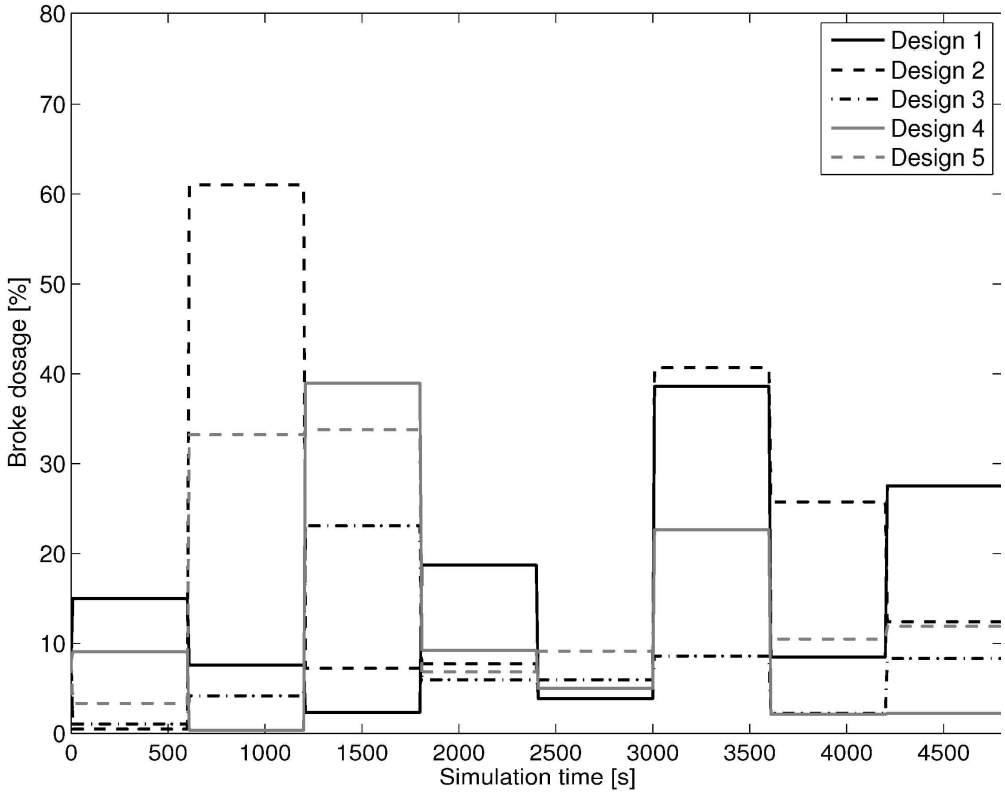
Receding horizon prediction principle.
162x59mm (600 x 600 DPI)



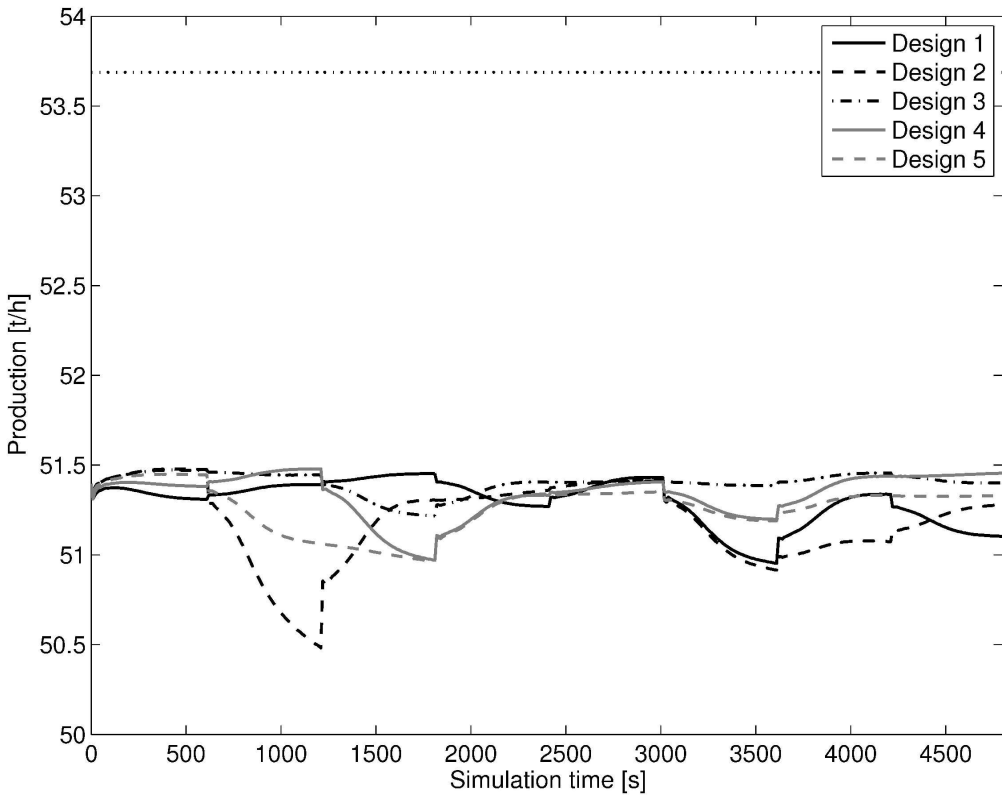
Fill percentages of wet broke (left) and dry broke (right) towers during the simulation. The dotted line represents the target value.
253x200mm (600 x 600 DPI)



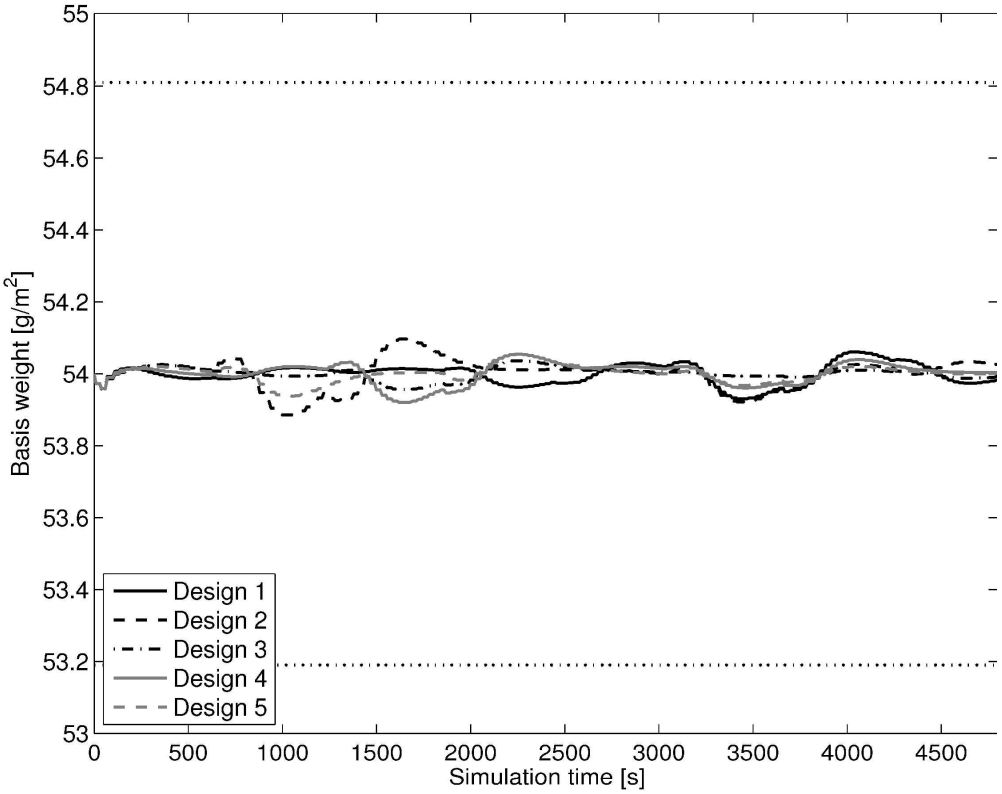
253x200mm (600 x 600 DPI)



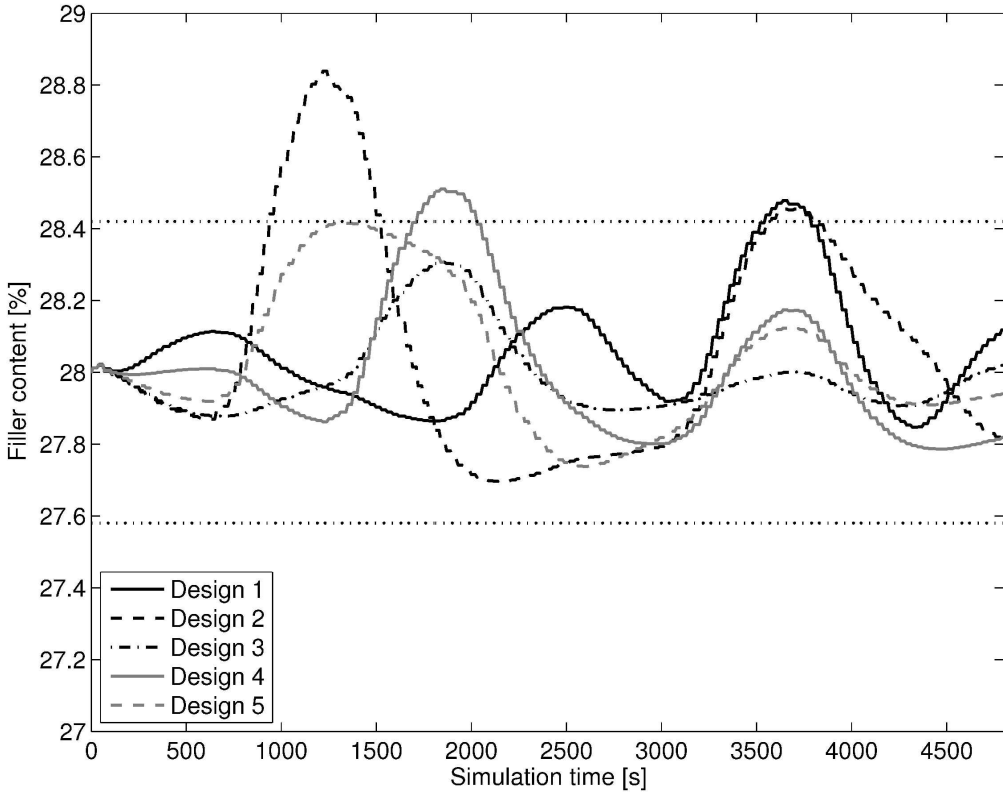
Broke dosage for each of the eight time stages (left) and production (right) during the simulation. The dotted line represents the target value.
190x150mm (600 x 600 DPI)



190x150mm (600 x 600 DPI)



Basis weight (left) and filler content (right) during the simulation. The dotted lines represent the target intervals.
190x150mm (600 x 600 DPI)



190x150mm (600 x 600 DPI)