We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

4,800
Open access books available

122,000
International authors and editors

135M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

CLARIVATE ANALYTICS BOOK CITATION INDEX INDEXED

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

13

An SQP and Branch-and-Bound Based Approach for Discrete Sizing of Analog Circuits

Michael Pehl and Helmut Graeb
Technische Universitaet Muenchen
Germany

1. Introduction

Analog circuits form an important part in integrated circuits and in particular in ASICs (Application Specific Integrated Circuits). However, due to the high complexity, design of this part has become a bottle-neck in the design flow (Gielen, 2007; Rutenbar et al., 2007). To overcome this problem and to guaranty that the analog part can be designed in reasonable time even for future technologies, methods supporting automatic design of analog circuits must be advanced.

This chapter focuses on sizing of analog circuits. It starts from the point where a topology is given. The task now is to choose design parameters, e.g., lengths and widths of transistors, such that certain properties of the circuit are fulfilled.

Current tools to solve the sizing task mostly treat it as a continuous optimization problem and use, e.g., certain gradient-based approaches to solve the problem in the continuous domain (Graeb, 2007). However, many design parameters are discrete in reality, e.g., transistor multipliers (i.e., the number of transistors connected in parallel), or must be discretized for some practical purposes, e.g., transistor lengths and widths which should match to a manufacturing grid. Furthermore, for some future technologies as, e.g., FinFETs (Knoblinger et al., 2005), the transistor parameters must fulfill certain geometrical properties, and accordingly have to be discrete.

Considering discrete parameters, it is not sufficient to treat the sizing task as a continuous optimization problem and rounding the result. This can be followed from mathematical theory, where it is shown that continuous optimization with sub-sequent rounding might not solve the original discrete optimization task (i.e., a optimization task that considers discrete and continuous parameters) and leads to a suboptimal result (Li & Sun, 2006; Nemhauser & Wolsey, 1988). This can be confirmed by experiments.

To solve discrete optimization problems, statistical and evolutionary approaches have been proposed (Alpaydin et al., 2003; Cao et al., 2000; Gielen et al., 1990; Ochotta et al., 1996; Phelps et al., 2000; Somani et al., 2007). However, for practical approaches these tools are usually more slowly in comparison to deterministic gradient-based tools if a good initial solution can be given for the task (what is normally true for analog sizing). Even if statistical and evolutionary approaches might be the first choice if a global search is necessary, for many cases deterministic gradient-based approaches are more suitable. Deterministic approaches for discrete sizing of analog circuits have barely been published till today (Pehl & Graeb, 2009; Pehl et al., 2008). In this chapter a new deterministic gradient-based approach is presented. It

consists of Sequential Quadratic Programming and Branch-and-Bound.
For the approach in this chapter, the problem is sub-divided into a non-linear program (NLP) and a discrete program (DP). Afterward, a discrete program is modeled by a discrete quadratic program (DQP) to speed up the algorithm.
Before the algorithm is presented, it is shown in Section 2 how the task of analog sizing can be formulated as a discrete minimization program. The task is said to be solved if any parameter set is found, where sizing constraints as well as performance specifications are fulfilled.
Introducing a relaxation of the parameters (i.e., all parameters are considered to be continuously scalable), a non-linear, but continuous sub-problem can be defined, called the relaxed program. To solve this NLP, in Section 3.1 of the chapter a sequential quadratic programming (SQP) algorithm is introduced.
Obviously, the result of the relaxed program is a point in the relaxed - i.e., continuous - domain. So, in Section 3.2 of the chapter a Branch-and-Bound approach is introduced to find a discrete solution to the sizing task.
The algorithm based on SQP and Branch-and-Bound can be used to solve the discrete sizing problem. However, to improve the run time of the approach, in Section 3.3 of the chapter a modification to speed up the algorithm is described. In the modification, the quadratic model of the objective function - which is computed in the SQP algorithm - is used to get a discrete quadratic model of the original sizing task. By solving the discrete quadratic program a discrete point can be found which gives an approximation for the obtainable discrete solution. This approximation can be used to cut non-promising parts of the Branch-and-Bound tree and to speed up the algorithm.
Experimental results in Section 4 show that in contrast to continuous optimization with subsequent rounding the presented approach is able to find a discrete feasible solution in each test case. Furthermore, it can be seen that the modification described in Section 3.3 decreases the run time of the algorithm significantly without reducing the result quality.
Section 5 concludes and gives an outlook to future research.

2. Problem formulation

2.1 Sizing task

In the analog sizing step appropriate values for the design parameters $\mathbf{d}$ of a given topology must be computed such that certain properties of the circuit are fulfilled. Typical design parameters are, e.g., lengths and widths of transistors, which were normally considered as continuous scalable in previous gradient-based approaches. However, in reality most parameters in the circuit sizing step are discrete, e.g., due to manufacturing grids, due to modern transistor types as FinFETs, or due to properties from the layout step.
For the approach presented in this chapter, the sizing task is formulated as a discrete optimization task, i.e., a sizing task considering scalable discrete and continuous parameters. For this purpose the vector of design parameters $\mathbf{d}$ can be subdivided into three parts corresponding to different parameter classes:

1. Continuous parameters $\mathbf{d}_c$ are used to model design parameters which do not require the consideration of any grid and which lie in an N_c-dimensional domain $\mathbb{D}^{N_c}$ that is bounded by any upper bound $\mathbf{d}_{c,U}$ and any lower bound $\mathbf{d}_{c,L}$

$$\mathbf{d}_c \in \mathbb{D}^{N_c} = \{\mathbf{d} \,|\mathbf{d}_{c,L} \leq \mathbf{d} \leq \mathbf{d}_{c,U}\} \tag{1}$$

2. Scalable discrete parameters $\mathbf{d}_d$ are used to model design parameters which can only lie on a – not necessarily uniform – N_d-dimensional grid. These parameters $\mathbf{d}_d$ are subset of a domain $\mathbb{D}^{N_d}$:

$$\mathbf{d}_d \in \mathbb{D}^{N_d} = \bigtimes_{k=1}^{N_d} \mathbb{D}_i \tag{2}$$

$\mathbb{D}_i$ is a set corresponding to the i-th discrete parameter d_i. Furthermore, $\mathbb{D}_i$ is ordered by a relation $<$ (Pehl & Graeb, 2009). Assuming n_i discrete parameter values for parameter d_i, the ordered set can be formulated as:

$$\begin{gathered} d_i \in \mathbb{D}_i := (\mathbb{D}, <) = \left\{ d_{i,1}, ..., d_{i,k}, ..., d_{i,n_i} \right\} \\ \bigvee_{1 \leq k < n_i} \quad d_{i,k} < d_{i,k+1} \end{gathered} \tag{3}$$

3. Non-scalable discrete parameters $\mathbf{d}_x$ can be used to consider design options which can not be expressed by a scalable parameter and must be enumerated instead, e.g., the exchange of different technologies. This class of parameters is non-numerical in many cases. One way to consider this class of parameters, which fits to the approach presented in this chapter, is to define binary surrogate parameter for each design option. Assuming n_i discrete design options $d_{i,1}, ..., d_{i,n_i}$ for a parameter d_i, i.e.,

$$d_i \in \mathbb{D}_i := \left\{ d_{i,1}, ..., d_{i,k}, ..., d_{i,n_i} \right\} \tag{4}$$

the values are collected in a vector $\mathbf{d}_{x,i}$:

$$\mathbf{d}_{x,i} = \left[d_{i,1}, ..., d_{i,k}, ..., d_{i,n_i} \right]^T \tag{5}$$

Additionally the vector $\mathbf{d}_{b,i}$ of surrogate design parameters is defined as:

$$\begin{gathered} \mathbf{d}_{b,i} = \left[d_{b,1}, ..., d_{b,k}, ..., d_{b,n_i} \right]^T \\ d_{b,k} = \begin{cases} 1; \text{ when option } d_i \text{ should be chosen} \\ 0; \text{ otherwise} \end{cases} \end{gathered} \tag{6}$$

Thus, the vector of surrogate design parameters can be mapped to the value of the corresponding non-scalable discrete parameter d_i by

$$d_i = \mathbf{d}_{b,i}^T \mathbf{d}_{x,i} \tag{7}$$

To avoid that different options are chosen for the same parameter, an additional constraint must be added to the optimization task defined below for each non-scalable discrete parameter:

$$\mathbf{d}_b^T \mathbf{d}_b = 1 \tag{8}$$

The set of all binary variables corresponding to options for non-scalable parameters is assigned as $\mathbb{D}^{N_b}$.

In this chapter only continuous and scalable discrete parameters are used. However, using the binary surrogate parameters defined above, the approach in Section 3 can be applied accordingly.
For continuous parameters $\mathbf{d}_c$, and scalable discrete parameters $\mathbf{d}_d$ the domain $\mathbb{D}^N$ of the design parameters $\mathbf{d}$ can be defined as:

$$\mathbf{d} \in \mathbb{D}^N = \mathbb{D}^{N_c} \times \mathbb{D}^{N_d} \tag{9}$$

Thus, vector $\mathbf{d}$ is composed by a continuous part $\mathbf{d}_c$ and a discrete part $\mathbf{d}_d$:

$$\mathbf{d} = \left[\mathbf{d}_d^T \ \mathbf{d}_c^T\right]^T \tag{10}$$

The task of choosing a parameter point, such that certain circuit properties are fulfilled, now is formulated as:

$$\min_{\mathbf{d}\in\mathbb{D}^N} \varphi(\mathbf{d}) \text{ s.t. } \mathbf{c}(\mathbf{d}) \geq 0 \tag{11}$$

wherein $\mathbf{c}(\mathbf{d})$ are sizing constraints, which ensure a reasonable sizing of the circuit (Graeb et al., 2001; Massier & Graeb, 2008). $\varphi(\mathbf{d})$ is the objective function, which maps a multi objective optimization task to a scalar minimization problem.
The objective function for analog sizing should support improvement of any circuit property when the specification for a certain performance is fulfilled as well as when the specification is violated. To build up such a function, an error $\varepsilon(\mathbf{d})$ for each performance $f_i(\mathbf{d})$ is defined, which is the normalized distance from the current performance value to the specification bound $f_{B,i}$ of the performance:

$$\varepsilon(\mathbf{d}) = \frac{f_i(\mathbf{d}) - f_{B,i}}{f_{N,i}} \tag{12}$$

$f_{N,i}$ is a normalization factor which ensures that the values for all performances are comparable. Without loss of generality it can be assumed that $f_{B,i}$ is a lower bound for the performance such that

$$\begin{aligned} &\varepsilon(\mathbf{d}) \geq 0 \text{ when specifications are fulfilled} \\ &\varepsilon(\mathbf{d}) < 0 \text{ when specifications are not fulfilled} \end{aligned} \tag{13}$$

This is illustrated in Figure 1. To support improvement of the performances when the specifications are violated as well as when the specifications are fulfilled, in this approach an exponential sum of the normalized errors for all N_f performances is used:

$$\varphi(\mathbf{d}) = \sum_{i=1}^{N_f} e^{-\varepsilon_i(\mathbf{d})} \tag{14}$$

Although the given formulation leads to a Pareto-optimal point, i.e. a solution where one performance can not be further improved without deteriorating another performance, we choose to stop the optimization problem – and consider the sizing task solved – as soon as a point is found which fulfills all specifications. Thus, the minimization is stopped as soon as a point is found with:

$$\bigvee_{i=1,\ldots,N_f} \varepsilon(\mathbf{d}) \geq 0 \tag{15}$$

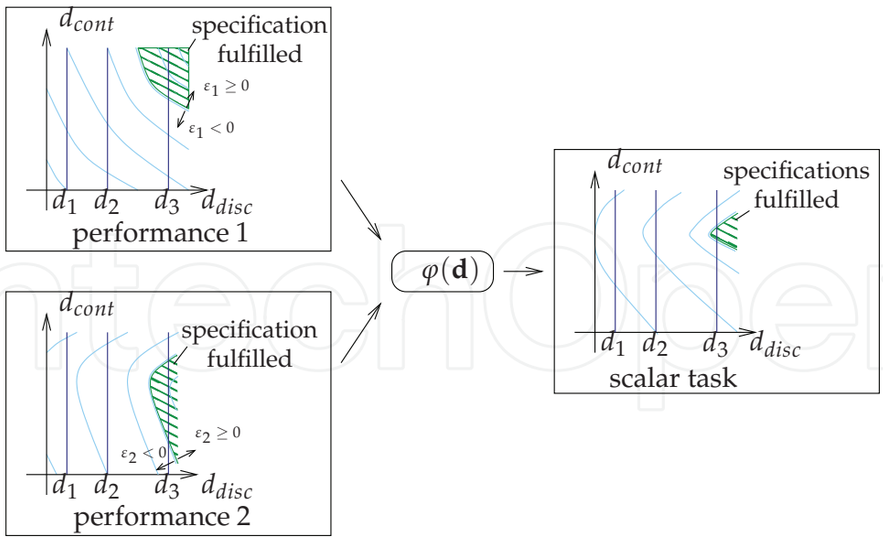


Fig. 1. Sizing task with two performances, one discrete parameter $d_{disc} \in \{d_1, d_2, d_3\}$, and one continuous parameter d_{cont} is mapped to a scalar optimization task by $\varphi(\mathbf{d})$.

2.2 Relaxation

To set up the relaxation of a discrete optimization task, the domain for each discrete parameter is replaced by a continuous domain. As the domain for the discrete parameters in (3) can be a ordered, the lower bound $d_{i,L}$ and upper bound $d_{i,U}$ for a discrete parameter d_i can be defined as the first and the last element of the ordered set $\mathbb{D}_i$

$$d_{i,L} := d_{i,1} \text{ and } d_{i,U} := d_{i,n_i} \tag{16}$$

For all discrete parameters, the lower bounds can be collected in a vector $\mathbf{d}_{d,L}$

$$\mathbf{d}_{d,L}^T = \left[... d_{i,1} ... \right] \tag{17}$$

and, respectively, the upper bounds can be collected in a vector $\mathbf{d}_{d,U}$

$$\mathbf{d}_{d,U}^T = \left[... d_{i,n_i} ... \right] \tag{18}$$

Thus, a vector of lower and upper bounds for discrete and continuous parameters can be built up:

$$\mathbf{d}_L^T = \left[\mathbf{d}_{d,L}^T \; \mathbf{d}_{c,L}^T \right]; \mathbf{d}_U^T = \left[\mathbf{d}_{d,U}^T \; \mathbf{d}_{c,U}^T \right] \tag{19}$$

For the relaxed optimization task all parameter points must be in the domain

$$\mathbb{D}_{rel}^N = \{ \mathbf{d} \, | \mathbf{d}_L \leq \mathbf{d} \leq \mathbf{d}_U \} \tag{20}$$

and the relaxed program can now be defined as:

$$\min_{\mathbf{d} \in \mathbb{D}_{rel}^N} \varphi(\mathbf{d}) \text{ s.t. } \mathbf{c}(\mathbf{d}) \geq 0 \tag{21}$$

The relaxation of a problem is illustrated in Figure 2.
Obviously, the discrete parameter set $\mathbb{D}^N$ is a subset of its relaxation $\mathbb{D}_{rel}^N$ and

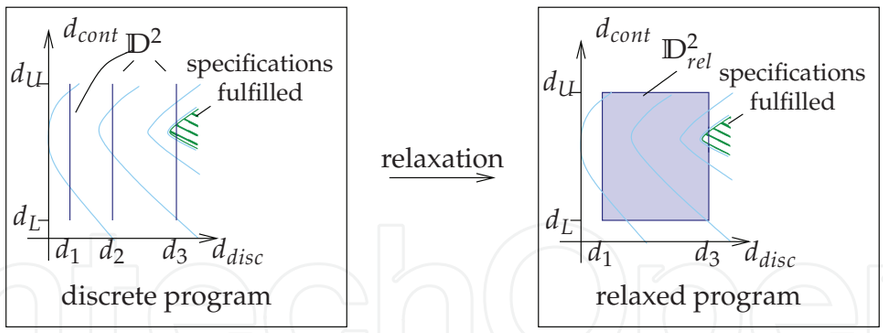


Fig. 2. Relaxation of the parameter domain $\mathbb{D}^2$ (see Figure 1) into the domain $\mathbb{D}^2_{rel}$

$$\mathbb{D}^N = \mathbb{D}^N \cap \mathbb{D}^N_{rel} \tag{22}$$

must be true.
The evaluation of the circuit performances in this approach is done by simulations. Thus, in the rest of this chapter it is assumed, that – even if parameters are discrete – simulation of the circuit is possible for each continuous point. In the future work the algorithm will be extended to use exclusively simulation results from discrete points.

3. Discrete sizing approach

3.1 Sequential Quadratic Programming

In the approach, presented in this paper, the relaxed optimization problem in (21) is solved by a Sequential Quadratic Programming (SQP) approach (e.g., (Nocedal & Wright, 1999)). SQP converts a constrained nonlinear optimization problem in the continuous domain into a sequence of unconstrained quadratic programming problems.
Using a vector of Lagrange multipliers $\boldsymbol{\lambda}$, the Lagrangian function of the problem can be given as

$$\mathcal{L}(\mathbf{d}, \boldsymbol{\lambda}) = \varphi(\mathbf{d}) - \boldsymbol{\lambda}^T \mathbf{c}(\mathbf{d}) \tag{23}$$

and the optimization problem (21) can be reformulated as the unconstrained optimization task:

$$\min_{\mathbf{d},\boldsymbol{\lambda}} \mathcal{L}(\mathbf{d}, \boldsymbol{\lambda}) \tag{24}$$

From the first-order necessary optimality conditions for unconstrained optimization, it follows that $\nabla_{\mathbf{d},\boldsymbol{\lambda}} \mathcal{L}(\mathbf{d}, \boldsymbol{\lambda}) = 0$ must hold true in the optimum point. Thus, to find the optimum of the Lagrangian function, Newtons method can be applied to the gradient $\nabla_{\mathbf{d},\boldsymbol{\lambda}} \mathcal{L}(\mathbf{d}, \boldsymbol{\lambda})$. Using $\Delta\mathbf{d}$ and $\Delta\boldsymbol{\lambda}$ for the change in parameters and μ as a linearisation index, the equation system which must be solved can be given as:

$$\begin{bmatrix} \nabla^2_{dd} \varphi(\mathbf{d}^{(\mu)} \boldsymbol{\lambda}^{(\mu)}) - \sum_i \lambda_{i,0} c_i \left(\mathbf{d}^{(\mu)} \boldsymbol{\lambda}^{(\mu)} \right) & -\nabla_{\mathbf{d}} \mathbf{c}^T \\ -\nabla_{\mathbf{d}} \mathbf{c} & 0 \end{bmatrix} \cdot \begin{bmatrix} \Delta\mathbf{d}^{(\mu+1)} \\ \Delta\boldsymbol{\lambda}^{(\mu+1)} \end{bmatrix} = \begin{bmatrix} -\nabla_d \mathcal{L}(\mathbf{d}^{(\mu)} \boldsymbol{\lambda}^{(\mu)}) \\ \mathbf{c}(\mathbf{d}^{(\mu)}) \end{bmatrix} \tag{25}$$

or with $\mathbf{H} = \nabla^2_{\mathbf{dd}} \varphi(\mathbf{d}^{(\mu)})$, $\mathbf{J} = \nabla_{\mathbf{d}} \mathbf{c}(\mathbf{d}^{(\mu)})$, $\mathbf{C}_i = \nabla^2_{\mathbf{dd}} c_i(\mathbf{d}^{(\mu)})$, and $\mathbf{g} = \nabla_{\mathbf{d}} \varphi(\mathbf{d}^{(\mu)})$.

$$\begin{bmatrix} \mathbf{H} - \sum_i \lambda_{i,0} \mathbf{C}_i & -\mathbf{J}^T \\ \mathbf{J} & 0 \end{bmatrix} \cdot \begin{bmatrix} \Delta\mathbf{d} \\ \Delta\boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} -\mathbf{g} \\ \mathbf{c}(\mathbf{d}_0) \end{bmatrix} \tag{26}$$

Algorithm 1: Branch and Bound($\mathbf{d}_{inc}$,$\mathbb{D}^N$, $\hat{\mathbb{D}}^N_{rel}$)

Input: incumbent $\mathbf{d}_{inc}$, parameter domain $\mathbb{D}^N$, relaxed domain $\hat{\mathbb{D}}^N_{rel}$

```
//compute minimum for given relaxed domain
```

$\mathbf{d}^* = \arg\min_{\mathbf{d}\in\hat{\mathbb{D}}^N_{rel}} \varphi(\mathbf{d})$ s.t.: $\mathbf{c}(\mathbf{d} \geq 0)$ 1

if *No feasible solution in* $\hat{\mathbb{D}}^N_{rel}$ **then** 2

```
    // pruning by infeasibility
```

    return $\mathbf{d}_{inc}$ 3

else if $\varphi(\mathbf{d}^*) \geq \varphi(\mathbf{d}_{inc})$ **then** 4

```
    // pruning by value dominance
```

    return $\mathbf{d}_{inc}$ 5

else if $\mathbf{d}^* \in \mathbb{D}^N$ *and* $\varphi(\mathbf{d}^*) < \varphi(\mathbf{d}_{inc})$ **then** 6

```
    // pruning by optimality
```

    return $\mathbf{d}^*$ 7

else 8

```
    // branching
```

    Choose parameter d_i **for branching** 9

    $\hat{\mathbb{D}}^N_{down} = \left\{\mathbf{d} \in \hat{\mathbb{D}}^N_{rel} \,\middle|\, d_i \leq \lfloor d_i^* \rfloor \right\}$ 10

    $\hat{\mathbb{D}}^N_{up} = \left\{\mathbf{d} \in \hat{\mathbb{D}}^N_{rel} \,\middle|\, d_i \geq \lceil d_i^* \rceil \right\}$ 11

    $\mathbf{d}_{inc}$ =**Branch and Bound(**$\mathbf{d}_{inc}$**,** $\mathbb{D}^N$**,**$\hat{\mathbb{D}}^N_{down}$**)** 12

    $\mathbf{d}_{inc}$ =**Branch and Bound(**$\mathbf{d}_{inc}$**,** $\mathbb{D}^N$**,** $\hat{\mathbb{D}}^N_{up}$**)** 13

    return $\mathbf{d}_{inc}$ 14

end 15

If the second derivative of the Lagrangian function is convex, the optimization problem in (24) can be solved by iteratively solving the equation system in (26). The result describes the direction from the current point to the minimum of the quadratic model of the objective function subject to the constraints. In the SQP approach a model of the matrix is usually built up iteratively. This can be realized by different approaches, e.g., BFGS (Broyden Fletcher Goldfarb Shanno) update formula, which is used in this approach.
After computing the direction by solving (24), a step size is computed at the original relaxed program using line search. In this approach a Wolfe Powell step size algorithm is used.
The solution which is computed by SQP on the relaxed program is obviously no discrete feasible point in general. In the next section of this chapter a Branch-and-Bound method is described, which can be used to find a discrete solution for the original sizing task based on the solution of the relaxed problem.

3.2 Branch and Bound

Branch and Bound (e.g., (Nemhauser & Wolsey, 1988)) is one of the most popular approaches in discrete optimization. In the form which is used in this work, it decomposes the discrete optimization task in a sequence of relaxed optimization tasks which are nonlinear but can be solved in the continuous domain. A description of the recursive method is given in Algorithm 1 and in the following.
The algorithm is primarily based on two principles:

1. As the domain of discrete points is a subset of its continuous relaxation (22), the optimum of the relaxed program is better than or equal to the continuous solution. For the minimization in (11) and with (21):

$$\left(\min_{\mathbf{d} \in \mathbb{D}_{rel}^N} \varphi(\mathbf{d}) \text{ s.t. } \mathbf{c}(\mathbf{d}) \geq 0 \right) \leq \left(\min_{\mathbf{d} \in \mathbb{D}^N} \varphi(\mathbf{d}) \text{ s.t. } \mathbf{c}(\mathbf{d}) \geq 0 \right) \tag{27}$$

   Consequently, the minimum of the relaxed program is a lower bound for the discrete minimum.
2. Each discrete point with objective function value better than the best discrete point so far, is an upper bound for the discrete solution of the optimization task.

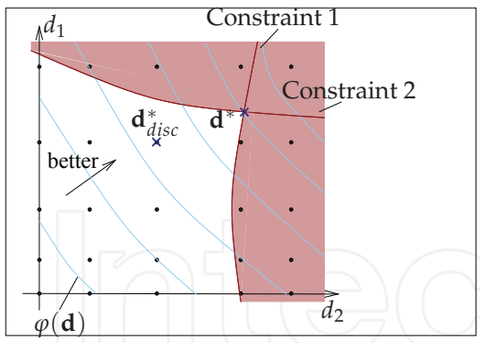
Initially in each recursion the relaxed optimization task is solved in the current relaxed domain $\hat{\mathbb{D}}_{rel}^N$ (Algorithm 1, line 1 and Figure 3(a)). In the approach presented in this chapter SQP is used at this point (Section 3.1). Following the first principle, the objective function value at the minimum of the relaxed task $\varphi(\mathbf{d}^*)$ is smaller than or equal to the value at the best discrete solution in the current sub-domain $\varphi(\mathbf{d}_{disc}^*)$, i.e.,

$$\left(\varphi(\mathbf{d}^*) = \min_{\mathbf{d} \in \hat{\mathbb{D}}_{rel}^N} \varphi(\mathbf{d}) \text{ s.t. } \mathbf{c}(\mathbf{d}) \geq 0 \right) \leq \left(\varphi(\mathbf{d}_{disc}^*) = \min_{\mathbf{d} \in \mathbb{D}^N \cap \hat{\mathbb{D}}_{rel}^N} \varphi(\mathbf{d}) \text{ s.t. } \mathbf{c}(\mathbf{d}) \geq 0 \right) \tag{28}$$

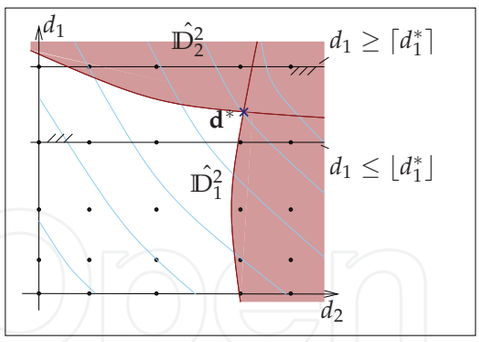
Thus, even if $\mathbf{d}_{disc}^*$ is not explicitly known at this point, the objective function value at the optimum of the relaxed sub-problem $\varphi(\mathbf{d}^*)$ is a lower bound for the sub-domain.
The minimum $\mathbf{d}^*$ which is computed for the relaxed optimization task is not necessarily discrete. Thus, one of the parameters $d_i \in \mathbb{D}_i$ (3) which must be discretized is chosen, and a constraint is set on it to be greater than the next higher or smaller than the next lower discrete value (called branching). In Algorithm 1 (lines 10, 11) this is assigned by $d_i \geq \lceil d_i^* \rceil$ and $d_i \leq \lfloor d_i^* \rfloor$, with

$$\lfloor d_i^* \rfloor = \max_{d \in \mathbb{D}_i} \text{ s.t. } d < d_i \quad \text{and} \quad \lceil d_i^* \rceil = \min_{d \in \mathbb{D}_i} \text{ s.t. } d > d_i \tag{29}$$

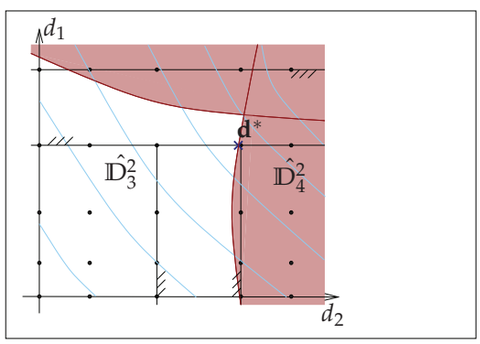
Figure 3(b) shows that adding one pair of such constraints can be considered as building up two new relaxed sub-problems with reduced parameter domain ($\hat{\mathbb{D}}_{up}^N$, $\hat{\mathbb{D}}_{down}^N$ in Algorithm 1 and $\hat{\mathbb{D}}_1^2$, $\hat{\mathbb{D}}_2^2$ in the example in Figure 3(b)). Typically, this is represented by a branching tree (Figure 4): If the parent node of this tree represents the current domain, branching is equivalent to adding two child nodes which correspond to the subsets $\hat{\mathbb{D}}_{up}^N$ and $\hat{\mathbb{D}}_{down}^N$. The edges of the branching tree correspond to the constraints, which are added to define the sub-problems.
For each sub-problem which is set up in the branching step, Algorithm 1 is executed recursively (Algorithm 1, lines 12, 13). Following the heuristic order in Algorithm 1, always $\hat{\mathbb{D}}_{down}^N$ is explored before considering $\hat{\mathbb{D}}_{up}^N$. Thus, in Figure 3 $\hat{\mathbb{D}}_1^2$ is explored before considering $\hat{\mathbb{D}}_2^2$. In the example, after computing the continuous solution of sub-domain $\hat{\mathbb{D}}_1^2$ the sub-problem is further branched, as the continuous solution of the sub-problem is not element of the original discrete domain (Figure 3(c)).
If for each sub-problem branching constraints are added, until the solution of the relaxed sub-problem is a discrete point and thus a leaf of the search tree is reached, the discrete solution with the best objective function value is the optimum of the discrete optimization task. However, without further modification, the computational effort of the method is



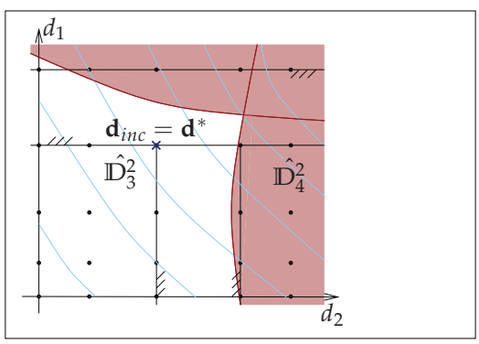
(a) The solution of the relaxed optimization task $\mathbf{d}^*$ is a lower bound for the solution of the discrete minimization $\mathbf{d}^*_{disc}$, which is not explicitly known at this point of time.



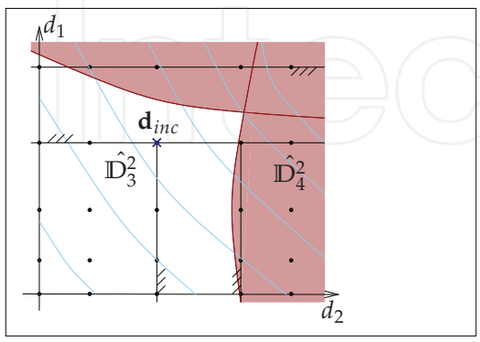
(b) Branching leads to two sub-sets e.g., $\hat{\mathbb{D}}^N_{down} = \hat{\mathbb{D}}^2_1$ and $\hat{\mathbb{D}}^N_{up} = \hat{\mathbb{D}}^2_2$, and two corresponding sub-problems. The parameter which is chosen for branching, is computed by equation (30).



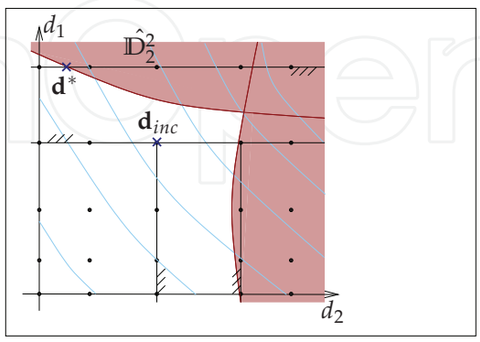
(c) Following Algorithm 1, the lower sub-domain $\hat{\mathbb{D}}_1^2$ is considered next. The relaxed solution of the optimization task in sub-domain $\hat{\mathbb{D}}_1^2$ is non-discrete in parameter d_2. Thus branching constraints are added for this parameter and the sub-domains $\hat{\mathbb{D}}^N_{down} = \hat{\mathbb{D}}^2_3$, $\hat{\mathbb{D}}^N_{up} = \hat{\mathbb{D}}^2_4$ are generated.



(d) Following Algorithm 1, $\hat{\mathbb{D}}^N_{down} = \hat{\mathbb{D}}^2_3$ is considered next. The solution of domain $\hat{\mathbb{D}}^2_3$ is the discrete optimum $\mathbf{d}_{inc}$. The region can be pruned by optimality.



(e) Domain $\hat{\mathbb{D}}_4^2$ is considered next. It does not include any feasible point and can be pruned by infeasibility.



(f) Finally sub-domain $\hat{\mathbb{D}}_2^2$ is explored. The continuous optimum in this region has a higher value than $\mathbf{d}_{inc}$ from Figure 3(d). It can be pruned by value dominance

Fig. 3. Illustration of Branch and Bound for a two-dimensional optimization task

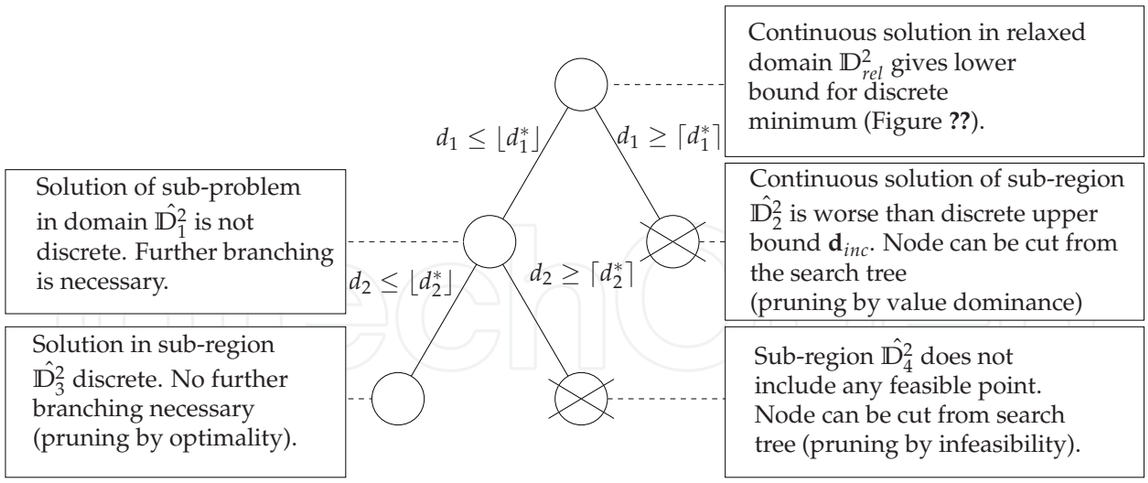


Fig. 4. Illustration of Branch and Bound from Figure 3 as a branching tree.

extremely high, as many non promising sub-problems must be solved. Thus pruning rules (e.g., (Nemhauser & Wolsey, 1988)) are introduced to reduce the run time of the algorithm. The pruning step can be understood as cutting non promising nodes from the search tree. Three pruning rules can be found:

1. If a relaxed sub-problem does not include any feasible solution, the corresponding node can be cut from the search tree. For this rule (known as **pruning by infeasibility**) it is considered that each discrete domain is subset of its relaxation (Algorithm 1, line 2, 3, and Figure 3(e)).
2. If a discrete solution $\mathbf{d}_{inc}$ has been found which is smaller than the value of the current relaxed sub-problem, the node corresponding to the relaxed sub-problem can be cut off. This **pruning by value dominance** uses that – due to principle 1 from above – the relaxed program can not include a discrete point which is better than $\mathbf{d}_{inc}$ (Algorithm 1, line 4, 5, and Figure 3(f)).
3. If the solution of the relaxed program is discrete and better than the best solution found so far, it is a new best solution ($\mathbf{d}_{inc}$) for the discrete sizing task. However, at the same time it is a lower bound for the corresponding relaxed sub-problem which can not include any better point. Thus, no further branching is necessary in the sub-region. This is called **pruning by optimality** (Algorithm 1, line 6, 7, and Figure 3(d)).

The recursive approach described so far realizes a "Depth-First" search. For branching always the most fractional parameter is used (most fractional or most infeasible branching, e.g., Achtenberg et al. (2005)), i.e., assuming an index $i = 1, ..., N_d$ for the discrete parameters d_i with value d_i^*, the branching index i is chosen by

$$i = \arg \max_{i=1,...,N_d} \left| 0.5 - \frac{d_i^* - \lfloor d_i^* \rfloor}{\lceil d_i^* \rceil - \lfloor d_i^* \rfloor} \right| \tag{30}$$

However, some problem specific properties can be used to speed up the algorithm in case of analog sizing. This is described in the following sub-section.

Algorithm 2: Modified Branch and Bound($\mathbf{d}_{inc}$,$\mathbb{D}^N$, $\hat{\mathbb{D}}^N_{rel}$)

Input: incumbent $\mathbf{d}_{inc}$, parameter domain $\mathbb{D}^N$, relaxed domain $\hat{\mathbb{D}}^N_{rel}$
if $\mathbf{d}_{inc}$ *solves the sizing task* **then**
  `// stop due to pruning rule 2'`
  return $\mathbf{d}_{inc}$
end
Run SQP (Section 3.1) on optimization problem

$$\min_{\mathbf{d}\in\hat{\mathbb{D}}^N_{rel}} \varphi(\mathbf{d}) \text{ s.t.: } \mathbf{c}(\mathbf{d} \geq 0)$$

until the sizing task is solved or an optimum is found.
Get solution $\mathbf{d}^*$ and quadratic model from SQP.
if $\mathbf{d}^*$ *does not solve the sizing task (i.e., no solution in* $\hat{\mathbb{D}}^N_{rel}$*)* **then**
  `// pruning rule 1'`
  return $\mathbf{d}_{inc}$
else if $\mathbf{d}^* \in \mathbb{D}^N$ *(i.e.,* $\mathbf{d}^*$ *solves the problem)* **then**
  `// pruning rule 2'`
  return $\mathbf{d}^*$
else
  `// compute incumbent`
  Get $\mathbf{d}^*_{model}$ by solving the discrete quadratic optimization task (31) using Algorithm 1
  if $\mathbf{d}^*_{model}$ *solves the sizing task* **then**
    `// stop`
    return $\mathbf{d}^*_{model}$
  end
  `// branching`
  Choose parameter d_i for branching
  Compute $\lfloor d_i^* \rfloor$ and $\lceil d_i^* \rceil$ according to (34) and (35)
  Set $\hat{\mathbb{D}}^N_{down} = \{\mathbf{d} \in \hat{\mathbb{D}}^N_{rel} \,|d_i \leq \lfloor d_i^* \rfloor\}$
  Set $\hat{\mathbb{D}}^N_{up} = \{\mathbf{d} \in \hat{\mathbb{D}}^N_{rel} \,|d_i \geq \lceil d_i^* \rceil\}$
  $\mathbf{d}_{inc}$ =Modified Branch and Bound($\mathbf{d}_{inc}$, $\mathbb{D}^N$,$\hat{\mathbb{D}}^N_{down}$)
  $\mathbf{d}_{inc}$ = Modified Branch and Bound($\mathbf{d}_{inc}$, $\mathbb{D}^N$, $\hat{\mathbb{D}}^N_{up}$)
  return $\mathbf{d}_{inc}$
end

3.3 Modification of Branch and Bound

To reduce the computational effort of Branch and Bound, the most promising way is to improve the pruning and the branching heuristic. Certain properties of the underlying optimization problem can be used to speed up the process. The modified approach is shown in Algorithm 2 and explained in the following.

3.3.1 Consideration of the quadratic model

As the continuous solution of the relaxed program is computed by an SQP approach, beside the continuous solution a quadratic model of the objective function is computed during

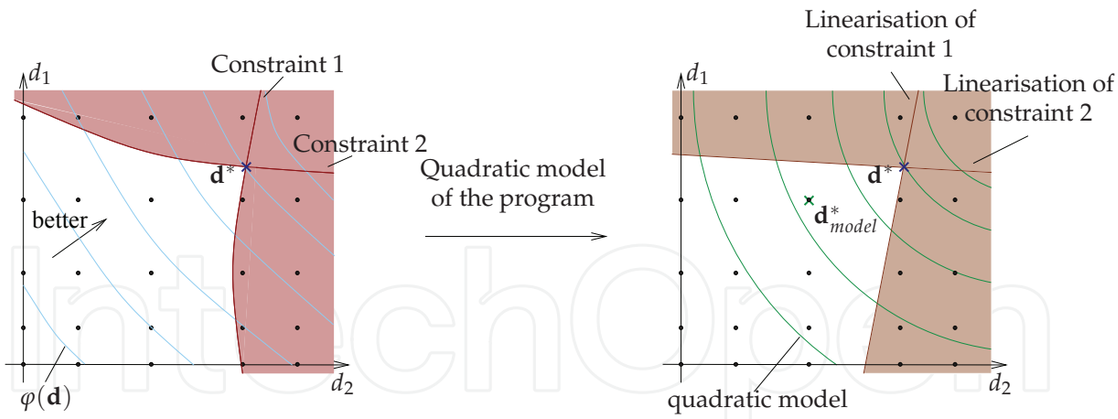


Fig. 5. The quadratic model (right) and the continuous solution of the optimization task $\mathbf{d}^*$ are computed for the original program (left) during SQP. Solving the quadratic model in the discrete domain gives a discrete solution $\mathbf{d}^*_{model}$ which is an upper bound for the original program. In this example, $\mathbf{d}^*_{model}$ is equal to the discrete optimum of the original task.

solving the relaxed program. Furthermore, a linear model of the constraints is computed (Algorithm 2, lines 4, 5). As these models are good local approximations for the relaxed program, they are also a good local approximation for the discrete approach. Thus a quadratic optimization task with linear constraints can be formulated as a surrogate optimization task for (11):

$$\min_{\mathbf{d}\in\hat{\mathbb{D}}} \frac{1}{2}\cdot\mathbf{d}^T\cdot\mathbf{H}\cdot\mathbf{d}+\mathbf{g}^T\cdot\mathbf{d} \;\text{ s.t. }\; \mathbf{J}\cdot\mathbf{d}+\mathbf{c}(\mathbf{d}_0)\geq 0 \tag{31}$$

wherein $\mathbf{H}$ and $\mathbf{g}$ are the Hessian matrix and the gradient for the objective function at the solution point of the relaxed program, $\mathbf{J}$ is the Jacobian matrix for the constraints and $\mathbf{c}_0$ are the constraint values at this point. $\hat{\mathbb{D}}^N$ is the set of discrete points $\mathbb{D}$ in a relaxed sub-problem, i.e.,

$$\hat{\mathbb{D}}^N=\mathbb{D}^N\cap\hat{\mathbb{D}}^N_{up} \quad\text{or}\quad \hat{\mathbb{D}}^N=\mathbb{D}^N\cap\hat{\mathbb{D}}^N_{down} \tag{32}$$

By solving the program in (31) using the discrete domain of the original task, the discrete optimum $\mathbf{d}^*_{model}$ for the approximation of the objective function can be found (Figure 5; Algorithm 2, line 11). Due to the second principle from Section 3.2, the value of $\mathbf{d}^*_{model}$ in the original objective function – i.e., $\varphi(\mathbf{d}^*_{model})$ – is an upper bound for the discrete optimum if it is feasible for the original task. Consequently, sub-regions with a continuous solution worse than the discrete optimum of the model can be cut from the search tree. Thus, early pruning by pruning rule 2 is possible, as – in contrast to standard branch and bound – a discrete upper bound exists in the first branching node and not after discretizing all parameters, i.e., in the first leaf of the search tree. This fact is especially important if many discrete parameters exist. Additionally, solving the quadratic surrogate problem is computational much less expensive, as no circuit simulations are necessary, which cause the highest time consumption in solving the sizing task. Thus, the Branch and Bound algorithm from Section 3.2 can be used to solve the discrete quadratic program with linear constraints in (31).

3.3.2 Consideration of non-optimality

For analog sizing the SQP approach is stopped as soon as any point is found which fulfills specifications and constraints. Thus, the solution which is found is in general non-optimal in terms of the objective function. Taking into account that it is a binary decision if a certain point solves the sizing task or not, the branching rules can be reformulated.

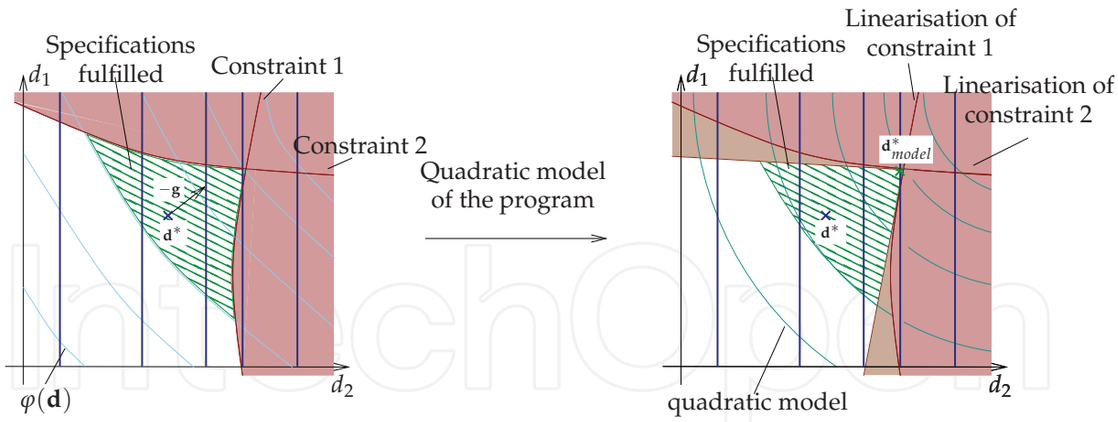


Fig. 6. SQP can be stopped as soon as a continuous point $\mathbf{d}^*$ is found which fulfills constraints and specifications (left). At this point the quadratic model (right) is set up. In the example, the objective function value at the discrete optimum of the quadratic model $\mathbf{d}^*_{model}$ is better than the value at $\mathbf{d}^*$ and specifications and constraints are fulfilled at $\mathbf{d}^*_{model}$.

Assuming that there is at least one discrete solution for the sizing task, obviously only these sub-domains must be considered during Branch and Bound which include such a point. As – due to (22) – the discrete solutions must be also in the relaxed domain, all sub-domains can be cut which do not include a solution of the sizing task in their relaxation. This can be considered by reformulating pruning rule 1 as:

1′. If a relaxed sub-problem does not include *any* solution for the sizing task, the corresponding node can be cut from the search tree (Algorithm 2, lines 6, 7).

If pruning rule 1 is replaced by 1′ the discrete point $\mathbf{d}_{inc}$ – which represents a solution candidate – is only set up, if a discrete solution is found. The Branch and Bound algorithm can be stopped in this case. Thus, pruning rule 2 (pruning by value dominance) becomes redundant and can be left out. Pruning rule 1 is reformulated as a stop criterion, to set up the discrete solution correctly and to avoid insufficient computational effort when the discrete solution has been found:

3′. If any discrete solution for the sizing task has been found, no further branching is required (Algorithm 2, lines 1, 2 and 8, 9).

The modifications of the pruning rules have an even stronger influence if the quadratic model from Section 3.3.1 is considered. In this case, the quadratic model is set up once again in the point $\mathbf{d}^*$ which is computed by SQP and solves the sizing task in the relaxed domain. The point $\mathbf{d}^*$ can be non-optimal in terms of the objective function and thus in may cases the solution of the quadratic optimization problem in the relaxed domain is also a better solution for the underlying sizing task. The continuous solution of the quadratic model is of course not evaluated by simulation. However, as the quadratic model is set up once again at $\mathbf{d}^*$, it is a locally better approximation of the objective function than the quadratic model used for the last SQP step. Thus, even the discrete optimum $\mathbf{d}^*_{model}$ of the quadratic problem in (31) computed by use of the quadratic model at $\mathbf{d}^*$ is often a better solution for the sizing task than $\mathbf{d}^*$ itself (Figure 6).

Hence, in many cases the discrete solution of the model solves the sizing task in the initial node of Branch and Bound and Branch and Bound can be stopped after computing the discrete solution of the quadratic model (Algorithm 2, lines 12, 13).

If the initial solution of the quadratic model does not fulfill the specifications, the

non-optimality of the SQP solution $\mathbf{d}^*$ can also be used to improve the branching heuristic which has significant influence on the runtime of standard Branch and Bound. The gradient at a non-optimal point $\mathbf{d}^*$ is not equal to zero. Thus, it can be assumed that discrete solution candidates can be found in direction of degression of the objective function. The gradient $\mathbf{g}$ at the solution of the SQP algorithm has been already computed to improve the quadratic model and comes without additional cost. For the branching heuristic used in this approach, now the parameter which should be discretized and which corresponds to the gradient component g_i with the strongest influence to the objective function is discretized first (Algorithm 2, line 15). In the "Depth First" search, then the sub-region is chosen which lies in direction of greatest improvement (Algorithm 2, line 16), i.e., assuming the next discrete values for the parameter d_i in domain $\mathbb{D}_i$ from (3) are d_a and d_b, with

$$d_a = \max_{d \in \mathbb{D}_i} \text{ s.t. } d < d_i \quad \text{and} \quad d_b = \min_{d \in \mathbb{D}_i} \text{ s.t. } d > d_i \tag{33}$$

the rounding operator $\lceil \bullet \rceil$ and $\lfloor \bullet \rfloor$ in Algorithm 1 is modified such that

$$\lceil d_i \rceil = \begin{cases} d_a & ; \text{if } g_i > 0 \\ d_b & ; \text{if } g_i \leq 0 \end{cases} \tag{34}$$

and, respectively,

$$\lfloor d_i \rfloor = \begin{cases} d_a & ; \text{if } g_i < 0 \\ d_b & ; \text{if } g_i \geq 0 \end{cases} \tag{35}$$

Thus, discrete points in gradient direction are considered first during branch and bound.

4. Experimental results

To show the effectiveness and efficacy of the algorithm, the sizing process of three different circuits will be presented in this section. For each example, the results and the runtime of SQP with sub-sequent rounding, of SQP and modified Branch and Bound (BaB) without quadratic model, and of SQP and modified Branch and Bound considering the quadratic model (Section 3.3) is presented. The modified Branch and Bound algorithm considering the quadratic model is presented in Section 3.3 Algorithm 2. The modified Branch and Bound algorithm without the quadratic model is implemented identically, but the consideration of the quadratic model (lines 11 - 14 in Algorithm 2) is switched off. I.e., both Branch and Bound approaches stop as soon as a discrete solution for the sizing task is found. Branching in both Branch and Bound algorithms is realized according to (34) and (35).

The circuit in the first example is the Miller amplifier in Figure 7. For the sizing tasks the lengths, widths, and multipliers of the transistors are used as discrete parameters. The lengths of all transistors shall be equal. Furthermore some multipliers and transistor widths (e.g., multipliers and widths of the differential pair) are set equal to avoid mismatch effects. For transistor lengths and widths a $5nm$ manufacturing grid is assumed. The Miller capacitance is represented by a continuous parameter. A $0.5pF$ load capacitance and a $2V$ supply voltage are given for the circuit and the $45nm$ low power predictive technology (PTM; (Balijepalli et al., 2007; Cao et al., 2000; Zhao & Cao, 2006)) from (Nanoscale Integration and Modelling Group, Arizona State University, 2008) is used.

The simulated performance values of the amplifier before and after sizing are shown in Table 1. It can be seen from the results, that – as proposed in Section 1 – the continuous optimization

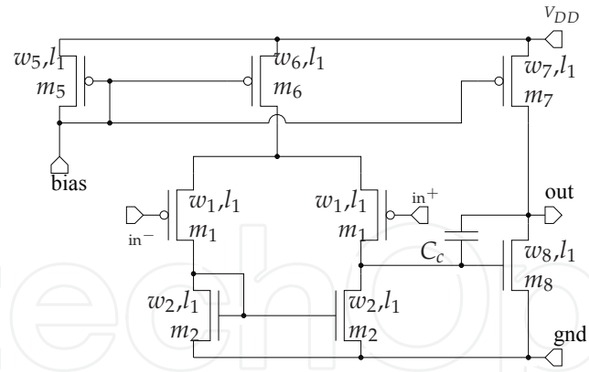


Fig. 7. Miller Amplifier

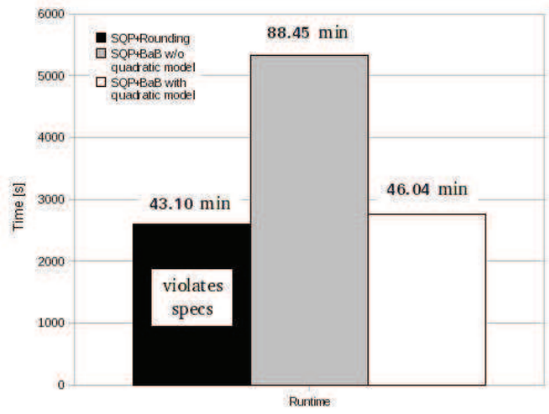


Fig. 8. Runtime for up to 8 times parallelized algorithm on a 16 core 2.67GHz computer for sizing of the Miller amplifier

and subsequent rounding violates two specifications in this case. In contrast, the goal of the discrete sizing task was achieved if Branch and Bound with or without quadratic model has been used. The result quality of Branch and Bound with quadratic model is as good as the result quality achieved without the modification. However, the runtime comparison in Figure 8 clearly shows that the additional runtime for Branch and Bound considering the quadratic model presented in this paper, is significantly smaller, than without the modification and the additional cost compared to the optimization with subsequent rounding is neglectable in this case.

In the second example the sizing of the more complex amplifier in Figure 9, which is proposed in (Martins, 1998), is shown. For this example the 45nm high performance predictive technology model from (Nanoscale Integration and Modelling Group, Arizona State University, 2008) is used and again a 5nm manufacturing grid is assumed. The lengths of all transistors and the widths of transistors which are in the same current mirror or in the same differential pair are set equal. Additionally, some multipliers are set equal considering the symmetries of the circuit. Thus, 14 multipliers, 11 widths, and the length are considered as discrete parameters. Additionally, the compensation capacitance C_c and the bias voltages $V_{bias,1}$ and $V_{bias,2}$ are represented by continuous parameters. A 20pF load capacitance and a 2V supply voltage are given for the circuit. Again the sizing rules from (Massier & Graeb, 2008) are used which define 93 constraints in this case. Specifications and simulated performances

Performance	Specification	Initial values	SQP + Rounding	SQP + BaB w/o quadratic model	SQP + BaB with quadratic model
PSRR $[dB]$	> 135	134	138	139	137
Gain $[dB]$	> 85	89	89	90	89
CMRR $[dB]$	> 135	172	167	167	166
$f_{transit}$ $[MHz]$	> 15	19	24	22	23
$\varphi\ [^\circ]$	> 60	50	**59 (violates spec)**	61	60
SR (rising) $[\frac{V}{\mu s}]$	>15	10	18	16	17
$\lvert$SR (falling)$\rvert$ $[\frac{V}{\mu s}]$	>15	14	37	31	32
Area $[(\mu m)^2]$	< 10	7	9	9	9
Power $[\mu W]$	< 50	56	**52 (violates spec)**	49	49

Table 1. Specification and performance values for Miller amplifier using $45nm$ PTM, $2V$ supply voltage, $1uA$ bias current

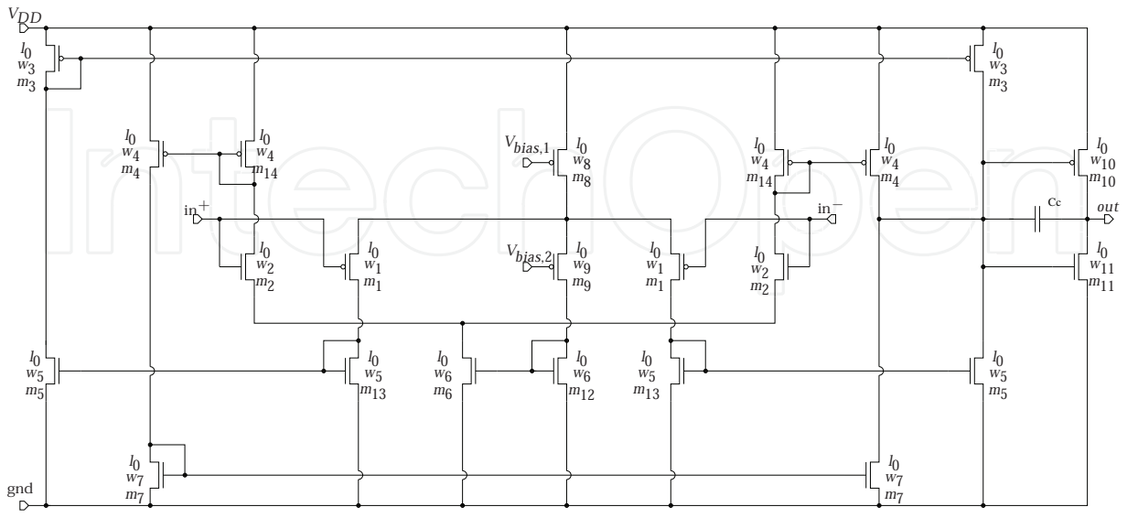


Fig. 9. Low-voltage low-power operational amplifier from (Martins, 1998)

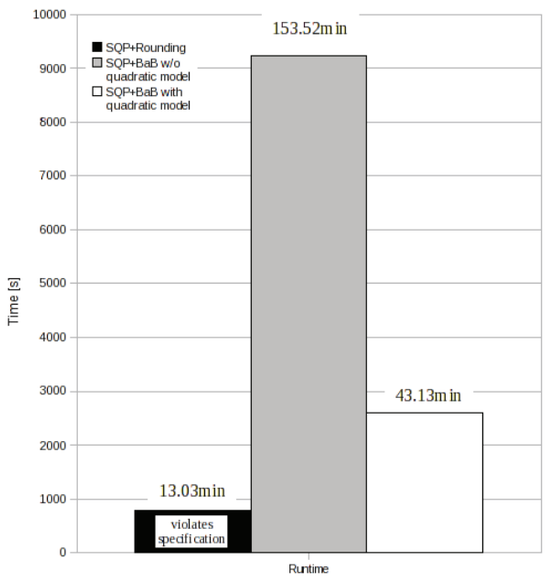


Fig. 10. Runtime for up to 8 times parallelized algorithm on a 16 core 2.67GHz computer for sizing of amplifier in Figure 9

are presented in Tabular 2.
Also in this case specifications are violated when SQP and sub-sequent rounding is used and also here Branch and Bound with and without quadratic model solves the sizing task. The runtime comparison in this case shows that also here Branch and Bound using the quadratic model is much faster than without consideration of the quadratic model. As the number of discrete parameters is much higher in this case, also the runtime of Branch and Bound on the quadratic model is relatively large. Thus potential for further improvement of the algorithm can be seen: The runtime of the algorithm can be reduced if the Branch and Bound algorithm presented in (3.2) is advanced, which is used to find the discrete optimum on the quadratic model and needs approximately half of the computational time in this experiment.
The third example shows the sizing process for the sense amplifier from (Yeung & Mahmoodi, 2006) (see Figure 11). Considering the symmetry of the circuit, 5 multipliers, 5 transistor widths, and the transistor length are used as parameters. For the sizing process a 16nm low power PTM is used and a 2nm manufacturing grid is assumed. Specifications and results are listed in Table 3. For the simulation of the delay it is assumed that the inputs (bit line BL and negative bit line BLB) are preloaded to $V_{DD} = 1.5V$ and the input signal is a voltage reduction by 10mV at one of them. "Delay $+$" in Table 3 is defined as the time between the change of the input signal at the positive input BL and the point of time when the positive output reaches $0.95 \cdot V_{DD}$. Accordingly, the value of "Delay $-$" is defined as the time between the change of the signal at the negative input BLB and the point of time when the positive output reaches $0.05 \cdot V_{DD}$.
The results for this experiment show that in this case continuous optimization with subsequent rounding leads to a solution of the sizing task. This especially happens, if only a few or week constraints and specifications are defined and if only a small number of parameters is used. However, the additional runtime for the modified Branch and Bound approach is only a few seconds. Further analysis of the results shows, that the additional

Performance	Specification	Initial values	SQP + Rounding	SQP + BaB w/o quadratic model	SQP + BaB with quadratic model
PSRR $[dB]$	> 70	59	**63 (violates spec)**	77	79
Gain $[dB]$	> 60	62	65	69	65
CMRR $[dB]$	> 70	57	**69.7 (violates spec)**	81	83
$f_{transit}$ $[MHz]$	> 150	92	274	256	221
φ $[^\circ]$	> 60	85	65	62	60
SR (rising) $[\frac{V}{\mu s}]$	> 10	8	23	16	23
$\lvert$SR (falling)$\rvert$ $[\frac{V}{\mu s}]$	> 10	8	17	14	15

Table 2. Specification and performance values for the amplifier in Figure 9 using $45nm$ PTM, $2V$ supply voltage, $20pF$ load capacity

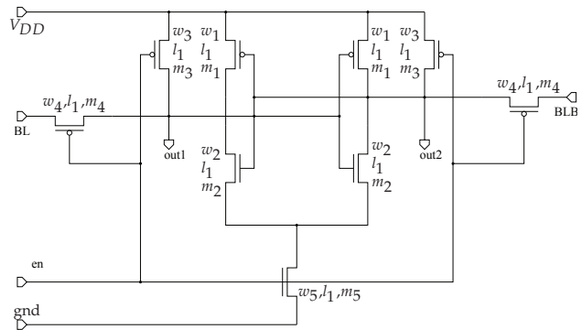


Fig. 11. Sense amplifier

runtime (approximately 30 seconds) is used for computing the gradient and setting up the quadratic model. In contrast to Branch and Bound with consideration of the quadratic model, Branch and Bound without the quadratic model has a significant higher runtime.

5. Conclusion

Sizing of analog circuits is one important task in the analog design flow. In this chapter a new deterministic and gradient-based method has been presented to solve this task. The method solves the relaxed, i.e., continuous sizing task using SQP. Discretization of the result is done by a subsequent Branch and Bound approach under consideration of the quadratic model which is computed during SQP. Additionally certain properties of the underlying sizing task are used to speed up the approach.

The experimental results show that SQP with subsequent rounding can not solve the sizing task in general. In contrast, SQP combined with Branch and Bound is a reasonable approach for sizing analog circuits with discrete parameters. Furthermore, the experimental results show, that the efficacy and efficiency of SQP and Branch and Bound can be increased

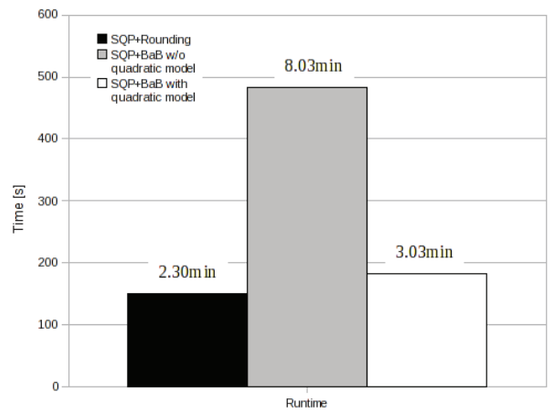


Fig. 12. Runtime for up to 8 times parallelized algorithm on a 16 core 2.67GHz computer for sizing of the sense amplifier

Performance	Specification	Initial values	SQP + Rounding	SQP + BaB w/o quadratic model	SQP + BaB with quadratic model
Delay + $[ps]$	< 60	402	59	46	55
Delay − $[ps]$	< 60	423	48	46	48
static power $[\mu W]$	< 5	27.5	1.4	1.4	2.0
Area $[(\mu m)^2]$	< 0.025	0.225	0.018	0.022	0.019

Table 3. Specification and performance values for sens amplifier using 16nm PTM, 1.5V supply voltage, 1fF Load capacity.

significantly, if the modifications in Section 3.3 are used.
The task presented so far is not able to solve the discrete sizing task, if the circuit performances can only be evaluated for discrete points. Thus, in the future work this problem will be tackled. Additionally, the experiments have shown, that the runtime of the algorithm can be reduced by accelerating the Branch and Bound approach which is used to solve the quadratic model. Also the consideration of non-scalable discrete parameters mentioned in Section 2 is an open task for the future work.

6. References

Achtenberg, T., Koch, T. & Martin, A. (2005). Branching rules revisited, *Operations Research Letters* 33(1): –42– –54.

Alpaydin, G., Balkir, S. & Dundar, G. (2003). An evolutionary approach to automatic synthesis of high-performance analog integrated circuits, *IEEE TEC* 7(3).

Balijepalli, A., Sinha, S. & Cao, Y. (2007). Compact modeling of carbon nanotube transistor for early stage process-design exploration, *ISLPED*.

Cao, Y., Sato, T., Sylvester, D., Orshansky, M. & Hu, C. (2000). New paradigm of predictive mosfet and interconnect modeling for early circuit design, *IEEE CICC*.

Gielen, G. G. E. (2007). Design tool solutions for mixed-signal/RF circuit design in CMOS nanometer technologies, *ASP-DAC*.

Gielen, G., Walscharts, H. & Sansen, W. (1990). Analog circuit design optimization based on symbolic simulation and simulated annealing, *IEEE JSSC* 25(3).

Graeb, H. (2007). *Analog Design Centering And Sizing*, Springer.

Graeb, H., Zizala, S., Eckmueller, J. & Antreich, K. (2001). The sizing rules method for analog integrated circuit design, *ICCAD*.

Knoblinger, G., Kutter, F., Marshall, A., Russ, C., Haibach, P., Patruno, P., Schulz, T., Xiong, W., Gostkowski, M., Schruefer, K. & Cleavelin, C. R. (2005). Design and evaluation of basic analog circuits in an emerging MuGFET technology, *IEEE International SOI Conference 2005*.

Li, D. & Sun, X. (2006). *Nonlinear Integer Programming*, Springer.

Martins, R. (1998). *On the Design of Very Low Power Integrated Circuits*, PhD thesis, Vienna University of Technology.

Massier, T. & Graeb, H. (2008). The sizing rules method for CMOS and bipolar analog integrated circuit synthesis, *IEEE TCAD* 27(12).

Nanoscale Integration and Modelling Group, Arizona State University (2008). URL: http://ptm.asu.edu/ [date: 08.06.2010].

Nemhauser, G. L. & Wolsey, L. A. (1988). *Integer and Combinatorial Optimization*, Jon Wiley & Sons, Inc.

Nocedal, J. & Wright, S. (1999). *Numerical Optimization*, Springer.

Ochotta, E. S., Rutenbar, R. A. & Calrley, L. R. (1996). Synthesis of high-performance analog circuits in ASTRX/OBLX, *IEEE TCAD* 15(3).

Pehl, M. & Graeb, H. (2009). **RaGAzi**: A **r**andom **a**nd **g**radient-based approach to **a**nalog si**zi**ng for mixed discrete and continuous parameters, *ISIC 2009*.

Pehl, M., Massier, T., Graeb, H. & Schlichtmann, U. (2008). A random and pseudo-gradient approach for analog circuit sizing with non-uniformly discretized parameters, *ICCD 2008*.

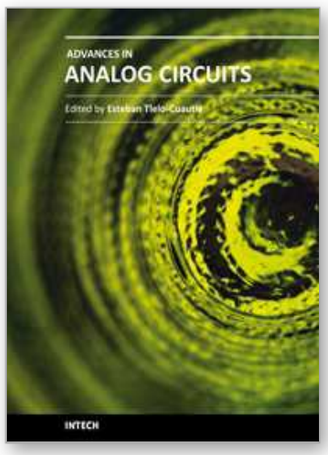
Phelps, R., Krasnicki, M., Rutenbar, R., Carley, L. & Hellums, J. (2000). Anaconda: simulation-based synthesis of analog circuits via stochastic pattern search, *IEEE TCAD* 19(6).

Rutenbar, R. A., Gielen, G. G. E. & Roychowdhury, J. (2007). Hierarchical modeling, optimization, and synthesis for system-level analog and RF designs, *Proceedings of the IEEE*, Vol. 95, IEEE.

Somani, A., Chakrabarti, P. & Patra, A. (2007). An evolutionary algorithm-based approach to automated design of analog and rf circuits using adaptive normalized cost functions, *IEEE TEC* 11(3).

Yeung, J. & Mahmoodi, H. (2006). Robust sense amplifier design under random dopant fluctuations in nano-scale cmos technoloties, *IEEE ISOCC*.

Zhao, W. & Cao, Y. (2006). New generation of predictive technology model for sub-45nm early design exploration, *IEEE Transactions on Electron Devices* 53(11).



Advances in Analog Circuits

Edited by Prof. Esteban Tlelo-Cuautle

ISBN 978-953-307-323-1
Hard cover, 368 pages
Publisher InTech
Published online 02, February, 2011
Published in print edition February, 2011

This book highlights key design issues and challenges to guarantee the development of successful applications of analog circuits. Researchers around the world share acquired experience and insights to develop advances in analog circuit design, modeling and simulation. The key contributions of the sixteen chapters focus on recent advances in analog circuits to accomplish academic or industrial target specifications.

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen