Stochastic branch & bound applying target oriented branch & bound method to optimal scenario tree reduction

Volker Stix

Vienna University of Economics

Department of Information Business

Augasse 2–6

A-1090 Vienna / Austria

volker.stix@wu-wien.ac.at

Dec 20, 2001

Abstract

In this article a new branch & bound method is described. It uses an artificial target to improve its bounding capabilities. Therefore the new approach is faster compared to the classical one. It is applied to the stochastic problem of optimal scenario tree reduction. The aspects of global optimization are emphasized here. All necessary components for that problem are developed and some experimental results underline the benefits of the new approach.

Keywords — scenario trees, global optimization, stochastic branch and bound

1 Introduction

A general stochastic maximization problem is of the following form:

$$\begin{aligned} E(f(\mathbf{x})_\omega) &\rightarrow \text{max!} \\ \mathbf{x} &\in \mathcal{M} \end{aligned} \tag{1}$$

where f is a random variable in $\mathbf{x}$ and ω. The vector $\mathbf{x}$ lies in the feasible set of possible actions $\mathcal{M}$, which can be a finite set (discrete optimization) or an infinite one (continuous optimization) and ω lies in the set of all possible scenarios Ω and models the influence of randomness. The function E is a real-valued function, which is usually the expectation of the random variable. For our investigations a real valued objective value is required only therefore from now we write shortly f and $f(\mathbf{x})$ to describe the concatenated functions $E \circ f$.

In Section 2, a new approach extending the classical $\mathcal{B}\&\mathcal{B}$ method is described. It can be applied to stochastic and deterministic $\mathcal{B}\&\mathcal{B}$ procedures. Section 3 describes a stochastic problem, to which the new approach is applied. The problem deals with optimal reduction of large scenario trees. Section 4 shows possible instances of the required inputs for the classical but also target oriented $\mathcal{B}\&\mathcal{B}$ algorithm. Experimental results which illustrate the benefit of the new $\mathcal{B}\&\mathcal{B}$ method end this section.

2 The target oriented $\mathcal{B}\&\mathcal{B}$ method

This section presents the idea of the introduction of a target in a classical $\mathcal{B}\&\mathcal{B}$ method. It is discussed in more detail in [9]. For a basic understanding, important issues are repeated in short and without theorems here.

2.1 Classical $\mathcal{B}\&\mathcal{B}$ methods

$\mathcal{B}\&\mathcal{B}$ methods are well known for a long time and often used in various field of continuous and discrete application domains. The basic idea behind $\mathcal{B}\&\mathcal{B}$ methods is the division of a problem in several smaller subproblems of the same kind. Horst/Tuy [3] describe the $\mathcal{B}\&\mathcal{B}$ method for continuous global optimization and criteria of convergence are developed which are necessary for infinite $\mathcal{B}\&\mathcal{B}$ procedures. They focus on the partition of the feasible set whereas the reduction in the dimension of the problem is more commonly applied to discrete GO problems, see e.g. [5, 2]. Due to the finiteness of the feasible sets, discrete GO problems need no theory of convergence criteria. Continuous domain problems, however, can be as well reduced in subproblems by reducing its dimension as for example it is achieved for standard quadratic problems in [1].

A classical $\mathcal{B}\&\mathcal{B}$ approach requires the following inputs which are assumed to be given:

1. The function f and the feasible set $\mathcal{M}$.
2. Upper bounds for a problem $p = (f, \mathcal{M})$.
3. A branching rule and a selection rule.

The classical $\mathcal{B}\&\mathcal{B}$ algorithm's outline looks like this:

```
Algorithm 1:

     Input: The problem $p = (f, \mathcal{M})$.
            A desired $\epsilon$-precision.
Initialize: Set problem list $pl = \{p\}$.
            Set as first maximizer any $\hat{\mathbf{x}} \in \mathcal{M}_p$.
    Output: $\hat{\mathbf{x}}$ is one global maximizer of $p$.

  1. Use the selection rule to remove the next problem $p \in pl$.
```

2. Use the branching rule to construct new subproblems $p_1, \dots, p_i$ out of p.

3. For each p_i do

   (a) Calculate a lower bound l_i for p_i (e.g. by evaluating any feasible point in $\mathcal{M}_{p_i}$).

   (b) If $(l_i > f(\hat{\mathbf{x}}))$ then set $\hat{\mathbf{x}} = \mathbf{x}$, for $f(\mathbf{x}) = l_i$.

   (c) Calculate an upper bound u_i for p_i.

   (d) If $(u_i - l_i > \epsilon$ and $u_i > f(\hat{\mathbf{x}}))$ then set $pl = pl \cup p_i$.

4. Repeat from step 1 if $pl \neq \phi$.

Algorithm 1 starts with the first (main) problem together with a desired ϵ-precision as input. It chooses any $\hat{\mathbf{x}} \in \mathcal{M}$ as candidate for the global maximizer. In step 1 the next subproblem (which is the main problem in case of the first run) is removed from the list by the selection rule. This rule can be a simple one like LIFO or FIFO if *pl* is organized as a list, which semantically implements depth-search-first and breadth-search-first respectively. It can also be a more complicated heuristic regarding structural aspects of the problem. We do not care about this selection rule and assume it to be given. Step 2 decomposes the problem using the branching rule into "smaller" instances. This is done either by reducing the problem's dimension or by partitioning the feasible set (or even both). Step 3 iterates through these newly generated subproblems and does the following: (a) it calculates a lower bound for the subproblem. This can be done by extracting one feasible point out of $\mathcal{M}_{p_i}$ and evaluating it under f. A more efficient alternative would be a (fast) local optimization procedure for p_i, if available. (b) it updates the best solution $\hat{\mathbf{x}}$ if necessary. (c) it calculates an upper bound for the subproblem. (d) it decides whether it is necessary to explore p_i further. This is done (i) by testing if the desired precision is already reached and (ii) by testing if the problem p_i is able to improve our so far best solution found by comparing u_i with $f(\hat{\mathbf{x}})$. Step 4 tests whether there are still subproblems left for exploration. After termination of Algorithm 1, $\hat{\mathbf{x}}$ is one global maximizer within an ϵ-precision. Its optimal value is $f(\hat{\mathbf{x}})$.

The initialization step can be improved as well as step (3a) by using not just any vector $\hat{\mathbf{x}}$ but a local optimizer of the respective problem, if it can easily be obtained.

2.2 Introducing a target

The improvement of the generic $\mathcal{B}\&\mathcal{B}$ algorithm presented in Section 2.1 for optimizing the problem $p = (f, \mathcal{M})$ is the introduction of a *target*. A target is a value which should be reached at least by a vector $\mathbf{x} \in \mathcal{M}$, i.e. $f(\mathbf{x}) \geq target$ should hold. Of course if *target* is chosen too large, e.g. $target > UB_p$, then $f(\mathbf{x}) \geq target$ can never be satisfied. On the other hand if *target* is chosen too small, e.g. $target = f(\mathbf{y}) = LB_p$ (for any $\mathbf{y} \in \mathcal{M}$), there is no challenge finding that particular $\mathbf{x}$ (simply choose $\mathbf{x} = \mathbf{y}$). If the target is chosen, however, to lie between LB_p and UB_p, e.g. $target = \frac{LB_p+UB_p}{2}$, it is unclear whether there is a $\mathbf{x} \in \mathcal{M}$ such that $f(\mathbf{x}) \geq target$. If such a vector $\mathbf{x}$ can be found, we have a new lower bound: $LB_p = f(\mathbf{x})$. If we fail finding such a vector $\mathbf{x}$ then a new upper bound is found: $UB_p = target$. In both cases the estimation of the global maximum of p is improved.

2.3 The algorithm

Before going into more detail we will show how this target oriented $\mathcal{B}\&\mathcal{B}$ method works and we will explain it afterwards.

Algorithm 2:

Input: The problem $p = (f, \mathcal{M})$.

A desired ϵ-precision.

```
Initialize: Set problem list $pl = \{p\}$.
            Set remember list $rl = \phi$.
            Set as first maximizer any $\hat{\mathbf{x}} \in \mathcal{M}_p$.
            Calculate the global upper bound $UB_g$ of the main problem $p$.
            Set $target = \frac{f(\hat{\mathbf{x}})+UB_g}{2}$.
Output: $\hat{\mathbf{x}}$ is one global maximizer of $p$.

1. Use the selection rule to remove the next problem $p \in pl$.
2. Use the branching rule to construct new subproblems $p_1, \ldots, p_i$ out of $p$.
3. For each $p_i$ do
   (a) Calculate a lower bound $l_i$ for $p_i$ (e.g. by evaluating any feasible
       point in $\mathcal{M}_{p_i}$).
   (b) If $(l_i > f(\hat{\mathbf{x}}))$ then set $\hat{\mathbf{x}} = \mathbf{x}$, for $f(\mathbf{x}) = l_i$ and set $target = \frac{f(\hat{\mathbf{x}})+UB_g}{2}$
       if $(f(\hat{\mathbf{x}}) \geq target)$.
   (c) Calculate an upper bound $u_i$ for $p_i$.
   (d) If $(u_i - l_i > \epsilon)$ then
           If $(u_i \geq target)$ then
               set $pl = pl \cup p_i$.
           else If $(u_i \geq f(\hat{\mathbf{x}}))$ then
               set $rl = rl \cup p_i$.
4. Repeat from step 1 if $pl \neq \phi$.
5. Set $pl = rl$, $UB_g = target$, $target = \frac{f(\hat{\mathbf{x}})+UB_g}{2}$.
6. If $(UB_g - f(\hat{\mathbf{x}}) > \epsilon$ and $pl \neq \phi)$ repeat from step 1.
```

Algorithm 2 delivers the global optimum together with one optimizer of problem (1) within precision ϵ in finite time or at least converges to it for $\epsilon = 0$ [9].

Algorithm 2 is very similar to Algorithm 1 except for minor changes. The most salient ones are in steps 3d and steps 5–6. In step 3d, besides precision, successive subproblems are only investigated further if their upper bound is not smaller than *target* (instead of $f(\hat{\mathbf{x}})$ as in Algorithm 1). This would imply that we have already found an optimizer $\tilde{\mathbf{x}}$ where $f(\tilde{\mathbf{x}}) = target$. Therefore all branches where the upper bound can not cope with these "harder" requirements are cut back. Consequently, under normal conditions, much fewer subproblems are calculated during recursion. In step 5, the newly gained information is stored (UB_g) and the target value is re-initialized. Because the target was set so high (there never was such a $\tilde{\mathbf{x}}$ as implied before), we might have discarded efficient subproblems. Therefore we have to look at these remembered problems in *rl* (*pl* = *rl*) with the newly gained information (step 6). On the other hand, if $UB_g - f(\hat{\mathbf{x}}) \leq \epsilon$, we found the global maximizer to be $\hat{\mathbf{x}}$.

We should examine step 2b as well where the statement $target = \frac{f(\hat{\mathbf{x}})+UB_g}{2}$ carries over dynamically the idea of the target once there is a $\hat{\mathbf{x}}$ with $f(\hat{\mathbf{x}}) \geq target$. Again the goal is set higher than necessary. Because *target* is only monotonically increasing there is no problem of inconsistency by changing the target value at run time.

2.4 The idea illustrated with a simple example

The following example tries to outline the benefits of this target oriented $\mathcal{B}\&\mathcal{B}$ method. Though the example might appear artificial, which is necessary because in normal applications, the problem tree is too large to be used as an elucidatory example.

For simplicity, avoiding lots of decimal points, we chose a discrete optimization example, therefore $\epsilon = 1$. The upper numbers inside the nodes are the upper bounds of this subproblem and the lower numbers are the local solutions (lower bounds) inside these problems.

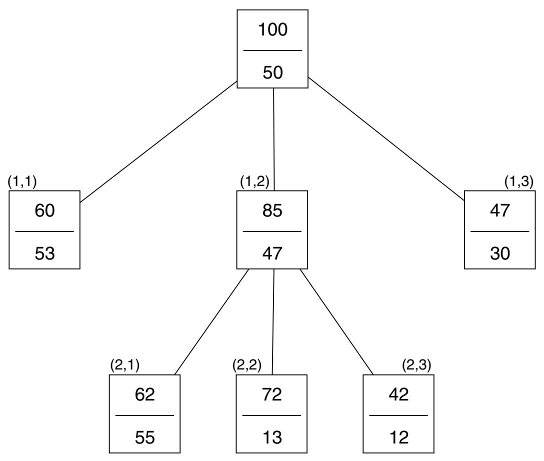

Figure 1: First run through the tree. $UB_f = 100$, $target = 75$.

Figure 1 shows the first run through the tree. The upper bound UB_g for the main problem is 100 and $f(\hat{\mathbf{x}}) = 50$ for some arbitrary chosen $\hat{\mathbf{x}}$ and thus $target = 75$. The branching rule then generates the problems (1,1),(1,2) and (1,3) and their local estimators (lower bounds) and upper bounds. A better optimizer $\hat{\mathbf{x}}$ is found in problem (1,1) with $f(\hat{\mathbf{x}}) = 53$ but the upper bound of this problem is smaller than *target*, therefore this problem is rejected this time (unlike the classical $\mathcal{B}\&\mathcal{B}$ method) but remembered on a list. Problem (1,3)'s upper bound does not exceed $f(\hat{\mathbf{x}})$ and is therefore discarded (like in the classical $\mathcal{B}\&\mathcal{B}$ method). Problem (1,2)'s upper bound exceeds *target* and therefore new subproblems and their values are constructed. A new maximizer is found in (2,1) with $f(\hat{\mathbf{x}}) = 55$. Again the upper bound is too small as it is in problem (2,2), therefore both are rejected this time but remembered whereas problem (2,3) is discarded permanently the same as problem (1,3). At the end of this run (step 5) we have $f(\hat{\mathbf{x}}) = 55$ and a new proven upper bound of 75. Additionally we know that only problems (1,1),(2,1) and (2,2) should be considered for the next run ($pl = rl$). Our new target now is $65 = (75 + 55)/2$.

Figure 2 shows the second run through the tree. The new upper bound now

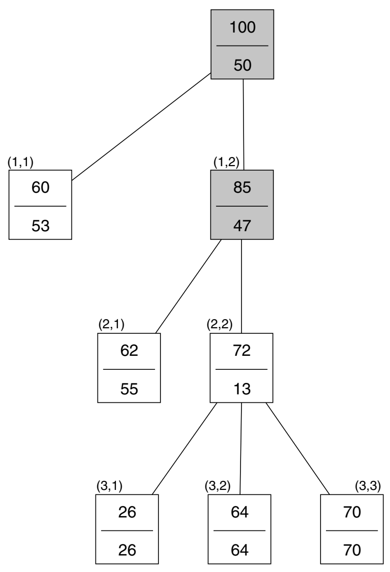

Figure 2: Second run through the tree.$UB_f = 75$, $target = 65$.

is 75, which is not considered in the main node and node (1,2) because these nodes are rendered and are no longer visited. They appear in the illustration simply for better orientation. We start with problem (1,1) and see that this problem's upper bound is still too low. We reject and remember it. The same result is valid for node (2,1). Node (2,2), however, constructs new subproblems, which are this time easy enough to be rendered globally (i.e. upper and lower bounds coincide). We improve the maximum to 64 in problem (3,2) and to 70 in problem (3,3), which resets the target value to $72,5 = (70 + 75)/2$ (step 3b), and after that this run is finished. We proved that 72,5 is an upper bound and found $f(\hat{\mathbf{x}}) = 70$ and know that we now only have to look at problems (1,1) and (2,1). The third run now is trivial with no computation and ends with the proof that the global optimum is 70.

In contrast to the classical $\mathcal{B}\&\mathcal{B}$ method not only problems (1,3) and (2,3) were never expanded, but also problems (1,1) and (2,1). The classical method might got stuck in the left-most branch starting with problem (1,1). Once again, this is an artificial problem, constructed to illustrate the main points.

3 Scenario tree optimization

It is well known that scenario trees tend to grow exponentially with the number of observed periods. This fact makes exact evaluation hard and often even impossible. Statistical methods like Monte-Carlo sampling [8] are sometimes not efficient enough because in practice the size of the sample set vanishes in comparison to the huge sample space. Therefore, a reduction of a scenario tree can aid to deal with such problems. The reduction must be done carefully in order to maintain the main properties of the original model. Such a reduction in an optimal sense is presented here.

3.1 Introduction to the problem

The problem which is shortly described below is well discussed in [6]. The problem has normally additional constrains as in [4]. Our scenario tree here is unconstrained as we would like to focus here on the global optimization aspects and the benefits of the target oriented $\mathcal{B}\&\mathcal{B}$ approach. Let X be a finite real valued data set. We would like to approximate X by a smaller data set Z. X represents time discrete sample paths of a stochastic process given through a random variable of a theoretical model or through historical observations. The data set Z has a given smaller cardinality, depending on the optimization model.

We can discriminate the problems between two cases:

1. A discrete random variable X must be supported by one real valued point z.

2. A discrete random variable X must be supported by another discrete random variable Z. Where $|Z| < |X|$.

The quality of the support is subject of optimization here. We measure the quality in the distance of the implied distribution functions. The following gives a general distance measure for distribution functions F and $\tilde{F}$ which implies the so called *transportation metric*, which is related to the mass transportation problem [7].

$$d(F, \tilde{F}) := \sup_g \{ \int g(u) dF(u) - \int g(u) d\tilde{F}(u) : L(g) \leq 1 \} ,$$

where $L(g)$ is the Lipschitz-constant of function g. For discrete distributions the distance between these two functions are

$$d(F, \tilde{F}) = \sum_{i=1}^{n} \min_j |x_i - z_j| ,$$

where x_i (z_i) is the i-th element in X (Z).

The transport metric measures the cumulative effort in L_1 of transporting each $x \in X$ to the nearest supplier $z \in Z$. All this forms the unconstrained optimization problem in the vector variables $z_1, \ldots, z_m$, where m is the desired cardinality of Z:

$$\sum_{i=1}^{n} \min_j |x_i - z_j| \quad \rightarrow \quad \text{min!} \tag{2}$$

3.2 Global optimization

Case 1: One supplier We are looking for the a single point z which supports the set X in the best way with respect to (2).

Lemma 1 *The optimal point z which minimizes (2) for a real valued data set X, is the median of X.*

Proof The proof is done indirectly. If z was not the median, then (without loss of generality) $P_l = P(X \leq z) < P(X > z) = P_h$. Moving z by Δz to the next larger $x \in X$ would increase the transportation effort of points smaller than z by $P_l \cdot \Delta z$ but decrease the transportation effort of points larger than z

by $P_h \cdot \Delta z$. P_h is larger than P_l and therefore z can not be optimal. Thus z must be the median.

Case 2: Multiple suppliers We are now interested in the best support of a discrete random variable X by a smaller random variable Z. It is clear that two given neighboring suppliers z_1 and z_2 imply a border $b_1 = \frac{z_1+z_2}{2}$ of their attraction region. Given any $z_1, \ldots, z_m$ they imply m attraction regions (for each supplier one) as follows:

$$b_0 < z_1 < b_1 < \ldots < b_{m-1} < z_m < b_m,$$

where $b_0 = -\infty$ and $b_m = \infty$ and else $b_i = \frac{z_i+z_{i+1}}{2}$.

Theorem 2 *Each z_i must be the median of the data samples $x \in X$ lying within its attraction region, i.e. for all $\{x : b_{i-1} \leq x < b_i\}$.*

Proof If z_i is not the median within its region, then by Lemma 1 moving z_i toward the median lowers the transport effort for all attracted points. During this movement the attraction region may change. The overall distance measure decreases, however, as long as the movement is towards the median of the attracted points. This is because points which are no longer attracted have shorter distances to other suppliers and newly attracted data points chose z_i just because it was the best choice then. The movement of z_i stops when it is the median of its attracted points.

This iterative process described in the proof of Theorem 2 can be used for a local optimization procedure as follows:

1. Choose any initial values $z_1 < \ldots < z_m$.

2. For each $i = 1, \ldots, m$ find all points A_i within the attraction region of z_i and set z_i to the median of A_i.

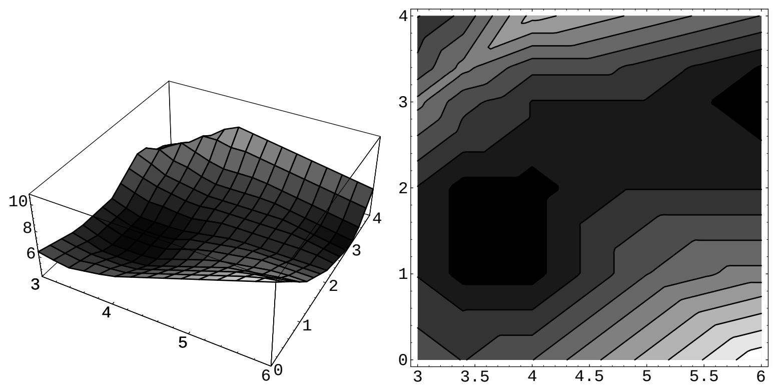

Figure 3: Infinite local optimizers only one global.

3. Repeat with step 2 until the optimality criterion of theorem 2 is satisfied.

Unfortunately this method finds local optimizers only but there are many of them as the following simple example demonstrates.

Example 1 Let us set $X = (1, 2, 3, 3.5, 4, 6)$ and Z of cardinality two, i.e. we are optimizing the program:

$$\sum_{i=1}^{6}(\min(|x_i - z_1|; |x_i - z_2|)) \quad \rightarrow \ \min!$$

There exist infinitely many minimizers $\hat{\mathbf{z}} = (\hat{z}_1, \hat{z}_2)$ out of the set $\{[1;2] \times [3.5;4]\} \cup \{(3,6)\}$ with minimal objective value 4.5. By slightly changing one data sample x_4 to 3.4, the local optimizers stay almost the same but minimizers out of $\{[1;2] \times [3.5;4]\}$ with an objective value of 4.6 are no longer efficient compared to the global minimizer $(3,6)$ and its objective value of 4.4. Figure 3 shows this second modified problem as a 3D and density plot respectively. You can observe the two black colored local pools. The left one is a plateau the right one a strict local and global minimizer. The last portion on the upper left of the drawing is just the mirror image which results by swapping z_1 with z_2.

For global optimization we investigate the problem further. From now on we assume the enumeration of data out of X (Z) to be ordered, i.e. $x_1 < \ldots < x_n$. We assume that we have a chain of data values X, the variables Z and the implied borders b_i:

$$x_1 \ldots x_{I_1} z_1 x_{I_1+1} \ldots x_{I_2} b_1 x_{I_2+1} \ldots x_{I_3} z_2 \ldots b_2 \ldots x_{I_{2m-1}} z_m x_{I_{2m-1}+1} \ldots x_n$$

, where I_i are monotone increasing index numbers between 1 and n.

The evaluation of this chain delivers the transportation effort for it. Note that by theorem 2 all $z \in Z$ are implied by their bounds and vice versa. Because all supporters $z \in Z$ can always coincide with a data sample $x \in X$ there are $\binom{n}{m-1}$ possible distributions of borders and thus implied chains. This can be a huge number of chains which can be bounded by intelligent $\mathcal{B}\&\mathcal{B}$ procedures as described in the next section.

4 Applying target oriented $\mathcal{B}\&\mathcal{B}$ to scenario tree optimization

In this section the global optimization problem as described in the previous section is solved using the target oriented $\mathcal{B}\&\mathcal{B}$ approach as introduced in Section 2. The required inputs for that algorithm are chosen as described below.

Branching rule The branching rule, as motivated in the previous section, simply iterates through all possible chains. It starts by setting the first border b_1 to all possible $n-m+1$ positions within the chain. These $n-m+1$ possibilities form new smaller subproblems of the same kind. They inherit all data samples $x_i > b_1$ and have only $m-1$ suppliers. Using this branching rule recursively the $\mathcal{B}\&\mathcal{B}$ tree is constructed. It reduces the problem's dimension and therefore it is finite.

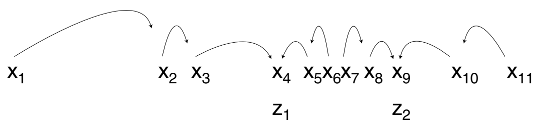

Figure 4: Neighborhood of data samples.

Upper and lower bounds Leaves of that tree can be evaluated in order to form upper bounds (simple evaluations of a defined chain). Lower bounds are more complex to extract, which is normal for minimization problems.

If the data samples $x \in X$ are transported to their nearest decision variable $z \in Z$ we know:

1. m samples out of X do not have to be moved.
2. At most $2 \cdot m$ of the remaining data samples must be moved one single step to the next data sample in the direction of its supplier.
3. At most $2 \cdot m$ of the remaining data samples must be moved two single steps to the next two data samples in the direction of its supplier.
4. and so on...

Step one is true because m data samples $x \in X$ form the suppliers Z itself (the median can always coincide with a data sample). Step two is true because each supplier can have at most two attracted immediate neighboring data samples, and can have at most two second nearest attracted neighboring data samples and so on. Figure 4 illustrates this fact. Samples x_4 and x_9 act as suppliers. They do not have to be transported. Samples x_3, x_5, x_8 and x_{10} must be moved one step only. The data x_1, however, must be moves for instance three steps, and so on. The size of the steps differs and depend on the data set X.

If the number n of data samples and the number m of suppliers is known, the described neighborhood order can be used to estimate a lower bound of the transportation effort. Let D_i be a set of the i-th order distances defined as

$$D_i = \{x_{1+i} - x_1, x_{2+i} - x_2, \ldots, x_n - x_{n-i}\} \ ,$$

and let S_i be an ordered list of D_i and let $S_i[j]$ be the j-th element out of S_i. Then, by the motivation above, the following algorithm forms a lower bound on the transportation effort:

```
$effort = 0;$
$order = 1; num = 1; anz = 1;$
for $n - m$ times do
    $effort = effort + S_{order}[num];$ $num = num + 1; anz = anz + 1;$
    if $(anz > 2 \cdot m)$ then
                    anz=1; num=1; class=class+1;
```

We know that m data points do not have to be moved. In best case, all suppliers attract the same number of data points. $2 \cdot m$ of the remaining $n - m$ samples must move only one step to its next supplier. This effort is successively added through the sorted list of 1-st order distances. The next $2 \cdot m$ samples move 2 steps, therefore the 2-nd order sorted distances are added and so on. The distances to the suppliers are not known therefore the shortest available distance (one distance can be used only once) is assumed, which yields to a lower bound of the effort.

4.1 Experimental results

Some experiments where made with the results developed in the previous section. Data sets consisting of 100 samples were artificially constructed. The first block of the tables below are 100 samples out of an equi-, normal-, chi-square-, exponential- and beta-distribution. The second block is a mixture (not an addition) of two and three normal-distribution and a random mixture of all data

Name	First	Global optimum	Iterations	Quality
Equi	466.12	464.58	676840	83%
Normal	22.63	22.44	979023	82%
Chi-Square	87.35	85.95	1112504	80%
Exponential	15.18	15.18	1874782	89%
Beta	2.90	2.82	818237	81%
2*Normal	104.3	83.66	1008521	63%
3*Normal	208.35	170.81	1269791	72%
Random	255.80	178.51	795719	61%

Table 1: Experimental Results I

samples mentioned so far. The idea of the second block is, to make distributions multi modal and more complex.

In Table 1, 10 supporters were used in order to find the global optimal reduction. The first local estimation of the transportation effort was made using Theorem 2 by starting with equi-distributed suppliers z_i. This result is shown in column “first”. The next two columns shows the global optimum together with the number of sub-problems required to solve the problem by the target oriented $\mathcal{B}\&\mathcal{B}$ method. Column “quality” shows the amount of iterations required by target $\mathcal{B}\&\mathcal{B}$ compared to the number of iterations consumed by the classical method. You can observe, that problems with a larger gap between first and global solution perform better.

In Table 2, 20 supporters were used. No experiment of these was able to find the global optimizer within one hour. The calculations were stopped and upper and lower bounds are illustrated for both the classical and the target oriented $\mathcal{B}\&\mathcal{B}$ method. You can observe that the estimation of the lower bounds is much better for the target oriented $\mathcal{B}\&\mathcal{B}$ method.

Name	First	Best	Lower bound
with target			
Equi	213.56	213.56	170
Normal	12.27	12.27	8.82
Chi-Square	46.08	46.08	34.85
Exponential	9.17	9.17	6.17
Beta	1.55	1.55	1.12
2*Normal	39.09	38.41	29.67
3*Normal	92.08	85.12	65.63
Random	197.67	74.10	64.95
without target			
Equi	213.56	213.56	127
Normal	12.27	12.27	5.38
Chi-Square	46.08	46.08	23.61
Exponential	9.17	9.17	3.17
Beta	1.55	1.55	0.69
2*Normal	39.09	38.68	20.24
3*Normal	92.08	88.78	39.18
Random	197.67	75.10	34.50

Table 2: Experimental Results II

5 Conclusion

This article should demonstrate how easy it is to implement the target oriented $\mathcal{B}\&\mathcal{B}$ strategy and the last section demonstrates the benefits of this approach. It should be noted that either all problems were rendered by both of the algorithms or they were too complex for both of them and had to be terminated. For many examples in [9], however, the global optimum was rendered with the target oriented $\mathcal{B}\&\mathcal{B}$ approach only, whereas for the classical approach these problems were too hard to solve. Additional even more stunning results on the quality improvements can be found there.

References

[1] I. M. Bomze and V. Stix. Genetic engineering via negative fitness: Evolutionary dynamics for global optimization. *Annals of Oper. Res.*, 89, 1999.

[2] W. Cook. *Combinatorial Optimization.* Wiley, New york, 1998.

[3] R. Horst and H. Tuy. *Global Optimization.* Springer, Heidelberg, 3rd edition, 1996.

[4] K. Høyland and S. W. Wallace. Generating scenario trees for multistage decission problems. *Management Science*, 47(2):295–307, 2001.

[5] R. G. Parker. *Discrete Optimization.* Academic Press, Boston, 1988.

[6] G. Ch. Pflug. Scenario tree genaration for multiperiod financial optimization by optimal discretization. *Math. Programming*, Ser. B 89:251–271, 2001.

[7] S. T. Rachev. *Probability metrics and the stability of stochastic models.* Wiley, New York, 1991.

[8] C. P. Robert and G. Casella. *Monte Carlo statistical methods.* Springer, New York, 1999.

[9] V. Stix. Target oriented branch & bound method for global optimization. Technical Report, submitted TR2001-04, Department of Statistics and Decision Support Systems, University of Vienna, April 2001.