Search Heuristics for Box Decomposition Methods

Stefan Ratschan
e-mail: Stefan.Ratschan@risc.uni-linz.ac.at*

February 7, 2003

Abstract

In this paper we study search heuristics for box decomposition methods that solve problems such as global optimization, minimax optimization, or quantified constraint solving. For this we unify these methods under a branch-and-bound framework, and develop a model that is more convenient for studying heuristics for such algorithms than the traditional models from Artificial Intelligence. We use the result to prove various theorems about heuristics and apply the outcome to the box decomposition methods under consideration. We support the findings with timings for the method of quantified constraint solving developed by the author.

1 Introduction

In box decomposition methods for problems such as global optimization [26, 15, 49], minimax optimization [53, 24] and quantified constraint solving [45, 46, 23] that proceed by decomposing the search space into boxes, heuristics for choosing boxes critically influence the run-time of the algorithms. Such search heuristics have been developed for global optimization [36, 49, 8] but it is not obvious how they can be applied to methods for solving the other problems mentioned.

The contribution of this paper is the clarification of how such heuristics can be used in the above box decomposition methods, and a common framework for studying them. This framework consists of: First, a formulation of the methods under consideration as branch-and-bound algorithms (that may be nested in the sense that we branch and bound on the result of further branch-and-bound procedures). Second, a formal model for according heuristic search that takes into account the specifics of box decomposition methods, without restricting itself to unnecessary algorithmic details. This makes the result better suited for our problems than the classical models from Artificial Intelligence [40, 37].

We demonstrate the relevance of the developed framework by showing how and why the box choice heuristics available in the literature for global optimization [36, 49, 8] can be applied to minimax optimization [53] and approximate quantified constraint solving [45]. The author believes that the presented framework provides an ideal basis for further exploration of heuristics for box decomposition algorithms and even general space decomposition methods in continuous domains.

*This work was supported by the Austrian Science Fund FWF in the frame of the project SFB F1303

The structure of the paper is as follows: In Section 2 we present our formulation of box decomposition methods as nested branch-and-bound algorithms; in Section 3 we develop a model of heuristic search in such algorithms; in Section 4 we apply the results to the box decomposition methods under consideration; in Section 5 we do some timings for the application of the resulting heuristics in quantified constraint solving; in Section 6 we discuss related work; and in Section 7 we make final conclusions.

2 Nested Branch-and-Bound Algorithms

We want to develop heuristics for algorithms that use box decomposition to solve problems such as the following:

Global Optimization: Given a function $f : D \to \mathbb{R}$, where $D \subseteq \mathbb{R}^n$, find its global minimum.

Minimax Optimization: Given a function $f : D_1 \times \ldots \times D_n \to \mathbb{R}$, where $D_1 \times \ldots \times D_n \subseteq \mathbb{R}^n$, find

$$\min_{x_1 \in D_1} \max_{x_2 \in D_2} \ldots \min_{x_{n-1} \in D_{n-1}} \max_{x_n \in D_n} f(x_1, \ldots, x_n).$$

Quantified Constraint Solving: Given a first-order formula over the reals [12], that is, an expression that contains quantifiers ($\exists$, $\forall$), connectives ($\wedge$, $\vee$, $\neg$, $\Rightarrow$), predicate symbols (e.g., $=$, $<$, $\leq$), function symbols (e.g., $+$, $-$, $\times$, sin, exp), rational constants and variables ranging over the reals, find its solution set (or truth-value in the case of a closed formula).

A naive application of game tree search methods from Artificial Intelligence [40, 37] to minimax optimization would start from the observation that a minimax optimization problem represents a game tree for which each tree level represents a certain variable, each node has infinitely many children (one for each substitution of a real constant for the according variable), and the weight of a tree leaf is given by the value of the function f at the according constant. However, game tree search methods are restricted to *locally finite* trees, that is, to trees for which every node has finitely many children.

One can search locally finite trees by extracting one child after the other from a node. However, this is impossible for infinite trees, and so we recursively split the infinite set of children into (finitely many) pieces and prune branches on which we can prove that no solution can be found. This is just branch-and-bound search. In the following we will illustrate this fact on the example of minimax optimization [53, 24] and then apply the findings to the other problems.

Assume that we want to find $\min_{x \in [0,10]} \max_{y \in [0,10]} xy + 1$. By taking the range of $xy+1$ on $[0, 10] \times [0, 10]$, which is the interval $[0, 101]$, we can infer that the resulting minimax value also is in this interval.

For refining the result we do a branching step that splits the range $[0, 10]$ of x into the pieces $[0, 5]$ and $[5, 10]$, resulting in

$$\min\{\min_{x \in [0,5]} \max_{y \in [0,10]} xy + 1, \min_{x \in [5,10]} \max_{y \in [0,10]} xy + 1\},$$

Here we can infer the bound $[0,5][0,10]+1=[0,51]$ for the first branch, and $[5,10][0,10]+1=[0,101]$ for the second one, resulting in an improved overall bound $[0,51]$.

Now we can continue to improve this bound by doing further branching along the variable x, and by computing bounds on the resulting branches. Note however, that for making these bounds converge, we have to do branching along the variable y as well. This means that, nested within the branch-and-bound procedure wrt. the variable x, we use the results of another branch-and-bound procedure wrt. the variable y.

If we always branch into pieces that consist of floating point intervals, we can compute the necessary bounds as follows:

- For functions that contain the usual arithmetical symbols such as addition, multiplication, or trigonometric functions we can use interval arithmetic [35, 26],
- a bound of a function of the form $\min_{x \in D'} f(x,y)$ on the set D is given by the bound of f on $D \times D'$,
- a bound of a function of the form $\min\{f,g\}$ on the set D is given by $[\min\{\underline{f},\underline{g}\}, \min\{\overline{f},\overline{g}\}]$, where $[\underline{f},\overline{f}]$ is the bound of f on D, and $[\underline{g},\overline{g}]$ is the bound of g on D, and
- a bound of a function of the form $\max\{f,g\}$ on the set D is given by $[\max\{\underline{f},\underline{g}\}, \max\{\overline{f},\overline{g}\}]$, where $[\underline{f},\overline{f}]$ is the bound of f on D, and $[\underline{g},\overline{g}]$ is the bound of g on D.

The last two cases correspond to their usual treatment in interval arithmetic [25], the second case gives a very crude overestimation of the actual value, but it improves due to branching.

By replacing the computed upper and lower bounds by Boolean values and by defining a similar interval overestimation for $\forall$, $\exists$ and the Boolean functions $\vee$, $\wedge$, and $\neg$, we get an algorithm for quantified constraint solving, and by minimizing a function on all its variables at once we get global optimization. The resulting algorithms are simplified versions of the original ones [53, 45, 26, 15, 49, 48], that keep the essentials needed for studying which sub-expression to choose for branching.

For studying this problem we will first develop an abstract formal model. Then we will consider various relevant heuristics within this model, and at last we will apply the result to the above algorithms.

3 Box Choice Heuristics

In this section we assume familiarity of the reader with the notion of a *tree*, and *sub-tree*, *root*, *node*, *leaf*, and *path* of a tree, and *child*, *ancestor*, and *depth* of a tree node. Given a sequence q and a property P, we say that q *eventually* fulfills P if there is a k such that for all $i \geq k$, q_i fulfills P.

Recall that for implementing a branch-and-bound scheme for minimization we split one of the elements of $\min\{\min_{x \in D_1} f, \ldots, \min_{x \in D_n} f\}$ into pieces. We want to study heuristics for choosing the optimal element. For this we use a representation of such branch-and-bound schemes that abstracts from all unnecessary details and just reflects its branching structure.

We use a tree for which the root is marked with the set D of the initial symbol $\min_{x \in D}$, $\max_{x \in D}$, $\exists x \in D$ or $\forall x \in D$, and each node is marked with a subset of D created by branching. We call such a tree a *search tree* and the sets at the nodes *search domains*.

Each branching step takes a leaf L with search domain D, splits D into pieces D_1 and D_2, and attaches according new leaves to L. For example, Figure 1 represents the search tree created when branching according to the example in Section 2.

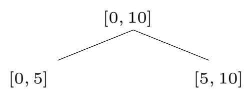

Figure 1: Search Tree

Note that such a search tree just represents branching wrt. one variable. However, its nodes can capture a sub-expression of the input on which we can again apply a branch-and-prune scheme wrt. a different variable. For a branch-and-prune scheme that appears nested within another one, each branching of the outer scheme splits the inner schemes into two duplicates.

Now we mark paths along which all search domains contain a solution of the problem as *solution paths* (similar trees already have been used by Lawler and Wood [31] to visualize branch-and-bound methods). For example, in global optimization and minimax optimization these are the paths containing optimizers. Since we cannot follow infinite paths throughout, in practice one usually finishes after following a solution path to a certain depth.

By using a breadth-first search in such a search tree we can now explore a solution path up to the desired depth. However, breadth-first search eventually expands the whole tree, which is very inefficient. So we want to find a better method. For this we adapt best-first search [40, 37] to our model and show how this results in special heuristics for box choice.

We use a *merit function* r that assigns a real value to a tree node and assume that branching always chooses the element of highest merit, where ties are broken in an arbitrary way.

Then a certain merit function r results in a sub-tree of a search tree T as follows: Assume a function $\text{expand}_{r,T}$ (the *expansion function*) that takes a subtree S of T and returns S plus all the children of the leaf of T with maximal merit according to r, where ties are broken in an arbitrary way. Furthermore we define that the application of $\text{expand}_{r,T}$ to the empty tree $\{\}$ results in the root of T.

Recursive application of the expansion function, beginning with the empty tree results in a tree $\text{expand}^{\infty}_{r,T}(\{\})$ (the smallest tree that contains all elements of $\{\text{expand}^{i}_{r,T}(\{\}) | i \in \mathbb{N}\}$, or in other words, the least upper bound of $\{\text{expand}^{i}_{r,T}(\{\}) | i \in \mathbb{N}\}$, see any textbook on denotational semantics [51] for similar constructions). Note that this usually results in a tree with paths of infinite length. The resulting sub-tree of a search tree T models the set of all search trees created when box choice uses a certain merit function r. Now we study the properties of the resulting search:

Definition 1 *A merit function r is* complete *for a tree T iff the tree $expand^{\infty}_{r,T}(\{\})$, contains a solution path. A complete merit function r is* weak *for a tree T iff $expand^{\infty}_{r,T}(\{\}) = T$, otherwise it is* strong.

Note that we only look for one solution path in this paper, because this is sufficient for quantified constraint solving (see Section 4). The following results can be easily reformulated for finding all solution paths. Now we need to choose a merit function for which a search will eventually follow at least one solution path to arbitrary depth. That is, we want to have complete merit functions.

Theorem 1 *Given a merit function r, and a finite set of infinite paths P in a tree T, if there is a path p in P s.t. for all nodes N in P all paths not in P eventually have merit lower than N then at least one element of P is contained in $expand^{\infty}_{r,T}(\{\})$.*

Proof.

Assume that there is a path p in P s.t. for all nodes N in P all paths not in P eventually have merit lower than N.

First we prove that $\text{expand}^{\infty}_{r,T}(\{\})$ contains infinitely many nodes contained in P. For this we prove that for all i, there is a $k > i$, such that $\text{expand}^{k}_{r,T}(\{\})$ contains at least one more node from P than $\text{expand}^{i}_{r,T}(\{\})$.

Let N be the node of p which is a leaf of $\text{expand}^{i}_{r,T}(\{\})$. By our assumption there is a $k > i$ such that all leaves of $\text{expand}^{k}_{r,T}(\{\})$ not in P have merit lower than N. Thus $\text{expand}^{k+1}_{r,T}(\{\})$ contains one more element of P than $\text{expand}^{i}_{r,T}(\{\})$.

This means that $\text{expand}^{\infty}_{r,T}(\{\})$ contains infinitely many nodes contained in P and thus, by König's lemma, it contains an infinite path, and this path is an element of P. ■

Corollary 1 *Given a merit function r and a search tree T with finitely many solution paths, if there is a solution path in T on which for all nodes with merit M, all non-solution paths eventually have merit lower than M, then r is complete for T.*

The theory of heuristic search provides analogous theorems for searching graphs (e.g., Theorem 1, page 104 of the book by Pearl [40]).

Corollary 2 *If the merit function goes to $-\infty$ on all non-solution paths of a tree T with finitely many solution paths then it is complete for T.*

By applying Theorem 1 to each path we get:

Corollary 3 *If the merit function goes to $-\infty$ on all paths of a tree T, then it is complete but weak for T.*

Now we prove a theorem on when paths will not be expanded:

Theorem 2 *Given a merit function r and a search tree T, if on a path P the merit is eventually smaller than the merit of all nodes on another path P', then* $\mathrm{expand}_{r,T}^{\infty}(\{\})$ *does not contain P.*

Proof. Assume that on P the merit is eventually smaller than the merit of all nodes on another path. Then there is an n such that the merit of the node N of P which is a leaf of $\mathrm{expand}_{r,T}^{n}(\{\})$, is smaller than the merit of the leaf belonging to P'. So N will never be expanded, that is, it is a leaf of $\mathrm{expand}_{r,T}^{k}(\{\})$, for all $k \geq n$. This means that that $\mathrm{expand}_{r,T}^{\infty}(\{\})$ does not contain P. ∎

Corollary 4 *A merit function r that has a lower bound on all solution paths of a tree T and that is eventually smaller than the smallest such bound on all non-solution paths of T, is complete and strong for T.*

Proof. By Theorem 1 all solution paths are in $\mathrm{expand}_{r,T}^{\infty}(\{\})$, and by Theorem 2 all non-solution paths are not in $\mathrm{expand}_{r,T}^{\infty}(\{\})$. ∎

Now we introduce a new type of search behavior:

Definition 2 *A merit function r is* depth-first wrt. a path P *in a tree T iff for all nodes N of other paths P' the merit of P is eventually greater than the merit of N.*

Clearly a merit function is depth-first wrt. every path with infinite limit. Depth-first behavior is only desired for solution paths — otherwise, if we ever explore such a path to a certain depth, we will not be able to explore any other path from then on.

4 Application to Box Decomposition Algorithms

Within this section we will restrict all our considerations to search trees with finitely many solution paths. We start with the situation when no special information about the sub-trees is available. This means that a merit function can only use information about the depth of a node in the search tree.

The merit function should not increase with the depth, because otherwise the function would show dangerous depth-first behavior. By Corollary 3 we can use the negative depth of a node in the tree, but in this case we still search the whole tree. We are also in a similar situation if we use information about search domains such as their volume or maximal side-length (heuristics of this type have been used in global optimization [15, 26] or minimax optimization [53]).

Now we consider the situation where we can use the bounds computed in a branch-and-bound approach (as defined in Section 2). For any node N we denote the according lower bound by $\underline{N}$ and upper bound by $\overline{N}$. The limit of $\overline{N} - \underline{N}$ goes to zero along all paths, and $\overline{N}$ is monotonically decreasing, and $\underline{N}$ is monotonically increasing. Furthermore we assume that our problem is posed in such a way that both the limit of $\overline{N}$ and the limit of $\underline{N}$ is strictly smaller on solution paths than on non-solution paths (which is the usual case for minimization problems). Our question is: Which merit function should we choose?

A classical merit function [36] is $r_B(N) := -\underline{N}$. It is complete by Corollary 4. In fact, this is the only reasonable complete merit function because any merit function that takes $\overline{N}$ into account is not complete for all search trees:

Theorem 3 *For every merit function $r(\underline{N}, \overline{N})$ that has a lower bound on non-solution paths, and for which for every $\underline{N} \in \mathbb{R}$,*

$$\lim_{\overline{N} \to \infty} r(\underline{N}, \overline{N}) = -\infty$$

there is a search tree for which it is not complete.

Proof. Assume such a merit function. We construct a search tree such that for all nodes of the tree that split into a solution path and a non-solution path, the merit of the first node N_s on the solution path is smaller than all the merits on the non-solution path (this can be done because of the lower bound on non-solution paths). For such a search tree no solution path will be fully expanded.

We construct this tree by letting $\underline{N_s}$ be arbitrary but letting $\overline{N_s}$ be such that $r(\underline{N_s}, \overline{N_s})$ is smaller than the merit of all nodes on the non-solution path. Such a $\overline{N_s}$ exists because $r(\underline{N}, \overline{N})$ goes to negative infinity as $\overline{N}$ goes to infinity. ∎

For example the negation of the average of the bounds $-\frac{\underline{N}+\overline{N}}{2}$ has these properties, but also every other reasonable combination of $\underline{N}$ and $\overline{N}$.

By Corollary 4 we even get:

Theorem 4 *The merit function r_B is strong for all search trees.*

So r_B is the optimal merit function for all branch-and-bound algorithms that fit into our model and have only information about the bounds $\underline{N}$ and $\overline{N}$ available. For example one can improve the algorithm by Z. Shen and others [53] by replacing the highest width merit function used there by r_B.

Now we assume that we have even more information available: The global minimum, or at least a good approximation of the global minimum. Lately a merit function that uses this information has appeared, the reject index [8, 4]. It is defined as follows:

$$r_R := \frac{\tilde{f} - \underline{N}}{\overline{N} - \underline{N}}$$

Here $\tilde{f}$ is an approximation of the global minimum of f.

Now let us study how the value $\tilde{f}$ influences the convergence of search algorithms. First note the following two observations:

- The limit of r_R is ∞ on all paths for which the limit of $\underline{N}$ is smaller than $\tilde{f}$.
- The limit of r_R is $-\infty$ on all paths for which the limit of $\underline{N}$ is bigger than $\tilde{f}$.

Now let us assume that $\tilde{f}$ is a constant that is strictly bigger than the limit of $\underline{N}$ (and equally $\overline{N}$) on solution paths. In this case r_R goes to ∞ for all solution paths, but possibly also for some non-solution paths.

Lemma 1 *The merit function r_R is eventually monotonically increasing on all paths for which the limit of $\underline{N}$ is smaller than $\tilde{f}$.*

Proof. We have to prove that on all such paths there is a node N such that the path is monotonically increasing beginning from N. Choose a node N such that $\overline{N}$ is smaller than $\tilde{f}$. To prove that r_R increases we show that $\tilde{f} - \underline{N}$ decreases by a smaller fraction than $\overline{N} - \underline{N}$ on P beginning from N. This is surely the case since $\underline{N} < \overline{N} < \tilde{f}$ there. ■

Now we get (compare with Theorem 1 in the paper by Csendes [8]):

Theorem 5 *The merit function r_R is depth-first wrt. the paths for which the limit of $\underline{N}$ is smaller than $\tilde{f}$.*

It is easy to find search trees for which r_R will do a depth-first search into a non-solution path, and we get:

Corollary 5 *If $\tilde{f}$ is strictly larger than the limit of $\underline{N}$ (or equally $\overline{N}$) on solution paths, then there is a search tree for which the reject index is not complete.*

In fact we cannot hope that it will be complete for any but a few search trees. So in this case the reject index will usually not find a solution path.

Now let us assume that $\tilde{f}$ is strictly smaller than the limit of $\underline{N}$ (or equally $\overline{N}$) on solution paths. Then r_R goes to $-\infty$ on all paths, even if $\tilde{f}$ is not constant but varies for each computation of r_R. We get by Corollary 2 (compare with Theorem 2 in the paper by Csendes [8]):

Theorem 6 *If $\tilde{f}$ is smaller or equal the limit of $\underline{N}$ (or equally $\overline{N}$) on solution paths, then the reject index is complete for all search trees.*

So we can use the reject index for our computation if we ensure that $\tilde{f}$ underestimates the global minimum. However, by Corollary 3, if $\tilde{f}$ does not converge to the global minimum, then we still do breadth-first search:

Theorem 7 *If $\tilde{f}$ is smaller or equal the limit of $\underline{N}$ (or equally $\overline{N}$) on solution paths, but does not converge to the global minimum, then the reject index is weak for all search trees.*

However, if the approximation $\tilde{f}$ is good, then by Corollary 4 our search behavior is better (compare with Theorem 3 in the paper by Csendes [8]):

Theorem 8 *If $\tilde{f}$ goes to the global minimum from below and is greater or equal than $\underline{N}$ for all nodes N on solution paths, then the reject index is complete and strong for all search trees.*

However, if we use the best currently available bound on the global minimum as the approximation $\tilde{f}$, then the reject index does not give any improvement over r_B:

Theorem 9 *If for all i, $\tilde{f}$ is equal to the minimal $\underline{N}$ of leaf nodes N of $expand^i_{r_R,T}(\{\})$, then $expand^\infty_{r_R,T}(\{\}) = expand^\infty_{r_B,T}(\{\})$.*

Proof.

We prove that in this case $\text{expand}_{r_R,T}$ always expands the same node as $\text{expand}_{r_B,T}$. Then the theorem holds by induction.

Let T be a tree and let N be the leaf of T chosen by r_B. This means that $r_B(N) = -\underline{N}$ is maximal, that is, $\underline{N}$ is minimal. So the value of $r_r(N)$ is $\frac{\underline{N}-\underline{N}}{\overline{N}-\underline{N}} = 0$. For all other nodes the value of the reject index is negative. Thus N will be also chosen by the reject index. ■

Now let us apply our results to quantified constraint solving. Consider the example of a constraint of the form $(\forall x)\ f(x) \geq 0$. Here one of the two following cases can occur:

- The whole constraint is true, that is, $f(x) \geq 0$ is true for all x.
- The whole constraint is false, that is, $f(x) \geq 0$ is false for at least one x.

However, we do not know beforehand, in which situation we are in, that is, whether to look for x where $f(x) \geq 0$ is true or where it is false. But observe that in the first case we have to prove the truth of $f(x) \geq 0$ for all x anyway. This means that in this case all search heuristics are equally efficient. But when $f(x) \geq 0$ is false somewhere, then we have to find an x that proves this as fast as possible. Thus we can always use heuristics that are optimal for the second case. A similar dual argument holds for existential quantification.

Now it remains to investigate for which x, we can prove fastest that $f(x) \geq 0$ is false. Clearly this is the global minimum of f. Thus we use exactly the same box choice heuristics for quantified constraint solving as for minimax optimization. One can even reformulate quantified constraint solving as a minimax optimization problem followed by a sign computation, for example when studying the numerical stability of quantified constraint solving [44], or when designing algorithms in the area [52].

5 Timings

We did experiments on the use of various merit functions in approximate quantified constraint solving [45], resulting in the timings listed in Table 1. The examples are from robust control [11] (problem name starts with C) and computational geometry [33] (problem name starts with M). The columns marked with r_V refer to a merit function which uses the volume of a box (this is the original merit function that we used), r_B denotes the merit function $-\underline{N}$ for universal quantification and the dual $\overline{N}$ for existential quantification, and r_R denotes the reject index and its dual.

The columns marked with T designate the time (in seconds), and the columns marked with A designate the number of refinements of atomic constraints. A refinement of an atomic constraint can be a range computation for the according polynomial or a tightening step [18] — this refinement usually takes about 60% of the runtime, and thus is a very good measure for the quality of heuristics.

As one can see, the reject index needs exactly as many atomic refinements as r_B, which confirms Theorem 9. Clearly our heuristics improve the behavior of the algorithm. Note furthermore that the examples $M4$, $M21$, $M24$ and $M24$ need the same number of atomic refinements for all cases. The reason is, that all these examples have the form $\exists \vec{x}\ p(\vec{x})$, without free variables, and furthermore they are false. As discussed in Section 4 this means that we have to prove that $p(\vec{x})$ is false for all $\vec{x}$, anyway, and no search heuristics can change this.

Table 1: Timings

Problem	r_V		r_B		r_R		$100\frac{r_B}{r_V}$	
	T	A	T	A	T	A	T	C
C01a	3.7	2285	0.2	283	0.2	283	5	12
C05b	47.3	5856	47.0	5852	46.6	5852	99	100
C05c	0.2	144	0.2	160	0.2	160	100	111
C08a	0.1	30	0.1	25	0.1	25	100	83
C12b	1.7	1799	1.4	1512	1.4	1512	82	84
C14a	0.5	81	0.3	55	0.3	55	60	68
C16a	0.4	679	0.3	474	0.3	474	72	70
C19a	11.6	3986	3.4	1754	3.5	1754	29	44
C21a	46.9	419	34.6	351	34.6	351	74	84
C24	1.5	2668	0.7	1333	0.7	1333	43	50
M1	0.4	74	0.2	52	0.3	52	50	70
M3	0.9	179	0.6	120	0.5	120	67	67
M4	0.2	44	0.2	44	0.2	44	100	100
M5	0.8	162	0.4	90	0.4	90	50	56
M8	0.7	108	0.6	103	0.6	103	86	95
M21	0.3	83	0.3	83	0.3	83	100	100
M22	0.4	86	0.3	86	0.4	86	75	100
M23	0.2	364	0.1	196	0.2	196	50	54
M24	0.5	124	0.5	124	0.5	124	100	100
Sum	118.3	19171	91.2	12697	91.1	12697	77	66
Average							71	76

6 Related Work

Related work on search heuristics has either been restricted to very specific algorithms (e.g. in global optimization or constraint satisfaction) [36, 49, 3, 16, 8, 14], to specific algorithm inputs [47, 42], to discrete domains [19], or it has been done from a general viewpoint of heuristic search [40, 37] but with an orientation toward problems specific to Artificial Intelligence (e.g., game tree search, automated reasoning).

For algorithms in global optimization Moore and Ratschek [36] discuss correctness and termination of global optimization algorithms and different box choice criteria. Ratz [49] (§2.2.5.1) uses a variant of the smallest lower bound criterion with an additional criterion for breaking ties. Berner [3] discusses the properties of various box choice strategies, and Csendes [8] investigates the reject index. For general discussion see various books on the subject [26, 15, 48].

Heuristics for branch-and-bound algorithms for discrete domains, already were discussed very early [31, 34] and have later been theoretically justified, for example using stochastic models [56]. They are especially important for parallel branch-and-bound [6].

Related research in Artificial Intelligence [40, 37] studies search procedures for two major classes of objects: (Locally finite) graphs [10, 13], and (locally finite) AND/OR graphs (especially game trees) [55, 1, 21, 27, 28, 41, 32, 50, 39]. Our most general case of nested branch-and-bound corresponds to the second case, but has to search a tree at *each* level, because we have to deal with locally infinite trees.

Main differences of our framework to heuristic search algorithms from Artificial Intelligence are the following: First, we do not have goal nodes — instead we are interested in solution paths that can be infinite. Second, we do not define the quality of a solution — we are just interested in finding a solution as fast as possible. Third, we have bounds $\underline{N}$ and $\overline{N}$ available that we can use for computing the merit of a node N (up to our knowledge the only algorithms from AI that use such bounds are the ones by Ibaraki [21], and Berliner's $B*$ [2, 38]).

There have been several efforts to unify heuristics in discrete branch-and-bound with heuristic search algorithms from Artificial Intelligence [19, 20, 29, 30]. There is also a bibliography on heuristic search from the branch-and-bound viewpoint [54] that covers literature up to the year 1992.

Search heuristics for Collin's algorithm [7], a symbolic analogon to the quantified constraint solving method of the author, have been developed by H. Hong [17].

7 Conclusion

We have studied box choice heuristics for various types for branch-and-bound algorithms by providing a common framework for them. The author believes that the presented framework provides an ideal basis for further exploration of heuristics for box decomposition algorithms and even general space decomposition methods in continuous domains.

Possible future work includes:

- Defining and studying a measure for the quality of merit functions that is more fine-grained than just being weak or strong, and investigating whether there is a merit function that is in some sense provably optimal
- Finding heuristics for other search decisions in the quantified constraint solving method developed by the author, such as choice of bisection direction [9], choice of the optimal sub-constraint for connectives, choice of the optimal box for approximate quantifiers [43] or for boxes corresponding to free variables.
- Investigating methods for finding a reasonable approximate optimum to use for the reject index (e.g., by using local optimizers or numerical sampling).
- Analyzing further connections with results in game tree search.

The author thanks Arnold Neumaier for important hints to the literature, and Hoon Hong for the introduction into the area of quantified constraint solving.

References

[1] B. Abramson. Control strategies for two-player games. *ACM Computing Surveys*, 21(2), 1989.

[2] H. Berliner. The $B*$ tree search algorithm: A best-first proof procedure. *Artificial Intelligence*, 12(1):23–40, 1979.

[3] S. Berner. New results on verified global optimization. *Computing*, 57(4):323–343, 1996.

[4] L. G. Casado, I. García, and T. Csendes. A heuristic rejection criterion in interval global optimization algorithms. *BIT*, 41:683–692, 2001.

[5] B. F. Caviness and J. R. Johnson, editors. *Quantifier Elimination and Cylindrical Algebraic Decomposition*. Springer, Wien, 1998.

[6] J. Clausen and M. Perregaard. On the best search strategy in parallel branch-and-bound: Best-first search versus lazy depth-first search. *Annals of Operations Research*, 90:1–17, 1999.

[7] G. E. Collins. Quantifier elimination for the elementary theory of real closed fields by cylindrical algebraic decomposition. In Caviness and Johnson [5], pages 134–183.

[8] T. Csendes. Convergence properties of interval global optimization algorithms with a new class of interval selection criteria. *Journal of Global Optimization*, 19:307–327, 2001.

[9] T. Csendes and D. Ratz. Subdivision direction selection in interval methods for global optimization. *SIAM Journal on Numerical Analysis*, 34(3):922–938, 1997.

[10] R. Dechter and J. Pearl. Generalized best-first search strategies and the optimality of A*. *Journal of the ACM*, 32(3):505–536, 1985.

[11] P. Dorato, W. Yang, and C. Abdallah. Robust multi-objective feedback design by quantifier elimination. *Journal of Symbolic Computation*, 24:153–159, 1997.

[12] H.-D. Ebbinghaus, J. Flum, and W. Thomas. *Mathematical Logic*. Springer Verlag, 1984.

[13] H. Farreny. Completeness and admissibility for general heuristic search algorithms-a theoretical study: Basic concepts and proofs. *Journal of Heuristics*, 5(3):353–376, 1999.

[14] A. Fernandez and P. Hill. Branching: The essence of constraint solving. In *Proceedings of the Sixth Annual Workshop of the ERCIM Working Group on Constraints*, 2001. `http://arXiv.org/html/cs/0110012`.

[15] E. Hansen. *Global Optimization Using Interval Analysis*. Marcel Dekker, New York, 1992.

[16] P. V. Hentenryck. Numerica: A modeling language for global optimization. In *Proceedings of the 15th International Joint Conference on Artificial Intelligence*, Nagoya, Japan, 1997.

[17] H. Hong. Heuristic search strategies for cylindrical algebraic decomposition. In J. Calmet et al., editors, *Proceedings of Artificial Intelligence and Symbolic Mathematical Computing, Springer Lecture Notes in Computer Science 737*, pages 152–165, 1992.

[18] H. Hong and V. Stahl. Safe starting regions by fixed points and tightening. *Computing*, 53:323–335, 1994.

[19] T. Ibaraki. Theoretical comparisons of search strategies in branch-and-bound algorithms. *International Journal of Computer and Information Sciences*, 5(4):315–344, 1976.

[20] T. Ibaraki. Branch-and-bound procedure and state-space representation of combinatorial optimization problems. *Information and Control*, pages 1–27, 1978.

[21] T. Ibaraki. Generalization of alpha-beta and SSS* search procedures. *Artificial Intelligence*, 29:73–117, 1986.

[22] L. Jaulin, M. Kieffer, O. Didrit, and E. Walter. *Applied Interval Analysis, with Examples in Parameter and State Estimation, Robust Control and Robotics*. Springer, Berlin, 2001.

[23] L. Jaulin, M. Kieffer, O. Didrit, and E. Walter. *Dealing with Quantifiers*, chapter 5.6.3, pages 126–134. In [22], 2001.

[24] L. Jaulin, M. Kieffer, O. Didrit, and E. Walter. *Minimax Optimization*, chapter 5.6, pages 133–134. In [22], 2001.

[25] R. B. Kearfott. Interval extensions of non-smooth functions for global optimization and nonlinear systems solvers. *Computing*, 57(2):149–162, 1996.

[26] R. B. Kearfott. *Rigorous Global Search: Continuous Problems*. Kluwer Academic Publishers, 1996.

[27] R. E. Korf. Space-efficient search algorithms. *Computing Surveys*, 27(3):337–339, 1995.

[28] R. E. Korf and D. M. Chickering. Best-first minimax search. *Artificial Intelligence*, 84(1–2):299–337, 1996.

[29] V. Kumar and L. Kanal. A general branch and bound formulation for understanding and synthesizing and/or tree search procedures. In J. Pearl, editor, *Search and Heuristics*. North-Holland, 1983.

[30] V. Kumar, D. S. Nau, and L. N. Kanal. A general branch-and-bound formulation for AND/OR graph and game tree search. In L. Kanal and V. Kumar, editors, *Search in Artificial Intelligence*, pages 91–130. Springer, 1988.

[31] E. L. Lawler and D. E. Wood. Branch-and-bound methods: A survey. *Operations Research*, 14:699–719, 1966.

[32] A. Mahanti and A. Bagchi. AND/OR graph heuristic search methods. *Journal of the ACM*, 32:28–51, 1985.

[33] S. McCallum. Solving polynomial strict inequalities using cylindrical algebraic decomposition. *The Computer Journal*, 36(5):432–438, 1993.

[34] L. G. Mitten. Branch-and-bound methods: General formulation and properties. *Operations Research*, 18:24–34, 1970. Errata in Vol. 19, p. 550.

[35] R. E. Moore. *Interval Analysis*. Prentice Hall, Englewood Cliffs, NJ, 1966.

[36] R. E. Moore and H. Ratschek. Inclusion functions and global optimization II. *Mathematical Programming*, 41:341–356, 1988.

[37] N. J. Nilsson. *Principles of Artificial Intelligence*. Springer, Berlin, 1982.

[38] A. J. Palay. The $B*$ tree search algorithm: New results. *Artificial Intelligence*, 19(2):145–163, 1982.

[39] A. J. Palay. *Searching with Probabilities*. Pitman, London, 1985.

[40] J. Pearl. *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Addison-Wesley, 1984.

[41] A. Plaat, J. Schaeffer, W. Pijls, and A. de Bruin. Best-first fixed-depth minimax algorithms. *Artificial Intelligence*, 87:255–293, 1996.

[42] J.-F. Puget and P. V. Hentenryck. A constraint satisfaction approach to a circuit design problem. *Journal of Global Optimization*, 13:75–93, 1998.

[43] S. Ratschan. Convergence of quantified constraint solving by approximate quantifiers. Technical Report 00-23, Research Institute for Symbolic Computation (RISC) - Linz, 2000. Submitted for Publication.

[44] S. Ratschan. Quantified constraints under perturbations. Technical Report 00-36, Research Institute for Symbolic Computation (RISC) - Linz, 2000. Submitted for Publication.

[45] S. Ratschan. Approximate quantified constraint solving by cylindrical box decomposition. *Reliable Computing*, 8(1):21–42, 2002.

[46] S. Ratschan. Continuous first-order constraint satisfaction. In J. Calmet, B. Benhamou, O. Caprotti, L. Henocque, and V. Sorge, editors, *Artificial Intelligence, Automated Reasoning, and Symbolic Computation*, number 2385 in LNCS, pages 181–195. Springer, 2002.

[47] Ratschek and Rokne. Experiments using interval analysis for solving a circuit design problem. *Journal of Global Optimization*, 3(4):501–518, 1993.

[48] H. Ratschek and J. Rokne. *New Computer Methods for Global Optimizations*. Ellis Horwood Limited, Chichester, UK, 1988.

[49] D. Ratz. *Automatische Ergebnisverifikation bei globalen Optimierungsproblemen*. PhD thesis, Universität Karlsruhe, 1992.

[50] A. Reinefeld and P. Ridinger. Time-efficient state space search. *Artificial Intelligence*, 71(2):397–408, 1994.

[51] D. A. Schmidt. *Denotational Semantics: A Methodology for Language Development*. W.C. Brown, 1988.

[52] S. P. Shary. Algebraic approach to the interval linear static identification, tolerance, and control problems, or one more application of Kaucher arithmetic. *Reliable Computing*, 2(1):3–33, 1996.

[53] Z. Shen, A. Neumaier, and M. C. Eiermann. Solving minimax problems by interval methods. *BIT*, 30:742–751, 1990.

[54] B. S. Stewart, C.-F. Liaw, and C. C. W. III. A bibliography of heuristic search research through 1992. *IEEE Transactions on Systems, Man, and Cybernetics*, 24(2), 1994.

[55] G. C. Stockman. A minimax algorithm better than alpha-beta? *Artificial Intelligence*, 12:179–196, 1979.

[56] B. W. Wah and C. F. Yu. Stochastic modeling of branch-and-bound algorithms with best-first search. *IEEE Trans. Software Eng.*, SE-11:922–934, 1985.