

Information-theoretic approaches to branching in search*

Andrew Gilpin
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213
gilpin@cs.cmu.edu

Tuomas Sandholm
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213
sandholm@cs.cmu.edu

ABSTRACT

Deciding what to branch on at each node is a key element of search algorithms. We present four families of methods for selecting what question to branch on. They are all information-theoretically motivated to *reduce uncertainty in remaining subproblems*. In the first family, a good variable to branch on is selected based on lookahead. In real-world procurement optimization, this entropic branching method outperforms default CPLEX and strong branching. The second family combines this idea with strong branching. The third family does not use lookahead, but instead exploits features of the underlying structure of the problem. Experiments show that this family significantly outperforms the state-of-the-art branching strategy when the problem includes indicator variables as the key driver of complexity. The fourth family is about branching using carefully constructed linear inequality constraints over *sets* of variables.

Categories and Subject Descriptors

I.2.8 [Artificial Intelligence]: Problem Solving, Control Methods, and Search

General Terms

Algorithms

Keywords

search, branching heuristics, entropic branching, mixed-integer programming

1. INTRODUCTION

Search is a fundamental technique for problem solving in AI and operations research (OR). At a node of the search tree, the search algorithm poses a question, and then tries out the different answers (which correspond to the branches emanating from that node). Many different ways of deciding which question to pose (branching strategies) have been studied. In this paper we introduce a new paradigm for developing branching strategies, employing information theory

*This work was funded by, and conducted at, Com-
bineNet, Inc., Fifteen 27th St., Pittsburgh, PA 15222.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...\$5.00.

as the principle that guides the development. In the context of solving integer programs, we develop four high-level families of strategies, and show that some of those families significantly improve performance over existing strategies. All four strategies aim to reduce the uncertainty in the current subtree.

The rest of this section introduces the necessary background material. Each of the following four sections introduces one high-level family of new strategies.

1.1 Integer programming

One of the most important computational problems in CS and OR is integer programming. Applications of integer programming include scheduling, routing, VLSI circuit design, and facility location [4]. Integer programming is the problem of optimizing a linear function subject to linear constraints and integrality constraints on some of the variables. While (the decision version of) MIP is $\mathcal{NP}$ -complete, there are many sophisticated techniques that can solve very large instances in practice.

1.2 Strong branching

At every node of a tree search, the search algorithm has to decide what question to branch on (and thus what the children of the node will be). One commonly used method is to branch on the variable whose value is furthest from being integral [6]. In other words, the question is: “What should the value of this variable be?”, and the children correspond to different answers to this question. For many problem instances, this approach is effective.

A more sophisticated approach, which is better suited for certain hard problem instances, is *strong branching* [1]. The algorithm performs a one-step lookahead for each variable that is non-integral in the LP at the node. The one-step lookahead computations solve the LP relaxation for each of the children.

2. ENTROPIC LOOKAHEAD FOR VARIABLE SELECTION

The main idea behind our approaches is to treat the fractional portion of integer-constrained variables as probabilities, indicating the probability with which we expect the variable to be greater than its current value in the optimal solution. Using this reasoning, our branching heuristic is designed to guide the search so as to remove uncertainty from the partial solution in the current search node. For any search algorithm, the beginning of the search is where there is the most uncertainty about the optimal solution to

the problem. When the search is complete, there is zero uncertainty as the optimal solution has been found. Our branching heuristic is designed to direct the search from the root node (with high uncertainty) to the optimal node (with zero uncertainty) in as direct a way as possible. We determine the variable to branch on using one-step lookahead as in strong branching. The difference is that instead of examining the objective values of potential child nodes, we examine the remaining *uncertainty* in potential child nodes, choosing a variable to branch on that yields children with the least uncertainty.

While there has been much work on developing branching heuristics, to our knowledge this is the first work that takes an information-theoretic approach to guiding the search process.

Before we describe our approach to branching in search, it will be useful to define how we can quantify the “uncertainty” of a partial solution. For this, we borrow some definitions from information theory, from which the primary contribution is the notion of *entropy*, which measures the amount of uncertainty in a random event. Given an event with two outcomes (say 0 or 1) where outcome 0 occurs with probability x , we can compute the entropy of the event from the probability of each outcome occurring:

$$e(x) = \begin{cases} -x \log_2 x - (1-x) \log_2 (1-x) & : 0 < x < 1 \\ 0 & : x \in \{0, 1\}. \end{cases}$$

Now we are ready to present our variable selection method:

ALGORITHM 1. Entropic Branching (EB)

1. $\text{candidates} \leftarrow \{i \mid x_i \text{ fractional}\}$
2. *For each* $i \in \text{candidates}$:
 - (a) $x_f \leftarrow x_i - \lfloor x_i \rfloor$
 - (b) Let $\hat{x}^l$ be solution vector of LP with $\hat{x}_i^l \leq \lfloor x_i \rfloor$.
 - (c) Let $\hat{x}^u$ be solution vector of LP with $\hat{x}_i^u \geq \lceil x_i \rceil$.
 - (d) $\text{entropy}(x_i) \leftarrow \sum_{j=1}^n (1 - x_f) e(\hat{x}_j^l) + x_f e(\hat{x}_j^u)$
3. $i^* \leftarrow \text{argmin}_{i \in \text{candidates}} \text{entropy}(x_i)$
4. *Return* i^*

EB is a *general* method, usable on any MIP as it does not make any assumptions about the underlying model for the problem on which it is used. EB (and SB) could be modified to perform more than one-step lookahead in a straightforward way. However, we suspect that this increased computational effort would not reduce tree size enough to be beneficial.

For illustrative purposes, there is an interesting (though not exact) analogy between our entropic lookahead method for question selection at search nodes and algorithms for decision tree induction. In most recursive decision tree induction algorithms, a question is inserted at a leaf that results in the greatest information gain. Similarly, in search, by choosing questions whose children have the least entropy, we are creating children that in a sense also result in the greatest information gain.

3. COMBINING SB AND EB

From the descriptions, it is clear that SB and EB are computed quite similarly. A natural question arises: can we develop a hybrid approach combining the strengths of both without using significantly more computational resources? In this section we answer this question in the affirmative

by introducing a second family of variable selection strategies. In this family, SB and EB are hybridized in different ways. Each of these methods requires only a small amount of additional computation compared to performing *only* SB or EB (because the same lookahead with the same child LP solves is used). We classify our hybrid approaches into two categories: tie-breaking and combinational.

3.1 Tie-breaking methods

In this approach, we first perform the SB computations, but instead of simply branching on the variable with best score, we break ties using an entropy computation. Since we have already computed the LP relaxations for each branch, computing the entropy is a negligible computational cost (relative to computing the LP relaxations). In addition to breaking exact ties, we can also consider two variables having SB scores within $x\%$ of each other as tied. For a given percentage x , we refer to this method as TB($x\%$).

3.2 Combinational methods

We present two methods of combining information from SB and EB in order to compute a single score for a variable. The variable with the best score will then be branched on.

The first method, RANK, performs the computation for SB and EB first. (Again, these computations are performed simultaneously at little additional cost beyond doing either SB or EB alone.) Define $\text{rank}_{SB}(x_i)$ to be the rank of variable x_i in terms of its SB score (*i.e.*, the variable with the largest score would have rank 1, the variable with the second-largest score would have rank 2, and so on). Similarly, define $\text{rank}_{EB}(x_i)$ to be the rank of variable x_i in terms of its EB entropy. Then, for each variable, we let $\text{rank}(x_i) = \text{rank}_{SB}(x_i) + \text{rank}_{EB}(x_i)$ and choose the variable x_i with the smallest rank.

The second method, COMB($\alpha, 1 - \alpha$), computes a convex combination of the SB score (with weight α) and the current entropy minus the EB score (with weight $1 - \alpha$). It then selects the variable with the highest final score.

3.3 Experiments on these lookahead-based branching strategies

We conducted a host of experiments with the search methods described above, both on MIPLIB and real-world procurement optimization instances. To briefly summarize, EB and SB performed comparably on the MIPLIB data, while EB was 29.5% faster on the real-world procurement optimization instances. All of our experimental results are available in a separate technical report [3].

4. ENTROPIC LOOKAHEAD-FREE VARIABLE SELECTION

In this section we introduce a third family of branching strategies, again within the entropy-based branching paradigm. This method is computationally less expensive than the methods we have presented thus far because it does not use lookahead. However, it does require an advanced knowledge of the structure of the problem.

The problem with which we experiment is motivated by a real-world electronic commerce application: a combinatorial procurement auction (aka. reverse auction) where the buyer specifies the maximum number of winning suppliers [2, 5].

DEFINITION 1. (*Combinatorial procurement auction with*

maximum winners constraint)

Let $\mathcal{M} = \{1, \dots, m\}$ be the m goods that the buyer wishes to procure (the buyer wants at least one unit of each good). Let $\mathcal{S} = \{1, \dots, s\}$ be the s suppliers participating in the procurement and let $\mathcal{B} = \{B_1, \dots, B_n\}$ be the bids, where $B_i = \langle G_i, s_i, p_i \rangle$ indicates that supplier s_i can supply the bundle of goods $G_i \subseteq \mathcal{M}$ at price p_i . Finally, the buyer indicates the maximum number of winners, k . The winner determination problem is to identify the winning bids so as to minimize the buyer's cost subject to the constraints that the buyer's demand is satisfied and that the maximum number of winners constraint is satisfied.

We can very naturally formulate the above generalized winner determination problem as a MIP:

$$\begin{aligned} & \text{minimize} && \sum_{i=1}^n p_i x_i \\ & \text{such that} && \sum_{i|j \in G_i} x_i \geq 1 && j \in \{1, \dots, m\} \\ & && \sum_{i|s_i=j} x_i - m y_j \leq 0 && j \in \{1, \dots, s\} \\ & && \sum_{j=1}^s y_j - k \leq 0 \\ & && x_i \in \{0, 1\} && i \in \{1, \dots, n\} \\ & && y_j \in \{0, 1\} && j \in \{1, \dots, s\} \end{aligned}$$

This problem is $\mathcal{NP}$ -complete, even if the bids are on single items only [5]. The formulation is typical of a common class of problems in which binary “indicator” variables—the y_j variables in the formulation above—are used to model logical connectives [4] (for example an implication that if an indicator is zero, then each of a set of other variables has to be zero). Constraints that state that at most (or exactly) k variables from a set of variables can be nonzero are an important special case. Typically, the LP relaxation gives poor approximate values for the indicator variables, and as we show, the branching method that we propose helps significantly to address this problem.

4.1 Branching strategy

The hardness in this problem comes primarily from determining the winning set of suppliers. In terms of the above MIP, we need to determine the y_j values. The main idea in our branching strategy for this problem is to branch on y_j values that correspond to suppliers about which we are most uncertain. But rather than deriving this uncertainty from the variable y_j (for which the LP gives very inaccurate values), we derive it from the variables corresponding to supplier j 's bids. The branching strategy works as follows. For each supplier j where $y_j \notin \{0, 1\}$, compute

$$\text{entropy}(j) = \sum_{i|s_i=j} e(x_i)$$

and branch on the variable $y_{j'}$ where

$$j' = \operatorname{argmin}_j \text{entropy}(j).$$

This strategy does not use lookahead: it only uses the LP values of the current search node. We call this branching strategy *Indicator Entropic Branching (IEB)*. Our experimental results (available in a longer version of this paper [3]) for this problem indicate that IEB significantly outperforms the state-of-the-art branching strategy.

5. ENTROPIC LOOKAHEAD FOR MULTI-VARIABLE BRANCHES

In this section we introduce a fourth family of methods for determining good questions to branch on. In EB, we performed a one-step lookahead for each non-integral variable. Here, we generalize entropic lookahead beyond branching on variables. It is known that in integer programming one can branch on the sum of the values of a set of variables. For example, if the LP relaxation at the current search node has $x_i = 0.2$ and $x_j = 0.6$, we could set one branch to be $x_i + x_j \leq 0$ and the other branch to be $x_i + x_j \geq 1$. In general, given a set $\mathcal{X}$ of variables and the current LP relaxation solution $\hat{x}$, we can let $k = \lfloor \sum_{i \in \mathcal{X}} \hat{x}_i \rfloor$ and we can generate the following two branches:

$$\sum_{i \in \mathcal{X}} x_i \leq k \quad \text{and} \quad \sum_{i \in \mathcal{X}} x_i \geq k + 1.$$

No other value of k is worth considering; any other integer value would cause one child to be exactly the same as the node. Then, instead of branching on the variable with the smallest amount of entropy in its child nodes, we select the set of variables for branching that results in the smallest amount of entropy in the two child nodes. In step 2.d of EB, we weighted the entropy of each child by the probability that we expect the optimal solution to occur in each child. In the multi-variable case, we still perform this weighting, but it is more complicated since the probability of each branch depends on several variables.

Preliminary experimental results indicate that this branching scheme is successful at reducing the size of the search tree, but the extra time at each node makes it not worth it.

6. CONCLUSIONS

We introduced a new paradigm for branch selection in search based on an information-theoretic approach. In the beginning of a search, there is the most uncertainty about the optimal solution. When the search is complete, there is zero uncertainty. Using this observation, we developed four families of methods for selecting what question to branch on at a search node so as to reduce the amount of uncertainty in the search process. All four of our families of methods are information-theoretically motivated to reduce remaining entropy.

7. REFERENCES

- [1] David Applegate, Robert Bixby, Vasek Chvátal, and William Cook. The traveling salesman problem. Technical report, Rutgers University, DIMACS, 1994.
- [2] Andrew J Davenport and Jayant Kalagnanam. Price negotiations for procurement of direct inputs. Technical Report RC 22078, IBM, May 2001.
- [3] Andrew Gilpin and Tuomas Sandholm. Information-theoretic approaches to branching in search. Technical Report CMU-CS-06-102, 2006.
- [4] George Nemhauser and Laurence Wolsey. *Integer and Combinatorial Optimization*. John Wiley & Sons, 1999.
- [5] Tuomas Sandholm and Subhash Suri. Side constraints and non-price attributes in markets. In *IJCAI-2001 Workshop on Distributed Constraint Reasoning*, pages 55–61, Seattle, WA, 2001. To appear in *GEB*.
- [6] Laurence Wolsey. *Integer Programming*. John Wiley & Sons, 1998.