Report Number: WUCS-91-04

1991-01-01

Completely Reliable Auto Shopping Heuristic (CRASH)

Authors: Todd Bashuk, David Donat, and Kieth Marrs

The major goal of the Completely Reliable Auto Shopping Heuristic (CRASH) system is to simulate a car buying marketplace using belief networks as the inference engine. This paper describes the design, knowledge acquisition, and implementation stages of the CRASH system. The paper also describes the additional work needed to implement CRASH as commercially viable product.

COMPLETELY RELIABLE AUTO SHOPPING HEURISTIC (CRASH)

Todd Bashuk, David Donat and Kieth Marrs

WUCS-91-04

January 1991

Department of Computer Science
Washington University
Campus Box 1045
One Brookings Drive
Saint Louis, MO 63130-4899

Completely Reliable Auto Shopping Heuristic (CRASH)

Todd Bashuk, David Donat, Kieth Marrs

Abstract: The major goal of the Completely Reliable Auto Shopping Heuristic (CRASH) system is to simulate a car buying marketplace using belief networks as the inference engine. This paper descibes the design, knowledge acquisition, and implementation stages of the CRASH system. The paper also describes the additional work needed to implement CRASH as a commercially viable. product.

Introduction: The objective of the Completely Reliable Auto Shopping Heuristic (CRASH) system is to simulate a car buying marketplace. Crash uses belief networks to match potential buyers with advertisers. Matching is a three step process. In the first step, the CRASH system picks the most likely ad based on the probabilities in the belief network. The user decides whether he likes, dislikes, or is ambivalent about the ad. Based on the user's response, the network is updated with either positive or negative evidence. The probabilities in the network are then recalculated, and the system repeats the pick-update-calculate cycle until the user quits.

A belief network is a type of Artificial Intelligence (AI) system which manages uncertainty well. Traditional expert system applications use rules and assertions and handle uncertainity only in an ad hoc manner. Belief networks provide a formal method based on probability theory for reasoning under uncertainity.

Belief networks represent knowledge as a collection of probabilistic variables with causal relationships. They are often illustrated as graphs where each node represents a variable and each edge or connection represents a relationship. The edges are directed, implying a causality between the two connected variables such that an arrow points toward the affected variable. Associated with each node is a conditional probability matrix defining the relationships of this node with the other nodes that affect it.

The nodes in the CRASH belief network represent attributes of cars. A node with a high probability indicates a high liklihood that the node is important to the buyer. A low probability does not mean that the buyer does not want the attribute, but does not deem the attribute as important.

The advantage of belief networks over expert systems is that a single belief node can encompass multiple rules. For example, if the MPG node in the CRASH network is high, the user probably wants a car that gets good gas mileage. If MPG is low, the buyer doesn't care if the car gets low gas mileage. Furthermore, there are many factors that contribute to the MPG of a car (size of engine, horsepower etc.) which can all be grouped into a single node, "Powerful". The number of rules needed to implement a belief network the size of CRASH would be quite substantial, an order of magnitude greater than the 25 nodes in the CRASH network.

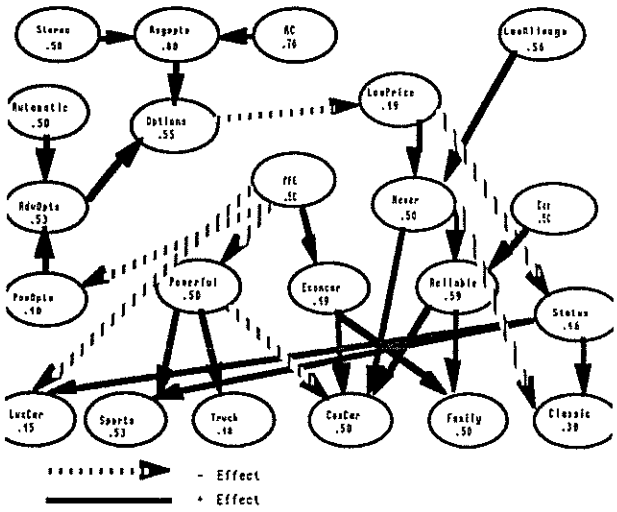
CRASH is a testbed which implements different pick-ad heuristics and network update methods. This paper describes the algorithms and theory used to implement the pick-update-calculate cycle in CRASH.

Knowledge Acquisition: CRASH's network was built in a four step process: identifying the variables, identifying the linkage between variables, estimating the probabilities, and refining the probabilities The variables of the network were determined by making a list of

common car traits and refining the list by reading car advertisements. The relationships between the variables were drawn as edges with a +/- to indicate how the parent effects its child. The resulting diagram is referred to as the variable relationship diagram. Because of the implementation of the belief network algorithm and the difficulty in visualizing a large set of probabilities, certain rules were used on our design. Foremost of the rules are that two is the optimal number of parents for each node. Except for the "top-level" nodes, each node is going to have parent nodes from which it gets probabilistic information. A node with only one parent can be indicative of a redundant node; the node could be eliminated, and the parent's information could be passed down directly. A two parent maximum was motivated because of the intuitive understanding of the effects of two variables (4 different probabilities) versus more variables (2^n probabilities where n is the number of variables).

The second rule is that the structure of the network should essentially be "vertical". Links between nodes can be thought of as links from one vertical level of the network to another. Because these are the medium of probability propagation, information is passed much more directly and freely from level to level (vertically) than between nodes on a single level (horizontally). Evidence that is introduced into a particular node will not have as much effect on its siblings as it has on its parents/children. Any dynamic effects desired by the designer will only be achieved in a vertical tree structure.

The next step was to establish the probabilities between the nodes. The project team used a series of rules for defining these probabilities. The rules are as follows: Strong positive relationship: .9, weak positive relationship: .6, strong negative relationship: .1, weak negative relationship: .4 These probabilities were then refined based on experimentation and testing of the network. The final variable relationship diagram with probabilities is shown below.



The specific numbers associated with the links, although less intuitive, are also less important than the relative positive/negative influences. The evidence inserted into the network will emphasize the relationships and de-emphasize the exact probabilities. Uncertain reasoning, an essential portion of a problem where so little information is concrete, is inherent and natural in the algorithm.

Implementation: CRASH uses the belief network as the inference engine to heuristically pick the best ad to present to the user. The algorithms used in CRASH can be divided into three distinct phases: inserting evidence into the network, recalculating the network, and picking the next ad. As mentioned earlier, this is the pick-update-calculate cycle.

The CRASH implementation environment uses a Macintosh II running Allegro Common Lisp version 1.3.2. The L&S algorithm and belief network browser used in CRASH were developed by the Medical Informatics Lab at Washington University. [Cousins, 1990].

The design of CRASH was divided into four phases. First was the design of an ad database to store information about car ads. Second was the design of the algorithms to insert evidence based on whether a user likes or dislikes a particualar car ad. Third was the

design of algorithms to select the best ad based on the probabilities in the network. And fourth was the design of the user interface. The following sections describe these four design phases. The architecture of crash was built modularly allowing the development of different methods for inserting evidence into the network and for picking ad. The testing portion of the project determined the best combination of methods.

Database: The ads in the CRASH system are stored in the advertisement database. The advertisement database serves two functions in this program. In one sense it is our interface with the user, the medium we use to collect information about the user's likes and dislikes.

The database currently contains a series of written advertisements for a wide array of makes and models of cars. The year, price, and description of the car in the advertisement is extracted and listed as seperate fields.No alterations or clarifications were made to the text; as it is necessary to preserve the integrity and "feel" of the original advertisement in order to get meaningful information from the user. Finally, each advertisement has a list of nodes from our belief network associated with it, the nodes to which the information in the ad is relevant. Each node listing also has a "weight factor", to facilitate weighting how much each ad affects a particular node. At present all ads were hand-indexed.

Inserting Evidence: Evidence is assigned to the nodes in the network in two ways). To get initial evidence into the network, the user is requested to complete a questionnaire about his preferences in basic car features. Each feature in the questionnaire corresponds to a node in the network. The user rates all, some, or none of these features by assigning a number from one to ten to the feature. Based on these ratings, positive or negative evidence is assigned proportionally to the appropriate nodes. The user is not required to fill out the questionnaire; it is not a necessary component of the system, but is only a tool to optimize performance. For each car ad, the user indicates whether he/she likes the car by selecting "yes", "no", or "maybe". Each ad, as described above, identifies which nodes of the network should be updated and by how much. Therefore, when the user responds to a given ad, the evidence of each node associated with that ad (i.e., ad node) is updated by an appropriate amount based on the weighted additon method (as described below). If the user selects "maybe", no updating is performed.

After updating the evidence for an ad, the evidence is propagated through the network producing new node probabilities. Then, based on these new probabilities, a new series of car ads are selected and presented to the user. With each ad presented and rated by the user, the network will move closer to the configuration desired by the user, resulting in more and more "yes" responses from the user.

Select Ad: To select an ad to show the user, CRASH ranks each ad against the current state of the network and chooses one of the top ranked ads that has not already been presented. Many algorithms were tested to rank the ads. The algorithm that gives the best results ranks each ad by comparing the current state of the network with the what the state of the network would be if the "like" evidence of only this ad was inserted. The concept of the this method is to compare the current probabilities of the network to the ideal probabilities for each ad, and to select the ad whose ideal is closest to the current state. To determine the ideal probabilities, each ad is independently "compiled" against the initial state of the network before any other processing is performed. The probabilities of the resulting network are then saved with the given ad. To "compile" an ad, the evidence of the nodes associated with that ad is updated as if a user had said he/she liked the car. The evidence is then propagated through the network producing new probabilities which are based only on this particular ad and the initial state. After all this preprocessing, Crash can begin to select ads. A regression algorithm is used to determine which ad has the "closest fit" to the current state of the network; this ad is to be the

best choice. The method used only saves the probabilities of a specialized set of nodes called pick nodes. An example of a pick node is "Compact Car", "Luxury Car", or "Sports Car". Crash only compares the states of the six pick-nodes instead of every node in the network.

User Interface: The user interface used in CRASH is a simplified implementation of an optimal interface. An ideal interface would include a multi-media display of the desired ad. The interface of CRASH is entirely text based.

When the system is initialized, the user is asked to fill out a questionnaire rating particular attributes of cars. The six most important or key attributes were included on the questionnaire. The user is asked to rate the level of importance of each attribute. The attributes selected are: regular options, gas mileage, price, powerful, age, and status

All values on the questionnaire are defaulted to 5 (or ambivalent). Once the questionnaire is completed the evidence is inserted into the network. Then the user is presented with the six most likely ads based on the resulting probabilities of the network. The user is given the options of like/dislike/ambivalent about each ad. Based on the user's response, the evidence included in the ads is then inserted into the network.

The buyer continues to rate ads until he/she has seen enough. The buyer has three options: continue, quit, or top ten. The "Continue" button calculates the next screen full of ads, allowing the user to continue the picking process. The "Top Ten" calculates the 10 ads in the system with the highest rating, and continues the picking process. The "Quit" button presents the user the list of best ads and exits the system.

Looking toward the future: Presently we hand index the ads, but his would be imparctical on a commerical scale. The obvious solution would be a parser for ads that would extract the relevant informatiation. However building the parser would be a huge effort because of the natural language understanding needed. An alternative to an ad parsing system is (instead of parsing existing ads that are in the classified ads) to create an Ad Design System (ADS). The ADS system would prompt the user for information about the car such as make, model etc. From the answers an expert system would determine what nodes in the network the ad should update. For example if the car is a Porsche 911, the expert system would know to update sports car and status positively, and reliability negatively. The ADS system would guarantee consistency in the ad indexing process. Because of the lack of consistency in the hand indexing process, the accuracy of the CRASH system is limited. The ADS system would enhance the reliability of the results of the CRASH system. The seller would also provide a written description of the car similar to the text they would write for the classified ad in the newspaper.

Another possibility for future work would be to increase the system's knowledge of cars. With larger databases of advertisements, a finer granularity of "types" of cars would be required. The additional knowledge would embody greater detail about cars potentially all the way down to differentiating models. This knowledge base of car types would require a significant amount of maintenance because car models are continually changing.

The current prototype does not achieve the sub-second response time desired in a commercial system. A commerical implementatation will want to use a more efficient implementation of Lauritzen or other belief network update algorithims.

When CRASH is scaled up to commercial proportions it will act as an ad filter which presents the buyer with a personlized classified ads.

Acknowledgements and Bibliography: We would like to thank Mark Frisse, Michael Kahn, and Steve Cousins from the Medical Informatics Laboratory for the belief network software and browser. They also helped

further our understanding of the design and theory of belief networks.Crash was implemented in a 1 semester class in an advanced Artificial Intelligence course at Washington University (CS513).

Cousins S. B., Silverstein, J. C., Frisse, M.E., "Query Networks for Medical Information Retrieval - Defining Probabilistic Relationships", Proceedings of 14th anuual symposium on Computer Applications in Medical Care, 1990. pp 800-804.

Lauritzen, S. L., and D. J. Spiegelhalter [1988], "Local Computation with Probabilities in Graphical Structures and Their Applications to Expert Systems," Journal of Royal Statistical Society B, Vol. 50, No 2.

Morawski, Paul. "Programming Bayesian Belief Networks" AI Expert, August 1989, pp. 74 - 79

Morawski, Paul. "Understanding Bayesian Belief Networks," AI Expert, May 1989, pp. 44-48

Neapolitan, Richard E, Probabilistic Reasoning in Expert System: Theory and Algorithms, 1990 , John Wiley & Sons, Inc.

Quimby, David, Knowledge Based Systems-Belief Networks, SRI International, TechLink, October 1990

Pearl, Judea, Probabilistic Reasoning in Intelligent Systems, Morgan Kaufmann, San Mateo, California, 1988

Steele, Guy L., Common Lisp, Digital Press ,1984