VT: An Expert Elevator Designer That Uses Knowledge-Based Backtracking

Sandra Marcus, Jeffrey Stout, John McDermott

VT (vertical transportation) is an expert system for handling the design of elevator systems that is currently in use at Westinghouse Elevator Company. Although VT tries to postpone each decision in creating a design until all information that constrains the decision is known, for many decisions this postponement is not possible. In these cases, VT uses the strategy of constructing a plausible approximation and successively refining it. VT uses domain-specific knowledge to guide its backtracking search for successful refinements. The VT architecture provides the basis for a knowledge representation that is used by SALT, an automated knowledge-acquisition tool. SALT was used to build VT and provides an analysis of VT's knowledge base to assess its potential for convergence on a solution.

Due to software problems at the typesetter, the publication of this article in volume 8, number 4 was flawed. A corrected copy is reprinted here. –Ed.

In some cases, plausible guessing combined with the ability to backtrack to undo a bad guess can be the most efficient way to solve a problem (Stefik et al. 1983). Even least commitment systems such as MOLGEN (Stefik 1981a, 1981b) are sometimes forced to guess. In the course of designing genetics experiments, MOLGEN tries to avoid making a decision until all constraints that might affect the decision are known. In some cases, this postponement is not possible, and the system becomes stuck; none of the pending decisions can be made with complete confidence. In such a case, a decision based on partial information is needed, and such a decision might be wrong. In this case, a problem solver needs the ability either to backtrack to correct bad decisions or to maintain parallel solutions corresponding to the alternatives at the stuck decision point. However, if alternative guesses exist at each point, and there are many such decision points on each solution path, a commitment to examine every possible combination of alternatives proves unwieldy. Such complexity exists in the VT task domain.

VT performs the engineering task of designing elevator systems. It must use the customer's functional specifications to select equipment and produce a parts configuration that meets these specifications as well as safety, installation, and maintenance requirements. Because of the large number of potential part combinations and the need for customizing the layout to the space available in individual buildings, VT must construct a solution. Like MOLGEN, VT tries to order its decisions so that they are made only when all relevant constraints are known; it guesses only when stuck. Unlike MOLGEN, VT's decisions about part selection and placement are so interdependent that plausible reasoning (guessing) is a major feature of its search for a solution. Thus, VT's problem-solving strategy is predominantly one of constructing an approximation and successively refining it.

Systems that use plausible reasoning must be able to identify bad guesses and improve on these decisions in a way which helps converge on a solution. VT is similar to AIR/CYL (Brown 1985) and PRIDE (Mittal and Araya 1986) in that it uses a knowledge-based approach to direct this search; that is, it uses domain-specific knowledge to decide what past decisions to alter and how to alter them. This approach contrasts with EL (Sussman 1977; Stallman and Sussman 1977), an expert system which shares many architectural features with VT but which uses domain-independent strategies to limit the search during the backtracking phase. As with EL, the VT architecture makes clear the role that domain-specific knowledge plays in the system and the interconnections among decisions used to construct and refine a solution. This architecture provides the basis for VT's explanation facility, which is similar to that of EL and the related CONSTRAINTS language (Sussman and Steele 1980), with some extensions. We have exploited the structure provided by this architecture even further by using it to manage VT's knowledge acquisition.

VT's architecture provides structure for a representation of its domain-specific knowledge that reflects the function of the knowledge in problem solving. This representation serves as the basis for an automated knowledge-acquisition tool, SALT (Marcus,

Welcome to VT —- The Elevator Design Expert System

1. INPUT	**Enter contract information**
2. RUN	**Process the input data**
3. SHOW	**Display output information**
4. EXPLAIN	**Explain the results of a run**
5. SAVE	**Save data for the current contract**
6. EXIT	**End this session with VT**

Enter your command [INPUT]: <*cr*>

Figure 1. VT's Top Level Menu.

INPUT GD DUTY	**GR 24364**	**ADMINISTRATION CENTER**
Car:1		
1.	**Type of loading**	**PASSENGER**
2.	**Machine**	**GEARED**
3.	**Machine location**	**OVERHEAD**
4.	**Power supply**	**208-3-60**
5.	**Capacity**	**3000**
6.	**Speed**	**250**
7.	**Travel**	**729**
8.	**Platform width**	**70**
9.	**Platform depth**	**84**
10.	**Counterweight location**	**REAR**
11.	**Counterweight safety**	**NO**
12	**Compensation specified**	**NO**
Action [EXIT]:		

Figure 2. Completed Sample Input Screen.

McDermott, and Wang 1985; Marcus and McDermott 1986, Stout et al. 1987), which has been used to build VT. SALT elicits from experts all the knowledge VT needs in order to design elevators and represents that knowledge in a way which enables VT's problem-solving method to use it. SALT's knowledge representation can also be used to assess the adequacy of the knowledge base for convergence on a solution.

The next section, "What VT Does," presents VT mainly from a user's point of view. "The VT Architecture" describes the VT architecture in detail, with respect to problem-solving, explanation, and knowledge acquisition. "Management of Knowledge-Based Backtracking" describes how SALT's knowledge base analysis supports VT's domain-dependent backtracking. "Comparison to Other Constructive Systems" compares VT to other expert systems that perform design, planning, or scheduling tasks. "VT's Performance" reports some of VT's performance characteristics.

What VT Does

VT is used by Westinghouse Elevator engineers to design elevator systems to customer specifications. VT has enough domain knowledge to perform the design task unaided. VT also has an interactive capability which allows a user to directly influence its decisions.

The Engineer's Task

Westinghouse Elevator design experts receive data collected from several contract documents. These data are transmitted to the engineering operation by the regional sales and installation offices. Three main sources of information exist: (1) customer requirement forms describing the general performance specifications, such as carrying capacity and speed of travel, and some product selections, such as the style of light fixture in the cab; (2) the architectural and structural drawings of the building, indicating such elements as wall-to-wall dimensions in the elevator shaft (hoistway) and locations of rail supports; and (3) the architectural design drawings of the elevator cabs, entrances, and fixtures. Because all this information is not necessarily available at the start of a contract, the engineer must sometimes produce reasonable guesses for incomplete, inconsistent, or uncertain data to enable order processing to tentatively proceed until customer verification is received. (These guesses are in addition to whatever guesses might be required during a problem-solving episode based on these data.)

Given this information, experts attempt to optimally select the equipment necessary and design its layout in the hoistway to meet engineering, safety code, and system performance requirements. This task is a highly constrained one. A completed elevator system must satisfy constraints such as the following: (1) there must be at least an 8-inch clearance between the side of the platform and a hoistway wall and at least 7 inches between the platform side and a rail separating two cars; (2) a model 18 machine can only be used with a 15, 20, or 25 horsepower motor; and (3) the counterweight must be close enough to the platform to provide adequate traction but far enough away to prevent collision with either the platform or the rear hoistway wall (by an amount dependent on the distance of travel).

The design task also encompasses the calculation of the building load data required by the building's structural engineers, the reporting of the engineering and ordering data required for the field installation department and regional safety code authorities, and the reporting of the mechanical manufacturing order information.

A Quick Look at VT in Action

VT is comprised of several distinct parts, described briefly in the sample interactions which follow. VT prompts appear in boldface. User replies appear in bold italics.

Figure 1 illustrates the top menu, where the user indicates what VT is to do. The INPUT command allows the user either to enter data on a new job or to modify data from an existing job. The other modes use previously input data. VT displays a default command in brackets at the bottom of the

screen that the user can issue by hitting a carriage return (<cr>). Users can also issue single or multiple commands by typing only a portion of a command word or the number in front of it.

VT's input is menu driven, allowing entire screens of questions to be answered at once by providing defaults wherever possible. The input mode also provides consistency checking of data and a general question-asking mechanism that is used throughout VT. A completed sample input screen is shown in figure 2. Prompts for data appear on the left, defaults and input on the right.

Using a simple command language, the user can confirm some or all values shown, enter or modify values, or register uncertainty about values. Fourteen of these data menus currently exist in the INPUT portion of VT. Once all the data have been entered, the user returns to the top menu, at which point the data can be saved for future use (SAVE) or used immediately in the design task (RUN).

As VT runs, it tentatively constructs an elevator system by proposing component selections and relationships. At the same time, VT specifies constraints with which to test the acceptability of the resulting design and tests each constraint whenever enough is known about the design to evaluate it. Whenever constraints are violated, VT attempts to alter the design (for example, by selecting more expensive equipment) in order to resolve the problem. We refer to these alterations as fixes. VT reports any such constraint violation and the fix that is made, as in figure 3.

There are two types of fix reports. The report shown for MAXIMUM-TRACTION-RATIO is the more common version. It mentions the constraint that was violated, describes the degree of the violation, and lists the corrective action taken. The fix report describing the change to CAR-RUNBY is a special case. This version is used when VT makes an initial estimate for a value in order to calculate a precise value for it. The value of the constraint is the precise value; the estimate is simply changed to this value.

During a noninteractive run, VT uses its own knowledge base to decide how to remedy constraint violations. This knowledge base represents engineering practices that Westinghouse plans to make standard. The RUN can also be done interactively, in which case VT asks for confirmation of each fix before it is actually implemented. If a particular fix is rejected by the user, VT can either find another fix or provide a list of all possible fixes and ask the user to suggest a particular one. Records are kept of user overrides. These overrides are taken into consideration by the system maintainers when modifying the knowledge base. The overriding of a VT-proposed fix by the user might indicate that a standard does not yet exist on a decision VT makes. It might also be the result of outside factors that were too transitory to make it into the VT knowledge or data base, such as a temporary surplus or a shortage of a particular equipment model.

On completion of the run, control returns to the top menu, at which point the user normally goes into

The CAR-RUNBY (estimated to be 6) has been changed to 6.125.

The MACHINE-SHEAVE-HEIGHT (estimated to be 30) has been changed to 26.

The CWT-STACK-WEIGHT (estimated to be 4316.25) has been changed to 4287.36.

The MAXIMUM-TRACTION-RATIO constraint was violated. The TRACTION-RATIO was 1.806591, but had to be <= 1.783873. The gap of 0.2272000E-01 was eliminated by the following action(s):
Decreasing CWT-TO-PLATFORM-FRONT from 4.75 to 2.25
Upgrading COMP-CABLE-UNIT-WEIGHT from 0 to 0.5000000E-01

The MINIMUM-MAX-CAR-RAIL-LOAD constraint was violated. The MAX-CAR-RAIL-LOAD was 6000, but had to be >= 6722.295. The gap of 722.3 was eliminated by the following action(s):
Upgrading CAR-RAIL-UNIT-WEIGHT from 11 to 16

The MINIMUM-PLATFORM-TO-CLEAR-HOISTWAY-RIGHT constraint was violated. The PLATFORM-TO-CLEAR-HOISTWAY-RIGHT was 7.5, but had to be >= 8. The gap of 0.5 was eliminated by the following action(s):
Decreasing CAR-RETURN-RIGHT from 3 to 2.5

The MINIMUM-PLATFORM-TO-CLEAR-HOISTWAY-LEFT constraint was violated. The PLATFORM-TO-CLEAR-HOISTWAY-LEFT was 7.5, but had to be >= 8. The gap of 0.5 was eliminated by the following action(s):
Decreasing CAR-RETURN-LEFT from 25.5 to 25

The MAXIMUM-MACHINE-GROOVE-PRESSURE constraint was violated. The MACHINE-GROOVE-PRESSURE was 149.5444, but had to be <= 119. The gap of 30.544 was eliminated by the following action(s):
Increasing HOIST-CABLE-QUANTITY from 3 to 4

The MINIMUM-HOIST-CABLE-SAFETY-FACTOR constraint was violated. The HOIST-CABLE-SAFETY-FACTOR was 8.395078, but had to be >= 10. The gap of 1.60492 was eliminated by the following action(s):
Upgrading HOIST-CABLE-DIAMETER from 0.5 to 0.625

The MINIMUM-MACHINE-BEAM-SECTION-MODULUS constraint was violated. The MACHINE-BEAM-SECTION-MODULUS was 24.7, but had to be >= 24.87352. The gap of 0.1735 was eliminated by the following action(s):
Upgrading MACHINE-BEAM-MODEL from S10X25.4 to S10X35.0

The CHOICE-SET-HOIST-CABLE-DIAMETER constraint was violated. The HOIST-CABLE-DIAMETER was 0.625, but was constrained to be 0.5. The HOIST-CABLE-DIAMETER became a member of the set by the following action(s):
Upgrading MACHINE-MODEL from 28 to 38

Figure 3. Constraint Violation and Fix Report.

SHOW LAYOUT SPECS GR 24364 **ADMINISTRATION CENTER**

Loading: PASSENGER
Capacity: 3000

Speed: 250

Operation: 1C-2BC-ERL

Travel: 729
Stops: 6 Openings: 6
Machine: 28 Sheave: 30
Deflector Sheave: 20
Groove: K3269 Pressure: 90.03
Angle of Contact: 159.09
Traction Ratio: 1.79
Machine Load: 11691
Motor H.P.: 20
Power Source: —-
Power Supply: 208-3-60
4287 Rails.....Car: 16 Cwt: 11

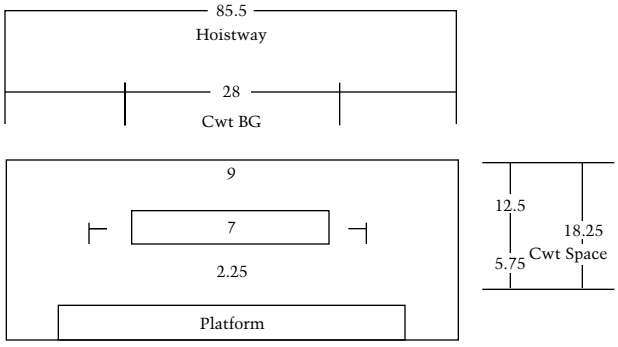
Guide Shoes..Car: 6-R Cwt: 3-R
Buffer.....Car: OH-1 Cwt: OH-1
Stroke.....Car: 8.25 Cwt: 8.25
Safety.....Car: B1 Cwt: —-

Governor: B5B Support: STEEL
Governor Cable: 0.375
Length: 2130
Hoist Cables: (3)-0.5
Length: 1089
Compensation: 3/16-CHAIN
Length: 993
Car sling: 2.5B-18
Crosshead Beam: W8X18
Platform Thickness: 6.625
Sling Weight..... 292
Platform Weight..... 738
Safety Weight..... 465
Cab Weight.....1668
Misc. Weight..... 434
Total Car Weight.....3609
Counterweight Weight: 4824
Subweight Weight:
Buffer Reaction Car: 26437
Cwt: 19296
Machine Weight: 1700
Heat Emission in M.R.: —-
Cable Hanger —-
Safety to Pit: 42

Press RETURN to continue [MENU]: *show layout cwt*

Figure 4. Show Screen for Layout Specs.

SHOW LAYOUT CWT GR 24364 ADMINISTRATION CENTER



Cwt Assembly Weight 537
Cwt Subweight Weight 4287
Total CWT Weight 4824

Overall Cwt Height 138
Maximum Subweight Weight 5273
Cwt Stack Height 87
Maximum Stack Height 107

Maximum Building Tolerance: 1
Stack Percent 81

Press RETURN to continue [MENU]:

Figure 5. Show Screen (Layout CWT).

SHOW mode. SHOW allows users to view data a screenful at a time. Some of the screens are intended for just such a review, and others are intended as input data for other Westinghouse systems (such as manufacturing-oriented programs, cost estimators, and a computer-aided drawing system). Figures 4 and 5 are representative of the sixteen SHOW screens that currently exist; the user accesses these screens by a tree of menus similar to the input menu.

If the user sees something unusual while in SHOW (for example, an unexpected value), the EXPLAIN mode can be used to determine the cause. EXPLAIN can also be used by relative novices to understand how VT performs the design task.

The user interacts with VT's explanation facility by asking questions. The type of information given in the explanation depends on the type of question asked. VT's explanation facility currently provides several types of queries that can be asked about individual system values. These query types are discussed in detail in the next section. The sample interaction in figure 6 demonstrates some of the tools the explanation facility provides, including the use of VT's lexicon of synonyms for system value names.

The only major part of VT that is not visible in figures 1-6 is VT's database. The database is read only and primarily contains data about pieces of equipment and machinery that VT must configure. Each piece of equipment has its own table; the rows of each of these tables represent different models of the equipment from which to choose, and the columns represent attributes relevant to the type of equipment. These attributes can be restrictions on each model's use (for example, maximum elevator speed or maximum load supported by the equipment), values of equipment attributes (for example, height and weight), or lists of model numbers of compatible pieces of equipment.

Calls to the database indicate which table is to be used and what value is to be returned. This value can be either the name of the particular model or the value of one of its attributes. A call might also include an arbitrary number of constraints on the values of each column.

In the event that multiple entries in the database satisfy all the constraints in a call, each table is ordered along an equipment attribute (for example, size) to indicate a preference or priority. The entries in a table are examined

from best to worst, and the first entry satisfying all the constraints is the one from which the return value is obtained.

The VT Architecture

VT solves its problem by constructing an approximate elevator design and successively refining it. The process of constructing an approximate design is forward chaining. Each step in this phase extends the design by procedures that use input data or results of prior decisions to determine a value for a design parameter. Some of these steps embody heuristic knowledge about how to propose an approximate design extension. These steps are needed when the decision is underconstrained or when it must be based on partial information. As VT builds a proposed design, constraints on the elevator system are specified whenever enough information is available to determine their values. The control in this constructive phase is data driven; any step can be taken as soon as the information called for by the procedure associated with the step is available. As it extends the design, VT also builds a dependency network that records for each value which other values were used to obtain it.

The dependency network developed during the forward-chaining constructive phase is enough to identify all contributors to a violated constraint and the value it constrains. These contributors represent potential points to backtrack to in order to revise the proposed design. However, domain expertise is needed to indicate what changes in the proposed design are least costly in real-world terms. Although it is not possible to assign a dollar cost to each revision, domain knowledge determines which of the potential alterations are legal as well as the order of preference among the legal ones.

Demons are used to check for constraint violations; whenever enough is known about the proposed design to supply values for both a constraint and the value it constrains, they are compared. Whenever VT detects a constraint violation, it tests the effectiveness of suggested changes in order of decreasing preference rating. As VT

EXPLAIN GR 24364 ADMINISTRATION CENTER
Explain: *how car runby*
The CAR-RUNBY was determined by a fix.
The CHOICE-SET-CAR-RUNBY constraint was violated.
The CAR-RUNBY was 6, but was constrained to be 6.125.
The CAR-RUNBY was changed from 6 to 6.125.
How[CHOICE-SET-CAR-RUNBY]: <*cr*>
The CHOICE-SET-CAR-RUNBY (6.125) = PIT-DEPTH (72) - [PLATFORM-HEIGHT (6.625) + SAFETY-HEIGHT (9) + CAR-BUFFER-HEIGHT (28.75) + CAR-FOOTING- CHANNEL-HEIGHT (3.5) + CAR-BUFFER-BLOCKING-HEIGHT (18)]
How [PIT-DEPTH]: <*cr*>
The PIT-DEPTH (72) was input by Bob Roche on 25-MAR-1985.
How [PLATFORM-WEIGHT]: *safety height*
The SAFETY-HEIGHT (9) was determined by a database lookup.
It was found in the HEIGHT column of the SAFETY table.
It met the following constraints:
MODEL = SAFETY-MODEL (B1)
How [SAFETY MODEL]: <*cr*>
The SAFETY-MODEL (B1) was determined by a database lookup.
It was found in the MODEL column of the SAFETY table.
It had the SMALLEST HEIGHT that met the following constraints:
MAX-SPEED > SPEED (250)
MAX-PLATFORM-WIDTH >= PLATFORM-WIDTH (70)
MIN-PLATFORM-WIDTH <= PLATFORM-WIDTH (70)
How[SPEED]: *what if safety model B4*
The SAFETY-MODEL is currently B1.
If it were B4, the following major changes would occur:

NAME:	ACTUAL:	PROPOSED:
MACHINE-GROOVE-PRESSURE	114.118	155.563.
TRACTION-RATIO	1.80679	1.76682.
CWT-OVERTRAVEL	49.835	52.835.
CAR-BUFFER-REACTION	26709.4	27652.4.
CWT-STACK-PERCENT	84.1122	88.148.
CWT-BUFFER-REACTION	19684	20627.0.
CWT-PLATE-QUANTITY	90	94.3184.
CWT-WEIGHT	4921.0	5156.76.
CAR-BUFFER-LOAD	6677.35	6913.11.
CAR-WEIGHT	3677.35	3913.11.
DEFLECTOR-SHEAVE-DIAMETER	25	20.
CAR-BUFFER-BLOCKING-HEIGHT	18	17.125.
HOIST-CABLE-MODEL	(4)-0.5	(3)-0.5.
CAR-RUNBY	6.125	6.
SAFETY-MODEL	B1	B4.

Would you like to see ALL values which would change [NO]: <*cr*>
Would you like to implement this [NO]: <*cr*>
How [MACHINE-GROOVE-PRESSURE]: *safety load*
There is more than one SAFETY-LOAD:
1. SAFETY-LOAD-CAR-SIDE-CAR-TOP
2. SAFETY-LOAD-CAR-SIDE-CAR-BOTTOM
3. SAFETY-LOAD-CWT-SIDE-CAR-TOP
4. SAFETY-LOAD-CWT-SIDE-CAR-BOTTOM
Which would you like to know about?
[SAFETY-LOAD-CAR-SIDE-CAR- TOP]: *2*

Figure 6. A Sample Interaction with the Explanation Facility.

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.