Data Modeling and Optimization for Wireless Drive-Through Applications

Zhe Xu, Qiang Zhu, *Senior Member, IEEE*, Yi Guo, T. J. Giuli, K. Venkatesh Prasad, *Senior Member, IEEE*, and Henry Huang

***Abstract*—The increasing capabilities and decreasing cost of information and communication technologies (ICTs) are enabling new categories of vehicular telematics applications, such as wireless drive-through (WDT) services, in which an external service provider provides services to in-vehicle occupants via a wireless connection. Most telematics systems rely on proprietary protocols for data exchange between an in-vehicle client and its external counterpart in individual applications, making it difficult to share data-exchange methods among applications. Sharing data-exchange methods is important because it allows a rapid return on ICT investments. In this paper, we study the data-modeling issues for data exchange between an in-vehicle client and its service provider for WDT services. We present a novel data model that captures the main characteristics of data exchange for the WDT application family, using a formal mathematical representation and an intuitive graphical representation. We show that the proposed data model provides generality for such data exchange. In addition, we propose several optimization techniques based on the data model to minimize user interaction so that a good usability can be achieved. Our simulations and analysis demonstrate that the proposed data model and relevant optimization techniques are promising in supporting efficient WDT applications.**

***Index Terms*—Data modeling, optimization strategies, telematics, vehicular application, wireless drive-through (WDT) service.**

I. Introduction

VEHICLE telematics systems have been increasingly deployed in production. The technology has advanced to a stage that communication through a brought-in phone or built-in wireless transceiver is a common practice in high-volume automobiles. The development of telematics applications provides great convenience to drivers and makes real-time management of transportation systems possible.

For the comfort and convenience of drivers, many passenger vehicles have been equipped with onboard computing modules running a great variety of applications, such as navigation systems, concierge services, and entertainment systems. The onboard computing modules and the applications running on them create an environment similar to that of a personal computer that is connected to the Internet. This, coupled with the growing power of consumer-chosen brought-in devices, enables a new application domain—*wireless drive-through* (WDT) services.

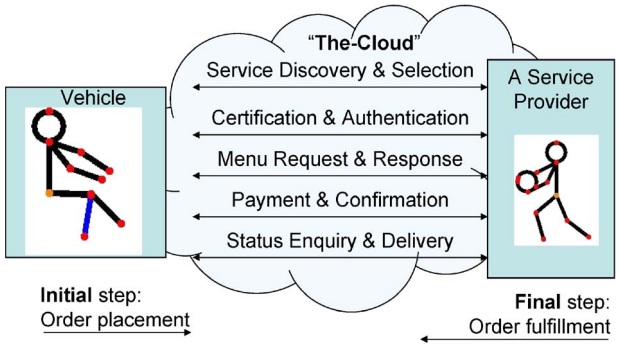

Fig. 1. Example of a WDT application.

When a traditional drive-through service is available at a merchant, a customer (e.g., the driver or a passenger) in a vehicle waits in a queue. The customer typically orders items or services from a menu board. Eventually, the pickup window will deliver items or services ordered by the customer. With a WDT application client running in a vehicle, a customer can place an order at any location that provides wireless network access, which allows the client in the vehicle to connect to a service provider. Similar to an online shopping site, the customer places an order via the in-vehicle client online. After the order has been fulfilled, the relevant items or services will be either delivered or picked up at a preagreed location.

Fig. 1 shows a typical WDT service scenario. When a vehicle arrives at a location of WDT services, the in-vehicle client discovers available services, selects a service of interest (e.g., a fast-food restaurant) from a list, and establishes a wireless connection with the WiFi network of the chosen service provider. Without any user interaction, the in-vehicle client and the service provider mutually authenticate using the driver's credentials stored in the vehicle. Next, the in-vehicle client requests a menu from the service provider. On the receiving of the menu data, the in-vehicle client displays the menu for the customer to place an order. After an order is submitted, the in-vehicle client also handles the payment using stored credit card information or via a third party (e.g., PayPal). Finally, the customer may or may not inquire the status of the order before the order is fulfilled and delivered or picked up. The customer may also cancel or modify an order.

Manuscript received March 12, 2010; revised October 21, 2010 and February 9, 2011; accepted May 5, 2011. Date of publication July 12, 2011; date of current version December 5, 2011. This work was supported by the Ford-University of Michigan Alliance Program. The Associate Editor for this paper was D.-H. Lee.

Z. Xu and Q. Zhu are with the Department of Computer and Information Science, University of Michigan–Dearborn, Dearborn, MI 48128 USA (e-mail: zhe_xu@yahoo.com).

Y. Guo is with the Department of Management Studies, University of Michigan–Dearborn, Dearborn, MI 48126 USA.

T. J. Giuli, K. V. Prasad, and H. Huang are with the Research and Advanced Engineering, Ford Motor Company, Dearborn, MI 48121 USA.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TITS.2011.2159378

Although it seems that the only difference between a traditional drive-through service and a WDT service is where and how an order is placed, this change brings great flexibility and convenience to customers on the road. A customer (driver) may place an order at a fast-food restaurant in the parking lot. Then, free from waiting in a drive-through line, the customer may decide to fill the gastank while the order is being fulfilled. Another example would be even more interesting. Suppose the vehicle has mobile Internet access—an order may be placed on the way and picked up at a service provider's location along the driver's route.

However, the vehicle customer does not have a conventional keyboard or a mouse. Instead, he/she most likely has a touch screen for user interaction. Hence, the implementation of a WDT application should minimize user interaction. In addition, the amount of information exchanged between an in-vehicle client and its service provider should also be minimized. Note that the menu data transferred from the service provider to the corresponding in-vehicle client is the major piece of information for a WDT application. This paper focuses on the data-modeling issues for the menu data and the relevant optimization techniques to minimize user interaction.

The communication method in telematics applications remains in the proprietary domain. Clearly, these proprietary approaches make it difficult to share the data-exchange methods among different applications, resulting in duplicate efforts in implementing similar applications. To our knowledge, we are the first to develop WDT applications, and no similar work has been reported in the literature. Since the WDT application family shares characteristics that allow common data-exchange methods to facilitate the development of new applications, we need to define a data model.

To tackle the aforementioned problem, we propose a novel data model that is applicable to the family of WDT applications. This model is derived by analyzing various requesting relations (e.g., AND, XOR, and IMPLICATION) among the data items/categories in menu data for different WDT applications. The model is presented using a formal mathematical representation, which can be directly implemented in a system, and an intuitive graph representation, which can be easily understood. We also show how to perform optimization to minimize user interaction based on the model. A set of optimization strategies, including a shortcut method and four optimization rules, is suggested. Our simulations demonstrate that the proposed optimization techniques can significantly improve the performance of a WDT service in terms of reduction of user interaction. Our experience with the implementation of two WDT application prototype systems has shown that the proposed data model can significantly reduce the implementation efforts for multiple WDT applications.

Note that wireless networking for a vehicular environment is not the focus of this paper. Much work on vehicular networks has been done previously. For example, a swarming protocol for vehicular ad-hoc networks was proposed to reduce the hot-spot problem in such networks [1]. Assuming an effective wireless network is in place, our work introduces a design that further assists in-vehicle customers in obtaining external services and reduces data exchange between the in-vehicle client and the service provider. The main contributions of this paper are to develop a data model for challenging contexts exemplified by in-vehicle WDT applications and to perform optimization for minimizing user interaction in such an environment.

II. Related Work

Telematics solutions deliver value-added services, including safety, security, information, entertainment, and mobile commerce [2]–[4]. Larsson *et al.* introduced a system [5] that is a typical application of telematics. A cargo-transportation company may install sensors in its vehicles so that relevant environment information, as well as vehicle location information, can be collected and instantly sent to a responsible party. The information enables the real-time monitoring and management of their vehicle fleet. In commercial applications, vehicle location information also helps in overall material planning and logistics by automatically taking into account traffic conditions [6], [7] and recommending optimum routes [8], [9] and dispatching [10]. See [11] for a great survey on in-vehicle positioning and navigation technologies. WDT services are emerging as a new type of telematics applications.

For a WDT service, the communication between the in-vehicle client and the service provider happens in a predefined order. Unlike the traditional online stores, in which the service provider has full control of the ordering process, the WDT service must be implemented in such a way that the vehicle client has a certain level of control over the ordering process so that the user interface is appropriate for driving [12], [13]. When designing interfaces for mobile devices (e.g., telematics devices in a vehicle), context and constraints have to be addressed [14], [15]. Research shows that users weigh usability attributes differently between Web sites and wireless sites cross banking, news, shopping, and tourism [16]. For interface design, Byrne provided a general framework for analyzing interactive tasks and proposed a system to capture menu data of a graphical user interface [17].

Most telematics systems rely on proprietary protocols for data exchange between an in-vehicle component and its external counterpart [18]. For example, Bures discussed possible usage of binary, XML (EXtensible Markup Language), and HTML (HyperText Markup Language) data formats and different coding methods for geographical information in [19]. In [20], the authors investigated communication protocols for vehicular ad hoc networks. Goodman *et al.* presented a system that was designed for information transfer in a cellular network in [21]. Note that the communication protocol should take consideration of the customer's privacy, which is important to vehicular telematics applications [22], [23]. The WDT applications possess some unique characteristics for data exchange and modeling. First, the format of the menu data needs to be well defined so that the service provider and the in-vehicle client understand each other without any ambiguities. Second, the menu data should be modeled in such a way that the implementation of the client-side software promotes the usability of the application, i.e., minimizes user interaction. Moreover, the data model should make optimization possible. In this paper, we study a new data-modeling method that is suitable for WDT

applications and develop optimization techniques that utilize the model to improve the WDT application usability.

Research shows that recommendations support and improve consumer decision making [24] and have impact on consumer choices [25]. Although existing algorithms for recommendation systems such as [26]–[29], to name a few of many examples, serve similar purposes, they are not suitable for the WDT application family. First, they are designed to run on a server. Second, the algorithms work on the history data of behavior of many customers and try to predict the behavior of one particular customer. Moreover, most existing recommendation systems are too complicated to fit in an in-vehicle environment and not necessarily efficient for our applications. The increased complexity makes these techniques infeasible for WDT applications. Hence, we take an adaptive personalized approach based on the individual customer's past behavior [30], [31] to develop our method for generating an item list for the shortcut menu, which is one of our proposed optimization techniques. With regard to our other optimization techniques, no related work has been found in the literature.

III. Modeling of Data

All merchants that provide WDT services should follow the data model for menu information. The model provides both accuracy of data syntax and generality that supports different applications in the family. Each application has a mathematical expression defined in such a way that, each time a customer selects an item following all the rules, the item is added into the shopping cart, or the *to-go bag*.

A. Mathematical Representation

The menu data from a service provider in a given application is a set of *items* organized under a number of *categories*, both of which are related to one another by a set of *rules* that are defined by a set of *operators*. Menu data $\mathbb{M}$ can be modeled as a mathematical expression as follows:

$$\mathbb{M} = \langle \mathbb{E}, \mathbb{N}, \mathbb{P}, \mathbb{R}, r \rangle \tag{1}$$

where $\mathbb{E}$, $\mathbb{N}$, $\mathbb{P}$, $\mathbb{R}$, and r are explained next.

B. Terminal Set $\mathbb{E}$ and Nonterminal Set $\mathbb{N}$

$\mathbb{E} = \{e_1, e_2, \ldots, e_n\}$ is a set of items from a given WDT service provider. For example, an item e_i, which is represented by a leaf node in the tree structure of the graphical representation, might be a "burger" or a "drink" in a fast-food restaurant.

$\mathbb{N} = \{n_1, n_2, \ldots, n_m\}$ represents a set of intermediate entries in the menu, such as submenus, categories, or combos. An element n_i is represented by a nonleaf node in the tree structure. In the case of a fast-food restaurant, an entry n_i could be "drinks" or "kids menu."

A nonleaf node represents a nonterminal symbol, and a leaf node represents a terminal symbol in the mathematical representation. In (1), r is a special nonterminal in $\mathbb{N}$. It is the root node of the tree structure as well as the entry point of ordering an item in the menu.[1] Fig. 2 shows a tree structure example.[2]

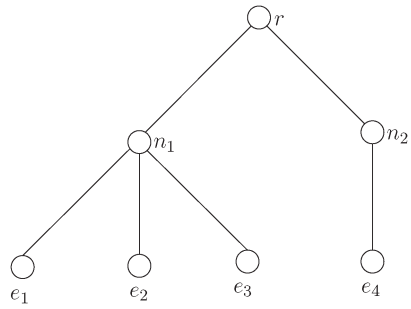

Fig. 2. Tree structure example.

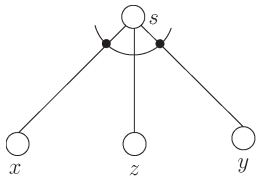

Fig. 3. Graphical representation of $x \wedge y$.

In the rest of this paper, we use the term "symbol" for a terminal or a nonterminal when it is not important to distinguish them.

C. Operator Set $\mathbb{P}$

$\mathbb{P}$ denotes a set of operators that define the relations among nonterminals/terminals. The possible operators are AND ("$\wedge$"), XOR ("$\oplus$"), IMPLICATION ("$\rightarrow$"), and GROUP ("()"). Since $\mathbb{P}$ is for a particular application, in general, $\mathbb{P} \subseteq \{\wedge, \oplus, \rightarrow, ()\}$.

1) AND/XOR*:* The $\wedge$ operator takes two symbols (terminal/nonterminal) as operands. The operator indicates that both symbols have to be selected as a bundle. For example, $x \wedge y$ means "choose both x and y." Although an $\wedge$ operator takes only two operands, more than one $\wedge$ operator can be applied. For example, $x \wedge y \wedge z$ means all three symbols must be selected together. Graphically, we use a solid curve to connect all branches that lead to participating symbols of an AND relation (see Fig. 3 for an example). Note that the solid dot at the cross of the curve and the branch indicates the inclusion of the branch in the relation.

Similarly, the $\oplus$ operator indicates the XOR (exclusive OR) relation of two symbols. For example, $x \oplus y$ means "choose either x or y, but not both." Like the $\wedge$ operator, the $\oplus$ operator can be also applied multiple times. Graphically, we use a dashed curve to connect all the branches that lead to symbols in the relation (see Fig. 4 for an example).

[1]The entry point (node) is where a customer starts in the menu tree structure when selecting an item/combo. A node other than the root could also be assigned as the entry node. For example, the node last visited for the previously selected item/combo could be the entry node for selecting the next item. However, using the last visited node as the entry node may make customers go through more links in some situations where the menu tree is balanced and all leaf nodes have the same probability of getting selected next.

[2]This example only illustrates the leaf nodes (terminals) and nonleaf nodes (nonterminals). Relations among branches introduced later are not shown.

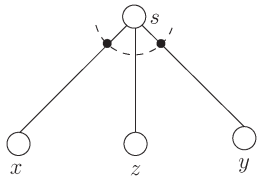

Fig. 4. Graphical representation of $x \oplus y$.

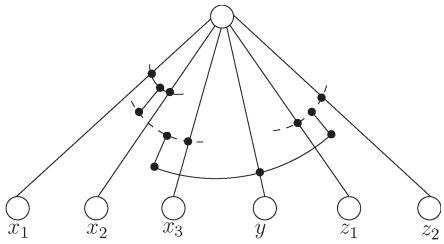

Fig. 5. Graphical representation of GROUP.

Note that the inclusive OR operator can be realized by multiple visits to the menu tree structure, with each visit selecting only one item or combo from those with XOR relations. For simplicity, OR is omitted from our operator set.

2) GROUP*:* In general, the $\wedge$ and $\oplus$ operators may be evaluated in any order (e.g., from left to right) in the mathematical expression. Should a mathematical expression be evaluated in a particular order, the GROUP operator (expressed by a pair of parentheses in the mathematical expression) is applied to indicate a higher priority. A GROUP operator can be embedded in another one recursively, with the most inner one having the highest priority. Graphically, each GROUP operator is represented by a curve that crosses the interested branches. When a group of branches after a GROUP operation is part of an AND/XOR operation, a short branch is created with both ends dotted to indicate "connections." Fig. 5 shows an example of a graphical representation for $((x_1 \wedge x_2) \oplus x_3) \wedge y \wedge (z_1 \oplus z_2)$. Note that the operators at the same level are either all $\wedge$ or all $\oplus$, but not both, after the GROUP operators. In the given example, the top level operator is $\wedge$, i.e., $x \wedge y \wedge z$, where $x = x' \oplus x_3$ and $z = z_1 \oplus z_2$, and in turn, $x' = x_1 \wedge x_2$.

Although the aforementioned graphical representation provides a direct and precise view of a menu tree structure, it may become difficult to be visualized when the tree has nodes with a large number of child nodes and involves multiple levels of $\wedge$ and/or $\oplus$ operators. A "normalization" method is suggested here to resolve the problem. For each GROUP operator in a given mathematical formula, we create a new node in the tree structure, as shown in Fig. 6.

Not surprisingly, the nodes created for the GROUP operators correspond to the intermediate variables created for the mathematical formula, namely, x, z, and x'. If we apply the normalization method throughout the menu tree structure, it is easy to see that all child nodes of a parent node are either all in the AND relation or all in the XOR relation, but not both.

Note that the nonleaf nodes in the tree structure before normalization have semantic meanings. For instance, they can be categories in which the items in a menu are organized such as a breakfast menu. However, the new nodes added during normalization are created only for the representation purpose.

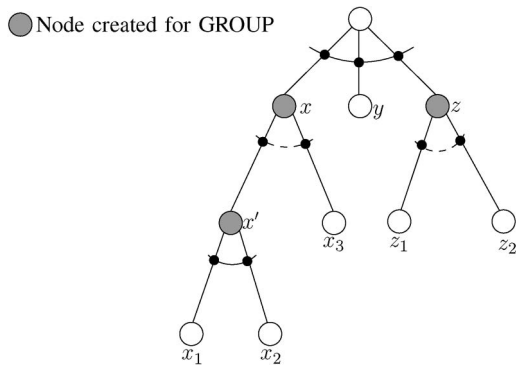

Fig. 6. Creation of new nodes for GROUP.

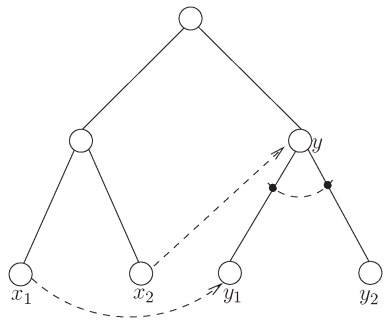

Fig. 7. Example of IMPLICATION.

It is possible that a node x involves in more than one $\wedge$ or $\oplus$ operator. If two nodes are created during the normalization process, both of which are the parent nodes for node x, then the tree structure is violated. In this case, we create a copy (instance) of x for each operator and treat each copy (symbol) as an independent node (symbol).

We have introduced all necessary operators to express the relations among sibling nodes. Nonetheless, the relation between two or more nodes that do not share the same immediate parent node is also useful when the merchant offers several items as a combo to customers. For example, a fast-food restaurant may offer a burger, fries, and a drink as a combo. One way of representing combos in the menu data is to list all valid combinations. However, in most cases, this is not practical due to the possible large number of valid combinations. To overcome the problem, we show the combo to a customer one item at a time and allow the customer to make a decision one item at a time as well. To realize this approach, we need to define another operator, which is discussed as follows.

3) IMPLICATION*:* The IMPLICATION operator, which is denoted by $\rightarrow$, is used in the case when the symbol (terminal or nonterminal) on the left side is selected, the symbol on the right side should be also selected. Fig. 7 shows the example for $x_1 \rightarrow y_1$ and $x_2 \rightarrow y$. Note that some other operators, such as $\wedge$ or $\oplus$, are not shown in this figure. The directed dashed line from the symbol on the left of the operator to the symbol on the right represents the IMPLICATION operator.

There are cases in which selection of an item requires two or more of another item as a combo. We use a number on the dashed line to indicate the number of the other item required. For example, $x \rightarrow y \wedge y$ would need a number 2 on the dashed directed line from node representing x to node representing y.

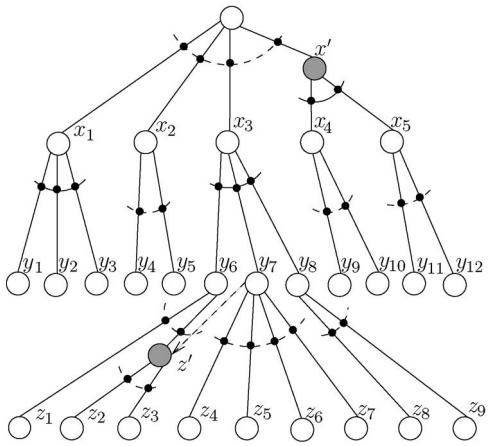

Fig. 8. Example of the rule set.

We can omit the number if the value is 1. It is also noted that each IMPLICATION operation in the menu tree can be applied only once during a cascading invocation. This constraint avoids possible dead loops.

D. Rule Set $\mathbb{R}$

$\mathbb{R}$ is a set of rules that utilize the aforementioned operators to define the root node in the tree structure. The set of rules starts with the definition for the symbol represented by root node r, then the symbols represented by the nodes that are the child nodes of the root node. Eventually, all the nonterminal symbols are defined by the terminal symbols. The rule set of an example mathematical expression is given as follows:

$$\begin{aligned}\mathbb{R} = \{&r := x_1 \oplus x_2 \oplus x_3 \oplus (x_4 \wedge x_5),\\ &x_1 := y_1 \wedge y_2 \wedge y_3,\\ &x_2 := y_4 \oplus y_5,\\ &x_3 := y_6 \wedge y_7 \wedge y_8,\\ &x_4 := y_9 \oplus y_{10},\\ &x_5 := y_{11} \oplus y_{12},\\ &y_6 := z_1 \oplus z_2 \oplus z_3,\\ &y_7 := z_4 \oplus z_5 \oplus z_6 \oplus z_7,\\ &y_8 := z_8 \oplus z_9,\\ &y_7 \rightarrow z_2 \oplus z_3\}\end{aligned}$$

where ":=" denotes "defined as." Fig. 8 shows the tree structure generated from this rule set. As explained in Section III-C2, node x' is added to the menu tree structure during the normalization process to simplify $(x_4 \wedge x_5)$. Node z' is created to simplify $(z_2 \oplus z_3)$ for an IMPLICATION.

E. Customer Order

After a customer has placed all his/her items or combos in the to-go bag, a customer order $\mathbb{O}$ is formed by two sets, namely, $\mathbb{S}$ and $\mathbb{W}$. The former is a set of terminal symbols or expressions of terminal symbols connected by $\wedge$ and/or () operators, and the latter is a set of nonterminal symbols that are not temporary ones created during the normalization process, i.e.,

$$\mathbb{O} = \langle \mathbb{S}, \mathbb{W} \rangle \tag{2}$$

where $\mathbb{S} \subseteq \mathbb{E}$, $\mathbb{W} \subseteq \mathbb{N}$, and elements in $\mathbb{S}$ and $\mathbb{W}$ were chosen following all the rules defined in $\mathbb{R}$.

F. Example

Let us consider an example of such menu data defined by the previously introduced modeling method. This example is a simplified version of menu data from a real fast-food restaurant.

The mathematical expression of the menu data is given as follows:

$$\begin{aligned}&\mathbb{M} = \langle \mathbb{E}, \mathbb{N}, \mathbb{P}, \mathbb{R}, r \rangle\\ &\mathbb{E} : \text{see leaf nodes in Fig. 9}\\ &\mathbb{N} : \text{see nonleaf nodes in Fig. 9}\\ &\quad (\text{excluding those generated by normalization})\\ &\mathbb{P} = \{\wedge, \oplus, (), \rightarrow\}\\ &\mathbb{R} = \{r := Regular \oplus Combos \oplus Kids \oplus Pick2,\\ &\qquad Regular := Burgers \oplus Fries \oplus Beverages,\\ &\qquad Burgers := Ham \oplus Chicken \oplus Fish,\\ &\qquad Fries := French \wedge Sizes,\\ &\qquad Sizes := Small \oplus Medium \oplus Large,\\ &\qquad Beverages := Drinks \oplus Juice,\\ &\qquad Drinks := (Tea \oplus Cola) \wedge Sizes,\\ &\qquad Combos := Burgers' \wedge Fries' \wedge Beverages',\\ &\qquad Fries' := French'' \wedge (Medium' \oplus Large'),\\ &\qquad Beverages' := (Tea'' \oplus Cola'')\\ &\qquad\qquad \wedge (Medium'' \oplus Large''),\\ &\qquad Kids := (French' \oplus Tea' \oplus Cola') \wedge Small',\\ &\qquad Pick2 := Snack \oplus Nuggets,\\ &\qquad Nuggets := Grilled \oplus Crispy,\\ &\qquad Pick2 \rightarrow Pick2\}\,.\end{aligned}$$

Fig. 9 shows the graphical representation of the menu data. The darkly filled circles represent nodes that are copies of other nodes. For example, node $Burgers'$ is a copy of node $Burgers$ under $Regular$, which indicates that node $Burgers'$ under $Combos$ has exactly the same number of child nodes as $Burgers$ under $Regular$ does, and all child nodes of $Burgers'$ are copies of the child nodes of $Burgers$. To save space and make the figure more readable, we do not show child nodes for copies. To avoid a dead loop in copying, the subtree rooted from a node x cannot include any part from the subtree rooted from a node x' if x' is a copy of x.

Note that a copy of node is not the same as the original node due to their different identities. A node is identified by its whole path from the root node. For example, node $Burgers$ has the identity $menu{-}Regular{-}Burgers$, and node $Burgers'$ has the identity $menu{-}Combos{-}Burgers'$.

An interesting IMPLICATION represented by the dashed directed line from $Pick2$, as well as to $Pick2$, is shown in this example. This operation is used to represent "Pick 2 from the following items."

Following the menu data tree structure, a customer could have a to-go bag containing Ham, $French' \wedge Small'$,

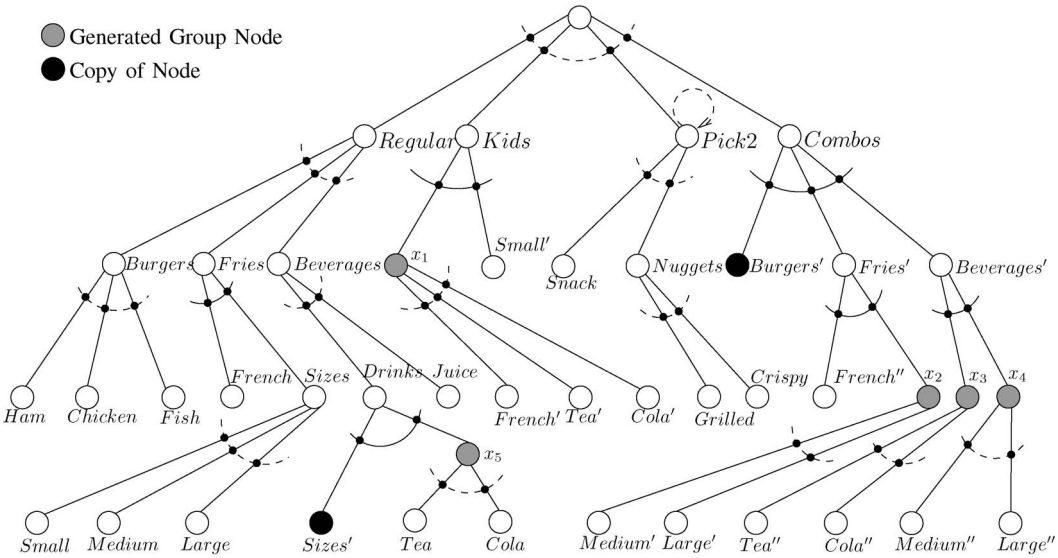

Fig. 9. Example of the menu data.

$Snack \wedge Crispy$, and $Fish' \wedge (Medium' \wedge French'') \wedge (Medium'' \wedge Cola'')$. This customer order is given as follows:

$$\begin{aligned}\mathbb{O} &= \langle \mathbb{S}, \mathbb{W} \rangle \\ \mathbb{S} &= \{Ham, French' \wedge Small', Snack \wedge Crispy, \\ &\quad Fish' \wedge (Medium' \wedge French'') \\ &\quad \wedge (Medium'' \wedge Cola'')\} \\ \mathbb{W} &= \{Regular, Burgers, Pick2, Nuggets, \\ &\quad Combos, Burgers', Fries', Beverages'\}.\end{aligned}$$

Note that $Fish'$ is a copy of node that is not shown in the figure. The parent node of $Fish'$ is $Burgers'$ under $Combos$.

IV. Optimization

Unlike applications running on desktop or laptop computers, which enjoy the convenience provided by a keyboard and a mouse, WDT applications have to use a relatively limited touch screen in a vehicle for user interaction. As a consequence, it is crucial to minimize user interaction throughout the whole ordering procedure. Since the menu data structure is highly related to the development of the user interface, which, in turn, affects the user interaction required, the menu data needs to be optimized as well.

A. Optimization Measures

To place an item into the to-go bag, a customer needs to start at the root node of the menu tree structure, then go through the nonleaf nodes, and eventually reach a leaf node for a terminal symbol. Repeating the said procedure completes a combo in the to-go bag when necessary. Hence, in the menu tree structure, each link on the path from the root node to a leaf node represents a click on the touch screen in the implementation of the user interface. We want to keep each leaf node as close to the root as possible, without violating other restrictions of the menu data so that a customer would go through fewer steps to select an item. Hence, the optimization objective becomes to optimize the menu tree structure.

There are several possible ways to evaluate optimization techniques. The total links in the tree structure is one of the possible measures to evaluate the quality of a menu tree. However, the total number of links is highly related to the total number of categories and items in the menu data. The total number of nodes in a menu tree structure exhibits a similar characteristic. Another possible measure would be the maximum depth of the tree structure. This represents the maximum number of clicks needed to place a single item into the to-go bag. Being pessimistic, this measure only states the worst-case scenario. On the other hand, the minimum depth is too optimistic. Obviously, the average number of links between leaf nodes and the root node is more appropriate for evaluation of menu tree quality. The worst case complexity of calculating each of aforementioned measures is $O(n)$, where n is the number of nodes in the entire menu data tree structure.

In this paper, we adopt the average and maximum depths as our optimization measures. In the rest of this section, we discuss several data optimization techniques for the WDT application family.

B. Shortcut Strategy

The shortcut optimization method tries to connect the desired items directly to the root node. The idea is to create a list of most likely being ordered items/combos from a customer's previous order history. These items/combos are displayed on the first screen of the user interface in an attempt to minimize user interaction.

Each item i in the order history is associated with a weight value w_i that is used for the sorting purpose. The most significant factor affecting the value of w_i would be the number of times the item has been ordered. However, some other factors can be also significant, which include how recent the order was placed, what time of the day the order was placed, what season of the year the order was placed, where the order was placed,

and who placed the order. Before discussing the general case, let us first use a fast-food restaurant as an example to illustrate the idea of our shortcut strategy with two variables defined as follows:

$$h = \begin{cases} 0, & \text{morning} \\ 1, & \text{day} \\ 2, & \text{evening} \end{cases} \quad \text{and} \quad s = \begin{cases} 0, & \text{summer} \\ 1, & \text{spring/fall} \\ 2, & \text{winter.} \end{cases}$$

Combining different values of h and s, we have in total nine classes for the order history data according to the time when an order was placed. These classes are denoted by $C^{h,s}$ $(0 \leqslant h, s \leqslant 2)$. In each class, an item i keeps a variable $p_i^{h,s}$ that is initialized to 0 and updated each time an order is placed during time h and season s as follows:

$$p_i^{h,s} = \begin{cases} 1 + \alpha \cdot p_i^{h,s}, & i \text{ is included in the order} \\ \alpha \cdot p_i^{h,s}, & i \text{ is not included in the order} \end{cases} \tag{3}$$

where α is a real number satisfying $0 \leqslant \alpha \leqslant 1$. Depending on the value of α, the more recent orders might be considered more important than earlier ones.

The general equation for w_i is defined as follows:

$$w_i = l_i \cdot \sum_{h=0}^{m-1} \sum_{s=0}^{n-1} \left(p_i^{h,s} \cdot e^{-(\beta \cdot |h-h_0| + \gamma \cdot |s-s_0|)} \right) \tag{4}$$

where e stands for the base of the natural logarithm; l_i is the number of links that would be saved if item i appears in the shortcut menu; β and γ are positive real numbers; m and n are the numbers of possible values that h and s have, respectively; and h_0 and s_0 are the current values.

In (4), the previous orders are classified by two dimensions, namely, h and s, into $m \times n$ classes. In our restaurant example, the two dimensions could be a small-scale time slot and a large-scale time zone. For other service providers, the two dimensions could be a time zone and a location, or other criteria to classify the previous orders. If a previous order with an h value has little effect on the current order with value h_0, a large β value can be chosen so that $e^{-(\beta \cdot |h-h_0|)}$ approaches 0 and the corresponding $p_i^{h,s}$ value contributes little to the overall weight. On the other hand, if the h value should affect the overall weight heavily, then a smaller value of β can be chosen. Similarly, selecting an appropriate value for γ can control the portion of contribution to the overall weight w_i by the offset of the s dimension.

The $p_i^{h,s}$ values in one of the $m \times n$ classes are updated at the time of placing an order according to (3). The time complexity of updating is $O(j)$, where j is the number of items in the history data. The cost of sorting occurs at the time when the application starts. Calculation of w_i for all items has time complexity of $O(k)$, where k is the number of valid items in the whole menu data. The cost of sorting is $O(k \log k)$ using merge-sort. Nonetheless, because of the relatively small k value of menu data and relatively advanced computer technologies, the cost for sorting is tolerable. When the k value for a certain application is considerably large, we could calculate all values at application idle time, e.g., at the time while the customer is waiting for the order to be delivered after placing an order, and use the appropriate one at the next start time.

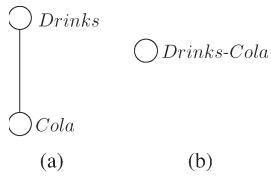

Fig. 10. Example: Optimization Rule 1—Single leaf. (a) Before. (b) After.

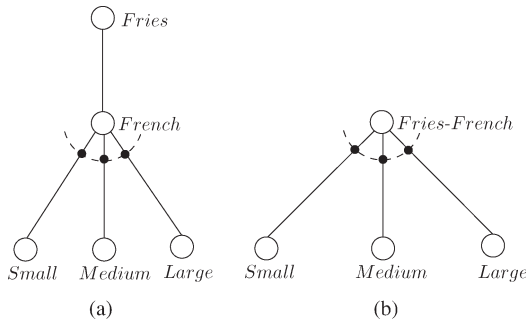

Fig. 11. Example: Optimization Rule 1—Single nonleaf. (a) Before. (b) After.

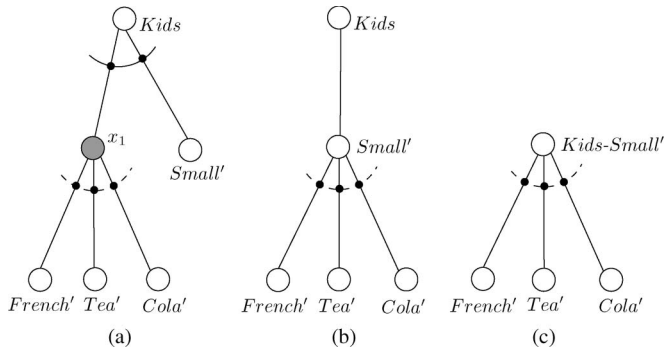

Fig. 12. Example: Optimization Rule 2.

C. Optimization Rules

We now discuss the optimization strategies on the menu data tree structure. Note that all the optimization rules to be discussed are not applied to root node r.

Optimization Rule 1: If a node x has no siblings, then merge x into its parent node, e.g., y. If x has k child nodes $z_1, z_2, \ldots, z_k$, then move $z_1, z_2, \ldots, z_k$ up as the direct child nodes of y. ■

Figs. 10 and 11 show examples of applying Optimization Rule 1 on a single leaf node and a single nonleaf node, respectively.

Optimization Rule 2: If a symbol represented by a leaf node x is in the AND relation with other symbols represented by the siblings of x, merge x into one of its siblings. ■

When one of the items in a combo, e.g., x, is directly in the AND relation with other groups of items, it has to be selected to place the combo into the to-go bag. Hence, item x can be in the parent node of a group of items that are to be selected by the customer as part of the combo. Fig. 12 shows such an example.

Interestingly, the new subtree in Fig. 12(b) then falls into the condition for applying Optimization Rule 1. Fig. 12(c) shows the subtree structure after applying two rules.

On each screen of the user interface, a number of icons show all the choices a customer currently has. These icons represent

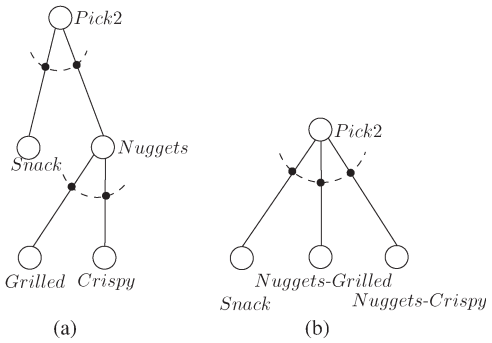

Fig. 13. Example: Optimization Rule 3. (a) Before. (b) After.

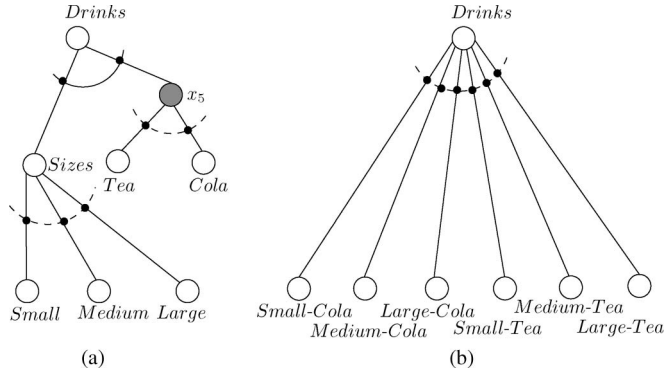

Fig. 14. Example: Optimization Rule 4. (a) Before. (b) After.

the nodes in the menu tree structure, and all the nodes are in the XOR relation. If the clicked icon represents a leaf node, then the corresponding item is added into the to-go bag. Otherwise, the child nodes, which are also in the XOR relation, are shown as icons on the screen. Limited by the size of the screen, the number of icons that can be shown on each screen is bounded from the top,[3] e.g., N. Since we can always split a node if it has more than N child nodes, without loss of generality, we can assume that each node in a given menu tree structure has less than or equal to N child nodes. To minimize the number of screens shown to a customer, we want to show the maximal possible number of icons on each screen. For this purpose, we introduce the following two rules to optimize the menu tree structure.

Optimization Rule 3: If a node x is in the XOR relation with its siblings and all of its child nodes are in the XOR relation as well, then distribute x into every child node of x, and move all the child nodes of x up to become direct child nodes of the parent node of x, provided that the total number of resulting siblings is less than or equal to N. ■

An example of applying this rule is shown in Fig. 13.

Optimization Rule 4: Let p and q be two sibling nodes that are in the AND relation. If the child nodes of p and q are in their respective XOR relations, then convert $p \wedge q$ into a disjunctive form with each clause represented by a single node, and the resulting nodes become the child nodes of a new node that is a combination of p and q, provided the resulting number of sibling nodes is less than or equal to N. ■

Fig. 14 shows an example of applying this rule.

D. Optimization Algorithm

When applying the optimization rules, we need to follow a sequence such that applying one of the rules shall not result in a (sub)tree structure that another rule that has been applied before can be applied again, in which case, the complexity of applying the rules may not be polynomial to the number of nodes in the tree. See Fig. 12 for an example.

Algorithm 1 introduced can avoid such repeated applications of the same rule. N in the algorithm is the maximum number of icons one screen can show.

[3]Note that the choice of N is based on the display size, the icon size, and the number of icons with which a user is comfortable dealing. We assume that, for a given application, N is chosen in such a way that these factors are properly considered.

Algorithm 1 Menu Tree Optimization

procedure $Optimize(node\ p, int\ N)$
1: **for** each unprocessed child node c of p **do**
2: $Optimize(c, N)$
3: **if** c = leaf & c has siblings & in $\wedge$ **then**
4: merge c into a sibling node
5: **else**
6: **for** each of c's processed siblings b that is in $\wedge$ with c **do**
7: $k \leftarrow$ number of nodes generated for disjunction form of $c \wedge b$
8: **if** $k \leqslant N$ **then**
9: merge b into c, having disjunction clause nodes as processed children
10: **end if**
11: **end for**
12: $m \leftarrow$ number of siblings of c
13: $n \leftarrow$ number of children of c
14: **if** c in $\oplus$ & children of c in $\oplus$ & $m + n \leqslant N$ **then**
15: incorporate c into each child, then replace c by children
16: **else if** c has no sibling **then**
17: **if** c has children **then**
18: move children up as children of p
19: **end if**
20: merge c into its parent p
21: **else**
22: mark c as processed
23: **end if**
24: **end if**
25: **end for**

The algorithm consists of a procedure that makes recursive calls on all child nodes of an input node. Initially, all nodes in the menu tree structure are marked as unprocessed. We take root node r as the initial input, and then, the procedure works through the whole tree structure, from leaf nodes up to the root (note that the root node r is not processed for optimization). First, lines 3–4 implement Optimization Rule 2. Then, lines 6–11 implement Optimization Rule 4. Next, Optimization Rule 3 is implemented by lines 12–15. Finally, lines 16–20

implement Optimization Rule 1. Line 22 marks either node c or its derived node (i.e., for the case in which other node(s) has merged into c) if c did not merge into other node(s).

The complexity of the algorithm is $O(n)$, where n is the total number of nodes. Alerted readers may have noticed that n may change during the optimization process. Nonetheless, it decreases during the process except for the application of Optimization Rule 4, in which case the total number of siblings of each involved node is bounded from the top by N. Line 6 involves pairing two nodes out of s sibling nodes; thus, the time complexity of this procedure is $O(s^2)$, where s is the largest number of sibling nodes throughout the entire menu tree. In addition, note that s may change during the optimization process. Fortunately, $s \leqslant N$ as well. Therefore, the overall complexity is $O(n \cdot N^2)$.

The following theorem ensures that the sequence of the optimization rules applied in the algorithm avoids the issue of the repeated applications of the rules on the same node.

Theorem 1: Algorithm 1 implements the four optimization rules in such a sequence that the rules that run later do not create a (sub)tree structure suitable to apply the rules that have run earlier in the algorithm.

Proof: The algorithm runs from leaf nodes to the root node, i.e., a node would not be processed by the algorithm until all of its child nodes have been processed. This fact allows us to prove the theorem by induction as follows.

For the initial step, we show the theorem stands for leaves.

For each node, the algorithm implements the four rules in the following sequence: Rule 2 $\rightarrow$ Rule 4 $\rightarrow$ Rule 3 $\rightarrow$ Rule 1.

Suppose a leaf node y is a child node of node x and the algorithm is executing $Optimize(x, N)$.

After application of Optimization Rule 1, y either remains unchanged or merges into x to form a new node $x - y$, which will run the algorithm for all four rules later. In either case, Optimization Rule 1 does not change any other existing node that has been applied the other three rules.

Next, let us consider Optimization Rule 3. Application of this rule does not create any new node that is in the AND relation with its siblings and, hence, does not create any structure to which Optimization Rule 4 or 2 is applicable.

Finally, Optimization Rule 4 does not create any leaf node that is in the AND relation with its siblings, which is the only situation in which Optimization Rule 2 applies.

For the induction step, suppose again x has a child node y, which has child nodes $z_1, \ldots, z_k$, and assume that the theorem stands for all nodes that are descendants of y. The arguments are very similar to those for the initial step except for Optimization Rule 1.

If y has no other siblings, after application of Optimization Rule 1, y merges into x, and $z_1, \ldots, z_k$ become child nodes of x. Nonetheless, the only nodes that have been changed and applied all the rules are $z_1, \ldots, z_k$, and neither their number of siblings nor their relation changes. Hence, no condition is created for applying the other three rules. ■

After applying the optimization algorithm on a specific menu data tree structure, some nodes that have been involved in an IMPLICATION operation might have changed or even merged into other nodes. Nonetheless, the application of the four rules does not affect the IMPLICATION operations when the following changes are made.

1) If a node y has merged into x to form a new node $x - y$ after applying Optimization Rule 1, 2, or 4, replace y and x by $x - y$ in all the operations.
2) If a node y is distributed into its child nodes $x_1, \ldots, x_k$ and replaced by newly generated nodes $y - x_1, \ldots, y - x_k$ after applying Optimization Rule 3, replace y by $y - x_1 \oplus \cdots \oplus y - x_k$ and replace x_i by $y - x_i$ $(1 \leqslant i \leqslant k)$ in all the operations.

V. Simulation Study

We conducted extensive simulation experiments to evaluate the performance of the optimization strategies, and the results are presented here.

A. Performance of the Shortcut Menu

The first set of simulation experiments was to evaluate the shortcut menu strategy. The simulation settings are described as follows.

For (4), we set $\alpha = 0.5$ as the fading speed of the effect of previous orders. To adequately emphasize the difference between classes, we set $\beta = \gamma = 1$. We chose the total number of classes to be 6, three values for variable h and two values for variable s. In addition, we chose $N = 8$.

For the order history, each class kept previously ordered items in a queue, whose size was limited by the window size. We assumed that each order only included one item. When an item was ordered in a certain class, it was added to the corresponding queue, and the oldest item in the queue was deleted when an overflow occurred.

We used a simplified menu from a typical fast-food restaurant to construct a menu tree structure in the simulation. The menu included 44 items in total. Among them, 10 items had a depth of 3, five items had a depth of 4, five items had a depth of 5, and 24 items had a depth of 6. Note that some items were combo items, and their depth reflected the number of links a customer needs to go through to reach all items of a combo.

Reordering happens when an item is selected from the history queue. An item is selected from the input menu when reordering does not happen. Human behavior dramatically affects the performance of a shortcut-menu-based system, but the most affecting factor is the reordering probability. A higher probability would result in a better performance. We repeated the simulation under different probabilities of reordering. For each fixed reordering probability, 600 orders were executed, averaging 100 for each class.

Fig. 15 shows a typical result for the aforementioned settings. The reordering probability ranged from 0.05 to 0.95. Before an order was placed, a shortcut menu was constructed with eight items with larger w_i values. The probability of finding an item in the shortcut menu is very close to the reordering probability, which indicates that the w_i value of an item provides an accurate means for sorting items and constructing a shortcut menu accordingly.

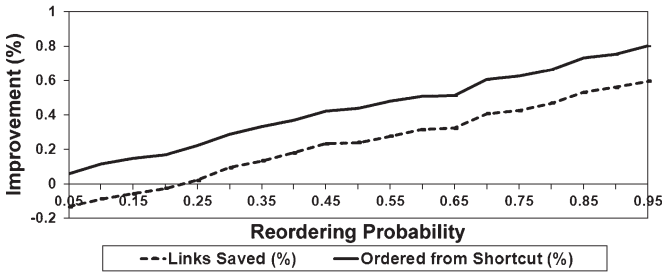

Fig. 15. Improvement (%) by the shortcut strategy.

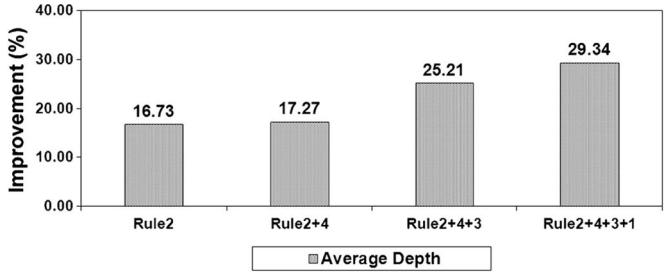

Fig. 16. Improvement (%) by four rules ($N = 8,\ D = 8,\ P = 0.2$).

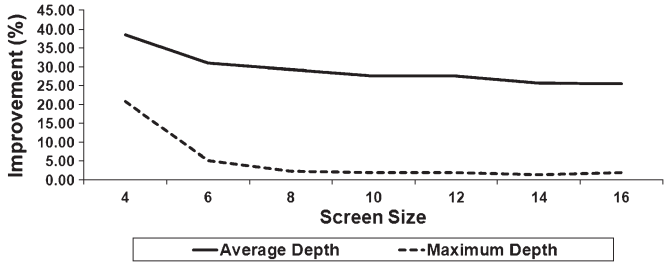

Fig. 17. Improvement (%) versus screen size N ($D = 8,\ P = 0.2$).

If the ordered item was included in the shortcut menu, the link count was 1. Otherwise, the link count was the depth of the item in the menu tree plus 1, since a customer would need to take one extra step to check the shortcut menu first. The percentage of the number of links saved was calculated by comparing the system with a shortcut menu to the system without a shortcut menu, in which case, the link count for an order was exactly the depth of the item in the order. The percentage of the number of links saved also increases proportionally to the reordering probability at a slope almost identical to that of the percentage of ordered from the shortcut menu.

When the reordering probability was less than 20%, the system worked better without a shortcut menu. This is because the customer takes an extra step of checking the shortcut menu without finding the desired item. When a customer had a more than 20% reordering probability, the system with a shortcut menu outperformed the original system significantly, particularly when the reordering probability was higher than 40%. For example, if the reordering probability was 50%, a customer had a 44% chance of finding a desired item in the shortcut menu, and on average, the system had a 24% performance improvement.

B. Performance of Optimization Rules

To evaluate the performance of the optimization rules, we adopted randomly constructed test menu trees. Starting from the root node, a tree structure was created one node at a time. The root node had child nodes that were in the XOR relation. For each child node, we generated child nodes randomly constrained by the following parameters.

The first parameter was screen size N, which was a limit for the number of child nodes that a parent node can have, and the number of child nodes was randomly chosen from 1 to N. The second parameter was the maximum depth D of the tree. We set the probability for a node to have child nodes decrease by the depth of the node. To be specific, when a random real number $rand$ between 0 and 1 was generated, the probability for a node to have child nodes was set to $rand/depth$, where $depth$ was the depth of the node. We set the cutoff probability to be 0.1, i.e., if $rand/depth < 0.1$, the node had no child nodes. Otherwise, a random number between 1 and N was chosen as the number of child nodes. From the aforementioned settings, the maximum depth was set to be bounded from the top by 10. The third parameter in the simulation was the probability P for sibling nodes to be in an AND relation.

We empirically applied the four optimization rules to each of the randomly generated test tree structures. For each set of parameters, 100 samples had been taken for the average depth of all leaf nodes in a particular tree structure. The results showed that each optimization rule contributes to the overall improvement (%) (i.e., decrease (%) for the average depth of all leaf nodes). Fig. 16 shows one of the results with a typical set of parameter values. Although the average depth does not fully reflect the contributions of Optimization Rule 4 since it does not consider the fact that a leaf node created by this rule represents a combo (multiple items), this rule still provided an overall saving of about 30% on average.

Next, we show the effect from each of the three parameters for the test tree structure construction on the improvement provided by the optimization rules. Each time, we varied one of the three parameters and fixed the other two. We again took the average of 100 samples.

First, we changed screen size N from 2 to 16 and fixed the other two parameters. Fig. 17 shows that the optimization rules provided a larger improvement when $N < 6$ since a smaller value of N leads to a larger probability of single child node cases. However, when $N \geqslant 6$, the screen size had a limited effect on the two performance measures. In general, the menu tree had a 25%–40% improvement on the average depth of all leaf nodes and a 2%–22% improvement on the maximum depth of all the leaf nodes.

Fig. 18 shows a typical result when we increased the value of D from 4 to 16 and fixed the other two parameters. The result indicated our optimization rules performed consistently across different values of the maximum depth. The optimization rules provided about 30% savings on the average depth of all leaf nodes of a menu tree structure and about 5% savings on the maximum depth of a menu tree structure.

Since our optimization rules work toward nodes that are in an AND relation more aggressively, the possibility of nodes in the AND relation when we generate a menu tree plays a

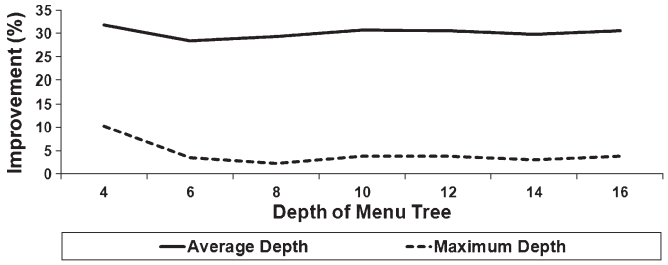

Fig. 18. Improvement (%) versus maximum depth D ($N = 8$, $P = 0.2$).

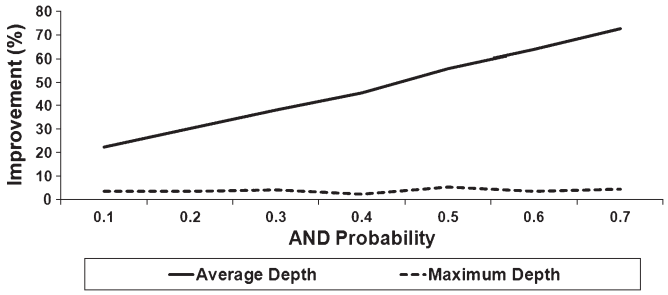

Fig. 19. Improvement (%) versus AND probability P ($N = 8$, $D = 8$).

very important role in the improvement (%) of the performance measures. Fig. 19 shows a typical result when we increased the value of P from 0.2 to 0.7 and kept the other two values constants. The result indicated that the improvement (%) on the average depth of all leaf nodes of a menu tree structure dramatically increased (40%–50% improvement on average) when the value of P was increased. The improvement (%) on the maximum depth of a menu tree structure stayed unchanged at about 4% across different P values.

The simulation results here show that our optimization strategies provide a significant improvement on the performance of WDT applications.

VI. Conclusion

We have introduced the notion of WDT services, proposed a formal mathematical framework for the applications, and built a core simulation platform with two prototypes. Both mathematical and graphical representations of the data model have been developed. The former is useful for both the efficient implementation and the repurposing of the model, whereas the latter makes the model easy to comprehend. Since the data model provides generality of data exchange for the WDT application family, its utilization can significantly reduce the implementation efforts for such applications.

The proposed data model also provides a base for performing optimization to minimize user interaction, which is crucial to WDT applications due to the attention-demanding vehicle environment. We have presented several optimization techniques. Our optimization algorithm has been made efficient by following a particular sequence of invoking four optimization rules to avoid the reconsiderations of invocations. Our simulation results demonstrate that the proposed optimization techniques can significantly improve the performance of a WDT application in terms of reduction of user interaction.

Our work is just the beginning of a rich body of research that needs to be carried out for the realization of efficient WDT applications. Our research plans include extending the data model to incorporate more constraints (e.g., membership privileges and general conditional policies), studying the shortcut algorithms, fine-tuning our models to best match the real-world expectations of the WDT service to maximize the customer experience, and conducting human behavior study and comprehensive statistical analysis to evaluate our techniques in real-world and hypothetical settings.

References

[1] A. Nandan, S. Das, G. Pau, M. Gerla, and M. Y. Sanadidi, "Co-operative downloading in vehicular ad-hoc wireless networks," in *Proc. 2nd Conf. Wireless On-demand Netw. Syst. Services*, 2005, pp. 32–41.

[2] M. Kim, Y. Choi, Y. Moon, S. Kim, and O.-C. Kwon, "Design and implementation of status based application manager for telematics," in *Proc. 8th IEEE Int. Conf. Adv. Commun. Technol.*, 2006, pp. 1364–1366.

[3] E. Scomavacca, S. J. Barnes, and S. L. Huff, "Mobile business research published in 2000–2004: Emergence, current status, and future opportunities," *Commun. Assoc. Inf. Syst.*, vol. 17, no. 28, pp. 635–646, 2006.

[4] T. Bauer, J. Herrmann, P. Liggesmeyer, and C. Robinson-Mallett, "A flexible integration strategy for in-car telematics systems," *ACM SIGSOFT Soft. Eng. Notes*, vol. 30, no. 4, pp. 1–7, Jul. 2005.

[5] T. Larsson, M. Taveniku, C. Wigren, P.-A. Wiberg, and B. Svensson, "T4-telematics for totally transparent transports," in *Proc. 8th IEEE Int. Conf. Intell. Transp. Syst.*, 2005, pp. 467–472.

[6] J. Munson, S. Lee, D. Lee, D. Wood, G. Thompson, and A. Cole, "A rule-based system for sense-and-respond telematics services," in *Proc. Workshop E2E, Sense-Respond Syst., Appl. Service, Int. Conf. Mobile Syst.*, 2005, pp. 31–36.

[7] J. Lee, G.-L. Park, C. O. Sung, and S.-W. Kim, "A group management scheme for an efficient location-based service," in *Proc. ACM Symp. Appl. Comput.*, 2008, pp. 1705–1709.

[8] A. Pathak, A. Tripathy, and P. K. Patra, "Algorithmic design and query processing of spatial data management in vehicular telematics system," in *Proc. Int. Conf. Adv. Comput., Commun. Control*, 2009, pp. 187–191.

[9] B. Bartin and K. Ozbay, "Determining the optimal configuration of highway routes for real-time traffic information: A case study," *IEEE Trans. Intell. Transp. Syst.*, vol. 11, no. 1, pp. 225–231, Mar. 2010.

[10] K. T. Seow and D.-H. Lee, "Performance of multiagent taxi dispatch on extended-runtime taxi availability: A simulation study," *IEEE Trans. Intell. Transp. Syst.*, vol. 11, no. 1, pp. 231–236, Mar. 2010.

[11] I. Skog and P. Händel, "In-car positioning and navigation technologies–A survey," *IEEE Trans. Intell. Transp. Syst.*, vol. 10, no. 1, pp. 4–21, Mar. 2009.

[12] M. Alpern and K. Mindardo, "Developing a car gesture interface for use as a secondary task," in *Proc. Conf. Human Factors Comput. Syst.*, 2003, pp. 932–933.

[13] T. Ersal, H. J. A. Fuller, O. Tsimhoni, J. L. Stein, and H. K. Fathy, "Model-based analysis and classification of driver distraction under secondary tasks," *IEEE Trans. Intell. Transp. Syst.*, vol. 11, no. 3, pp. 692–701, Sep. 2010.

[14] Y. E. Lee and I. Benbasat, "A framework for the study of customer interface design for mobile commerce," *Int. J. Electron. Commerce*, vol. 8, no. 3, pp. 79–102, Spring 2004.

[15] D. J. Wheatley, "Beyond the desktop: And into your vehicle," in *Proc. Conf. Human Factors Comput. Syst.*, 2000, pp. 43–44.

[16] V. Venkatesh and V. Ramesh, "Web and wireless site usability: Understanding differences and modeling use," *MIS Quart.*, vol. 30, no. 1, pp. 181–206, 2006.

[17] M. Byrne, "ACT-R/PM and menu selection: Applying a cognitive architecture to HCI," *Int. J. Human-Comput. Stud.*, vol. 55, no. 1, pp. 41–84, Jul. 2001.

[18] C. Bisdikian, I. Boamah, P. Castro, P. Castro, A. Misra, J. Rubas, N. Villoutreix, D. Yeh, V. Rasin, H. Huang, and C. Simonds, "Intelligent pervasive middleware for context-based and localized telematics services," in *Proc. 2nd Int. Workshop Mobile Commerce*, 2002, pp. 15–24.

[19] P. Bures, "The architecture of traffic and travel information system based on protocol TPEG," in *Proc. Euro Amer. Conf. Telematics Info. Syst.*, 2009, pp. 29–31.

[20] M. Dikaiakos, A. Florides, T. Nadeem, and L. Iftode, "Location-aware services over vehicular ad-hoc networks using car-to-car communication," *IEEE J. Sel. Areas Comm.*, vol. 25, no. 8, pp. 1590–1602, Oct. 2007.
[21] D. J. Goodman, J. Borrás, N. B. Mandayam, and R. D. Yates, "Infostations: A new system model for data and messaging services," in *Proc. IEEE Veh. Technol. Conf.*, 1997, vol. 2, pp. 969–973.
[22] S. Duri, M. Gruteser, X. Liu, P. Moskowitz, R. Perez, M. Singh, and J.-M. Tang, "Framework for security and privacy in automotive telematics," in *Proc. 2nd Int. Workshop Mobile Commerce*, 2002, pp. 25–32.
[23] S. Duri, J. Elliott, M. Gruteser, X. Liu, P. Moskowitz, R. Perez, M. Singh, and J.-M. Tang, "Data protection and data sharing in telematics," *Mobile Netw. Appl.*, vol. 9, no. 6, pp. 693–701, Dec. 2004.
[24] B. Xiao and I. Benbasat, "E-commerce product recommendation agents: Use, characteristics, and impact," *MIS Quart.*, vol. 31, no. 1, pp. 137–209, 2007.
[25] S. Senecal and J. Nantel, "The influence of online product recommendations on consumers' online choices," *J. Retailing*, vol. 80, no. 2, pp. 159–169, 2004.
[26] N. Golvin and E. Rahm, "Reinforcement learning architecture for Web recommendations," in *Proc. Int. Conf. Inf. Tech.—Coding Comput.*, 2004, vol. 1, pp. 398–402.
[27] Li-Tung, Y. Xu, and Y. Li, "A framework for e-commerce oriented recommendation systems," in *Proc. Int. Conf. Active Media Technol.*, 2005, pp. 309–314.
[28] Q. Yang, S. Zhang, and B. Feng, "Research on personalized recommendation system of scientific and technological periodical based on automatic summarization," in *Proc. 1st IEEE Int. Symp. Inf. Tech. Appl. Educ.*, 2007, pp. 34–39.
[29] Y. Kawai, J. Zhang, and H. Kawasaki, "Tour recommendation system based on Web information and GIS," in *Proc. IEEE Int. Conf. Multimedia Expo*, 2009, pp. 990–993.
[30] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions," *IEEE Trans. Knowl. Data Eng.*, vol. 17, no. 6, pp. 734–749, Jun. 2005.
[31] T. S. Chung, R. T. Rust, and M. Wedel, "My mobile music: An adaptive personalization system for digital audio players," *Marketing Sci.*, vol. 28, no. 1, pp. 52–68, 2009.

Zhe Xu received the M.S. degree in telecommunications engineering, the M.S. degree in computer science, and the Ph.D. degree in telecommunications engineering from the University of Texas at Dallas in 2001, 2005, and 2007, respectively.

He is currently a Research Fellow in computer and information science with the University of Michigan–Dearborn. His research interests include scalable quality-of-service scheduling in high-speed networks, service-oriented computing, and telematics.

Qiang Zhu (SM'05) received the Ph.D. degree in computer science from the University of Waterloo, Waterloo, ON, Canada, in 1995.

He is currently a Professor of computer and information science with the University of Michigan–Dearborn. He is also an IBM Center for Advanced Studies Faculty Fellow with the IBM Toronto Laboratory. He has published numerous papers in various top journals and conference proceedings in his field. Some of his research results have been included in several well-known database books. His research has been funded by highly competitive sources, including the U.S. National Science Foundation, IBM, and Ford. His current research interests include query processing and optimization, multidimensional indexing, streaming data processing, Web information systems, service-oriented computing, and autonomic computing.

Dr. Zhu is the Editor-in-Chief of the *International Journal of Computers and Their Applications* and has served as a Program/Organizing Committee Member for many international conferences.

Yi Guo received the M.S. degree from the University of Nebraska at Omaha and the Ph.D. degree from Texas A&M University, College Station.

She is currently an Associate Professor of management information systems with the University of Michigan–Dearborn. Her work has appeared in *Decision Support Systems*, the *Information Systems Journal*, *Communications of AIS*, *Decision Sciences Journal of Innovative Education*, the *Information Resources Management Journal*, the *Journal of Organizational and End User Computing*, the *International Journal of Information Quality*, the *Service Industries Journal*, and other journals. Her research interests include flow theory, online shopping experience, flow and business education, consumer behaviors in electronic commerce, and agent-based systems in knowledge management.

T. J. Giuli received the Ph.D. degree in computer science from Stanford University, Stanford, CA.

He is currently a Research Engineer with Ford Motor Company's Infotronics Research and Advanced Engineering organization, based in Dearborn, MI. His research interests are mobile computing and secure privacy-preserving vehicular software architectures. His recent work involves architecting research software platforms to enable third-party software development in cars.

K. Venkatesh Prasad (SM'08) received the B.E. degree in electronics and communication from the University of Madras, Madras, India, in 1980, the M.S.E.E. degrees from the Indian Institute of Technology, Madras, in 1984 and from Washington State University, Pullman, in 1987, and the Ph.D. degree in electrical and computer engineering from Rutgers University, New Brunswick, NJ, in 1990.

He is the Founder of Ford Motor Company's Infotronics Technologies Group, Dearborn, MI. He is on the advisory board of several private–public academic partnerships involving academia, the government, and industry, including the Ford–Massachusetts Institute of Technology Strategic Alliance and the Department of Electrical Engineering and Computer Science, Northwestern University, Evanston, IL. He is the Chair of the Department of Electrical and Computer Engineering Visiting Committee, Michigan State University, East Lansing. He has more than 30 journal, conference, and book chapter publications. He is the holder of six patents. His research topics include electronic imaging, pattern recognition, associated vehicle system integration, visual information processing, and secure network transactions of digital documents.

Dr. Prasad is a member of the Association for Computing Machinery and the Society of Automotive Engineering International and a Life Member of Sigma Xi (the Scientific Research Society), Eta Kappa Nu (the Electrical Engineering Honor Society), and Tau Beta Pi (the Engineering Honor Society).

Henry Huang received the B.E. and M.E. degrees from Hefei University of Technology, Hefei, China, in 1982 and 1985, respectively, and the M.S. degree from the University of Detroit, Detroit, MI, in 1991.

He is currently working on in-vehicle infotainment system development at Ford Motor Company, Dearborn, MI. He has also worked on electrical/electronic system simulation, dual-voltage electrical architecture design, electrical motor drive simulation/control, electrical/electronic system hardware-in-the-loop simulation/testing, and Bluetooth phone module design.