DESCRIPTION

METHOD AND DEVICE FOR SCALABLE INCREMENTAL WEIGHTED SAMPLING

TECHNICAL FIELD

[0001] Various aspects of this disclosure relate to methods and devices for scalable incremental weighted sampling.

BACKGROUND

[0002] The problem of weighted sampling involves using logical constraints to specify a sampling set, and using a weight function to define a distribution over items in the sampling set. Subsequently, samples are drawn from the distribution defined by the logical constraints and weight function. An incremental setting refers to a setting for which samples are drawn in an incremental manner, with weights being updated at a start of each sampling round.

[0003] Applications involving sampling routines may provide a weight function W and a logical formula F (in a conjunctive normal form) to a sampler, which draws satisfying assignments according to weights specified by the weight function W. For example, a space of valid configurations for a software system may be defined using a logical formula, and subsequently, a sampler may sample configurations from the space. The sampled configurations may then be used as test cases for the software

system. The weighted sampling may be performed to sample test configurations covering a wide range of scenarios.

[0004] Tight integration of sampling routines in applications such as software test generation highlights a need for samplers to handle not only single round sampling settings but also incremental sampling settings in an efficient manner.

SUMMARY

[0005] Various embodiments concern a method for scalable incremental weighted sampling, the method including compiling an input Boolean formula representing a space to be sampled from, to obtain an ordered binary decision diagram[∧] (OBDD[∧]), and parameterizing the obtained OBDD[∧] to obtain a probabilistic OBDD[∧] (PROB). The method further includes smoothing the obtained PROB by adding conjunction nodes to the obtained PROB, and performing the weighted sampling from the smoothed PROB to obtain at least one assignment corresponding to a traversal through the smoothed PROB to a true node included in the smoothed PROB, while performing annotation of the smoothed PROB.

[0006] The parameterizing the obtained OBDD[∧] may include determining a lo normalized weight of a false literal of an input Boolean variable associated with a decision in the input Boolean formula.

[0007] The parameterizing the obtained OBDD[∧] may include determining a hi normalized weight of a true literal of the input Boolean variable associated with the decision in the input Boolean formula.

[0008] The lo normalized weight $\theta_{\mathrm{Lo}(n_d)}$ and the hi normalized weight $\theta_{\mathrm{Hi}(n_d)}$ may be determined based on following equations:

[0009]
$$\theta_{\mathrm{Lo}(n_d)} = \frac{W(\neg x_i)}{W(\neg x_i) + W(x_i)} \quad \theta_{\mathrm{Hi}(n_d)} = \frac{W(x_i)}{W(\neg x_i) + W(x_i)},$$

[0010] where *W*() is a weight function that returns a weight for an input Boolean variable, $\neg x_i$ is a false literal of an input Boolean variable x_i, and x_i is a true literal of the input Boolean variable x_i.

[0011] The method may further include determining whether the obtained PROB is smooth, and the smoothing the obtained PROB may include, based on the obtained PROB being determined to not be smooth, smoothing the obtained PROB.

[0012] The obtained PROB may be determined to not be smooth when a first variable set of a first child node for a decision node included in the PROB is different from a second variable set of a second child node for the decision node.

[0013] The smoothing the obtained PROB may include replacing the first child node and the second child node of the decision node with the conjunction nodes, respectively.

[0014] The smoothing the obtained PROB may further include setting the replaced first child node and the replaced second child node as child nodes of the conjunction nodes, respectively.

[0015] The smoothing the obtained PROB may further include, for a missing variable of each of the conjunction nodes, setting a new decision node representing the missing variable of a respective one of the conjunction modes as a child node of the respective one of the conjunction nodes.

[0016] The smoothing the obtained PROB may further include, for each of the conjunction nodes, setting both child nodes of the new decision node to a true node.

[0017] The weighted sampling from the smoothed PROB and the annotation of the smoothed PROB may be performed in a bottom-up pass of the smoothed PROB, starting at true and false nodes of the smoothed PROB and ending at a root node of the smoothed PROB.

[0018] The performing the weighted sampling from the smoothed PROB may include ignoring one or more false nodes comprised in the smoothed PROB.

[0019] The performing the weighted sampling from the smoothed PROB may include obtaining, as a portion of the at least one assignment, a union of variables for child nodes of each conjunction node included in the smoothed PROB.

[0020] The performing the weighted sampling from the smoothed PROB may include obtaining, as a portion of the at least one assignment, a decision on a Boolean variable associated with each decision node included in the smoothed PROB.

[0021] The performing the annotation of the smoothed PROB may include calculating, for each decision node, a lo joint probability of choosing a false literal of a Boolean variable, and a hi joint probability of choosing a true literal of the Boolean variable, the lo joint probability and the hi joint probability being weighted and being in a log space, and calculating, for each decision node, a sum of the calculated lo joint probability and the calculated hi joint probability, using a LogSumExp function.

[0022] The performing the weighted sampling from the smoothed PROB may further include determining the decision on the Boolean variable associated with each decision node, based on a binomial distribution that is calculated based on the calculated sum of the lo joint probability and the hi joint probability.

[0023] The obtained at least one assignment may correspond to a suitable path for a transport task of a vehicle included in a transport system, and the method may further include controlling the transport system to perform the transport task along the suitable path.

[0024] A server may be configured to perform the method.

[0025] A computer program element may include program instructions, which, when executed by one or more processors, cause the one or more processors to perform the method.

[0026] A computer-readable medium may include program instructions, which, when executed by one or more processors, cause the one or more processors to perform the method.

BRIEF DESCRIPTION OF THE DRAWINGS

[0027] The invention will be better understood with reference to the detailed description when considered in conjunction with the non-limiting examples and the accompanying drawings, in which:

[0028] [Fig. 1] shows a diagram illustrating a communication arrangement for usage of an e-hailing service, including a smartphone and a server;

[0029] [Fig. 2] shows a flow diagram illustrating a method for scalable incremental weighted sampling, according to embodiments;

[0030] [Fig. 3A] shows a diagram of a smoothed $\mathrm{PROB}_{\psi 1}$ including 9 nodes *n1* to *n*9 and representing a Boolean formula $F = (x \vee y) \wedge (\neg x \vee \neg z)$, according to embodiments;

[0031] [Fig. 3B] shows a diagram of a non-smoothed $\mathrm{PROB}_{\psi 2}$ including the 5 nodes *n*1, *n*5, *n*6, *n*8 and *n*9 and representing the Boolean formula $F = (x \vee y) \wedge (\neg x \vee \neg z)$, according to embodiments; and

[0032] [Fig. 4] shows a block diagram of the server of [Fig. 1], implementing the method of [Fig. 2].

DETAILED DESCRIPTION

[0033] The following detailed description refers to the accompanying drawings that show, by way of illustration, specific details and embodiments in which the disclosure may be practiced. These embodiments are described in sufficient detail to enable those skilled in the art to practice the disclosure. Other embodiments may be utilized and structural, and logical changes may be made without departing from the scope of the disclosure. The various embodiments are not necessarily mutually exclusive, as some embodiments may be combined with one or more other embodiments to form new embodiments.

[0034] Embodiments described in the context of one of the devices or methods are analogously valid for the other devices or methods. Similarly, embodiments described in the context of a device are analogously valid for a vehicle or a method, and vice-versa.

[0035] Features that are described in the context of an embodiment may correspondingly be applicable to the same or similar features in the other embodiments. Features that are described in the context of an embodiment may correspondingly be applicable to the other embodiments, even if not explicitly

described in these other embodiments. Furthermore, additions and/or combinations and/or alternatives as described for a feature in the context of an embodiment may correspondingly be applicable to the same or similar feature in the other embodiments.

[0036] In the context of various embodiments, the articles “a”, “an” and “the” as used with regard to a feature or element include a reference to one or more of the features or elements.

[0037] As used herein, the term “and/or” includes any and all combinations of one or more of the associated listed items.

[0038] In the following, embodiments will be described in detail.

[0039] An e-hailing app, typically used on a smartphone, allows its user to hail a taxi or also a private driver through his or her smartphone for a trip.

[0040] [Fig. 1] shows a diagram of a communication arrangement 100 for usage of an e-hailing service, including a smartphone 105 and a server 115 (computer).

[0041] The smartphone 105 includes a screen showing a graphical user interface (GUI) 106 of an e-hailing app that a user of the smartphone 105 previously installed on his smartphone and opened (started) to e-hail a ride (taxi or private driver).

[0042] The GUI 106 includes a map 107 of a vicinity of a position of the user, which the app may determine based on, e.g., a GPS-based location service. Also, the GUI 106 includes a box for a point of departure 108, which may be set to the user’s current location obtained from the location service, and includes a box for a destination 109, which the user may touch to enter the destination, e.g., opening a list of possible destinations. There may further be a menu (not shown) allowing the user to select various options, e.g., how to pay (cash, credit card, credit balance of the e-hailing service). When the user selects the destination and makes any necessary option

selections, he or she touches a “find car” button 110 to initiate searching of a suitable vehicle.

[0043] For the above, the e-hailing app communicates with the server 115 of the e-hailing service via a radio connection. The server 115 may consult a memory of the server 115 or a data storage 121 including information of current locations of registered vehicles 111, of when they are expected to be free, of traffic jams, etc. From this, a processor of the server 115 selects the most suitable vehicle (if available, e.g., if a request may be fulfilled) and provides an estimate of time when a driver will be there to pick up the user, a price of a ride and how long it will take to get to the destination. The server 115 communicates this back to the smartphone 105, and the smartphone 105 displays this information on the GUI 106. The user may then accept (book) by touching a corresponding button. If the user accepts, the server 115 informs a selected one among the vehicles 111 (or, equivalently, its driver), e.g., the vehicle the server 115 allocated for fulfilling the transport request.

[0044] It should be noted while the server 115 is described as a single server, its functionality (e.g., for providing an e-hailing service for a whole city) may in practical application be provided by an arrangement of multiple server computers (e.g., implementing a cloud service). Accordingly, functionalities described in the following provided by the server 115 may be understood to be provided by an arrangement of servers or server computers.

[0045] The data storage 121 may, for example, be part of a cloud-based system 120 provided by a cloud storage provider to store and access data, which it may use for taking decisions, such as information of locations of passengers and vehicles, their history (earlier bookings and routes taken), etc.

[0046] The server 115 together with the vehicles 111 provide the e-hailing service, and form a transport system. It should be noted that while the example of [Fig. 1] relates to an e-hailing service in which persons are transported, a transport service for transporting items like fresh food and parcels may similarly be provided.

[0047] To determine the most suitable driver, an estimate of a time when the driver will be there to pick up the user, a price of a ride and how long it will take to get to the destination, the server 115 may include a memory storing a trained route prediction model that a processor may run to predict a route that will be taken for the ride by the driver allocated to a requested trip. The server 115 may also estimate traffic, using the route prediction model, to predict routes taken by other vehicles and to estimate travelling times based on the estimated traffic.

[0048] For training the route prediction model, the data storage 121 may store map data (i.e. location network data) and historical trip data specifying routes taken between respective points of departure and destinations. The server 115 may, for example, gather this information by recording information of routes taken by drivers of the e-hailing service.

[0049] A task of predicting a route taken by a driver between a point of departure (e.g. a customer's location or a current location of the driver) and a destination (e.g. a customer's desired destination or a customer's location for pickup) may be seen as popular route mining, or a task of finding popular routes on a map (given a point of departure "location A" and a point of arrival "location B"). The task of finding popular routes on the map from location A to location B may be formulated as a sampling problem. A popular route (and thus a predicted route) is then a route with high probability in a distribution of all possible routes between location A and location B.

[0050] Locations may be represented by nodes of a graph. They may correspond to points of interest on a map and/or road segments of a road network (given by map data stored in the memory of the server 115). For example, level 5 geohashes may be used for the nodes of the graph. In another example, an open street map road network may be used as a basis for the nodes of the graph. A location may be a starting location of a trip (a point of departure), an end location of a trip (a destination) or a point via which a route goes (a via-point). A probability distribution may be included over a space of all valid paths (through the graph), and probabilities of routes are learned from training data, e.g. historical trip data.

[0051] In the following embodiments, an OBDD[∧] is used as a graph to represent locations and whether combinations of locations form valid routes. OBDD[∧] is an augmentation of an ordered binary decision diagram (OBDD) with a conjunctive decomposition.

[0052] A sampling algorithm is provided that allows obtaining a route (or path) in one go, rather than in an iterative manner, although both is possible for the OBDD[∧] representation. The OBDD[∧] is augmented with probabilities that are learnt from training data and that allow determining (sampling) popular routes, to predict a route taken between two given locations. Approaches are described in the following that allow for calculating a probability of a route (according to a learnt distribution of routes) and to sample a route based on the learnt route probability.

[0053] Embodiments described herein include methods and devices for scalable incremental weighted sampling. The embodiments may overcome scalability challenges of incremental weighted sampling (e.g., of transport task routes), and may be faster in both weighted sampling and incremental sampling. The embodiments use

a knowledge compilation (KC) paradigm, which seeks to succinctly represent all satisfying assignments of a Boolean formula with a directed acyclic graph (DAG) called an OBDD[∧]. Two additional aspects of the embodiments allow the embodiments to outperform a current weighted sampler. First, the embodiments use a PROB (a probabilistic OBDD[∧]). Second, the embodiments include performing annotation, or joint probability computations, in a log-space to avoid a slower alternative of using arbitrary precision math computations.

[0054] Advantageously, the embodiments may be more scalable and significantly faster than prior art in both single round weighted sampling and incremental weighted sampling. The embodiments may also be fundamental tools in many modern-day applications such as route sampling and software test case generation. Having a scalable sampler may further lead to more performant downstream applications.

[0055] [Fig. 2] shows a flow diagram illustrating a method 200 for scalable incremental weighted sampling, according to embodiments.

[0056] Referring to [Fig. 2], in operation 205, the method 200 includes determining whether an input Boolean formula F (in a conjunctive normal form) representing a space to be sampled from is compiled. Based on the Boolean formula F being determined to not be compiled, the method 200 continues in operation 210. Otherwise, the method 200 continues in operation 230.

[0057] In operation 210, the method 200 includes compiling the Boolean formula F to obtain a KC DAG called OBDD[∧]. The Boolean formula F may be compiled using a KC compiler such as KCBox.

[0058] In operation 215, the method 200 includes parameterizing the obtained OBDD[∧] to obtain a probabilistic OBDD[∧] (PROB). The PROB is a DAG that may include four types of nodes: conjunction; decision; true; and false. Internal nodes of the PROB may consist of conjunction and decision nodes, whereas leaf nodes of the PROB may consist of true and false nodes. The PROB may be recursively made up of sub-PROBs representing sub-formulas of the Boolean formula F. VarSet(*n*) refers to a set of variables of the Boolean formula F represented by the PROB, with node *n* as a root node. Subdiagram(*n*) refers to a sub-PROB starting at the node *n*, and Parent(*n*) refers to an immediate parent of the node *n* in the PROB.

[0059] [Fig. 3A] shows a diagram of a smoothed $\mathrm{PROB}_{\psi 1}$ 300 including 9 nodes *n1* to *n*9 and representing a Boolean formula $F = (x \vee y) \wedge (\neg x \vee \neg z)$, according to embodiments.

[0060] A conjunction node (∧ -node) n_c represents a conjunction in an assignment space. There are no limits to a number of child nodes that the conjunction node n_c may have. However, a set of variables (VarSet(·)) for each child node of the conjunction node n_c must be disjoint.

[0061] Referring to [Fig. 3A], an example of a conjunction node is *n*2. Notice that VarSet(*n*4) = {*z*} and VarSet(*n*5) = {*y*} are disjoint.

[0062] A decision node n_d represents a decision on an associated Boolean variable Var(n_d) in the Boolean formula F represented by the $\mathrm{PROB}_{\psi 1}$ 300. The decision node n_d has exactly two children: lo-child (Lo(n_d)); and hi-child (Hi(n_d)). Lo(n_d) represents the assignment space when the associated Boolean variable Var(n_d) is set to false, which is represented by a dashed line. Hi(n_d) represents the assignment space when the associated Boolean variable Var(n_d) is set to true, which is represented by a

solid line. $\theta_{\mathrm{Hi}(n_d)}$ and $\theta_{\mathrm{Lo}(n_d)}$ refer to parameters associated with the edge connecting decision node n_d and with the assignment spaces Hi(n_d) and Lo(n_d), respectively.

[0063] Referring again to [Fig. 3A], an example of a decision node is $n1$. Notice that Var($n1$) = x, Hi($n1$) = $n3$ and Lo($n1$) = $n2$.

[0064] Referring again to operation 215 of [Fig. 2], the OBDD[∧] is parameterized into the PROB by augmenting each decision node n_d with the two additional parameters $\theta_{\mathrm{Lo}(n_d)}$ and $\theta_{\mathrm{Hi}(n_d)}$. $\theta_{\mathrm{Lo}(n_d)}$ is a normalized weight of a false literal of the Boolean variable ¬Var(n_d), and $\theta_{\mathrm{Hi}(n_d)}$ is a normalized weight of a true literal of the Boolean variable Var(n_d). Let x_i be the Boolean variable Var(n_d). Given a weight function $W(\cdot)$ that returns a weight for an input literal Boolean variable, the parameters $\theta_{\mathrm{Lo}(n_d)}$ and $\theta_{\mathrm{Hi}(n_d)}$ may be calculated using the following equations:

[0065]
$$\theta_{\mathrm{Lo}(n_d)} = \frac{W(\neg x_i)}{W(\neg x_i) + W(x_i)} \quad \theta_{\mathrm{Hi}(n_d)} = \frac{W(x_i)}{W(\neg x_i) + W(x_i)} \quad (1)$$

[0066] ¬x_i is the false literal of the input Boolean variable x_i, and x_i is the true literal of the input Boolean variable x_i.

[0067] The parameters $\theta_{\mathrm{Lo}(n_d)}$ and $\theta_{\mathrm{Hi}(n_d)}$ in the PROB are respectively the normalized weights of the literal Boolean variables in the original Boolean formula.

[0068] Referring again to [Fig. 3A], true (⊤) and false (⊥) nodes are leaf nodes in the $\mathrm{PROB}_{\psi 1}$ 300. Let τ be an assignment of all variables of the Boolean formula F. τ corresponds to a traversal of the $\mathrm{PROB}_{\psi 1}$ 300 from the root node to the leaf nodes. The traversal follows τ at every decision node n_d, and visits all child nodes of every conjunction node n_c encountered along the way. τ is a satisfying assignment if all parts

of the traversal eventually lead to the true node. τ is not a satisfying assignment if any parts of the traversal leads to the false node.

[0069] For example, {*x*,*y*,¬*z*} is a satisfying assignment because following decisions for each variable always leads to the ⊤ node, which is *n*9. In contrast, {*x*,*y*,*z*} is not a satisfying assignment because Hi(*n*6) is the ⊥ node, which is *n*8.

[0070] Referring again to [Fig. 2], in operation 220, the method 200 includes determining whether the obtained PROB is smooth. Based on the obtained PROB being determined to not be smooth, the method 200 continues in operation 225. Otherwise, the method 200 continues in operation 230.

[0071] In operation 225, the method 200 includes smoothing the obtained PROB. The smoothing the obtained PROB_{ψ} may be performed using Algorithm 1 below:

Algorithm 1 Smooth - returns a smoothed PROB

Input: PROB ψ
Output: smooth PROB

```
1: κ ← initMap()
2: for node n of ψ in bottom-up order do
3:    if n is true node or false node then
4:       κ[n] ← ∅
5:    else if n is ∧-node then
6:       κ[n] ← unionVarSet(Child(n), κ)
7:    else
8:       if κ[Hi(n)] − κ[Lo(n)] ≠ ∅ then
9:          lset ← κ[Hi(n)] − κ[Lo(n)]
10:         lcNode ← new ∧ −node()
11:         lcNode.addChild(Lo(n))
12:         for var v in lset do
13:            dNode ← checkMakeTrueDecisionNode(v)
14:            lcNode.addChild(dNode)
15:         Lo(n) ←lcNode
16:      if κ[Lo(n)] − κ[Hi(n)] ≠ ∅ then
17:         rset ← κ[Lo(n)] − κ[Hi(n)]
18:         rcNode ← new ∧ −node()
19:         rcNode.addChild(Hi(n))
20:         for var v in rset do
21:            dNode ← checkMakeTrueDecisionNode(v)
22:            rcNode.addChild(dNode)
23:         Hi(n) ←rcNode
24:      κ[n] ← Var(n) ∪ unionVarSet({Hi(n), Lo(n)})
25: return ψ
```

[0072] Algorithm 1 processes nodes of the input PROB_ψ in a bottom-up manner, while keeping track of a variable set VarSet(*n*) for every node *n* of the PROB_ψ, using a map *κ*. True and false nodes have a value of ∅ in the map *κ* as they are leaf nodes and do not represent any variables. The variable set of each conjunction node (∧-node) is a union of variable sets of its child nodes unionVarSet(Child(*n*), *κ*).

[0073] The smoothing is determined to be performed at a decision node *n* of the PROB_ψ when VarSet(Lo(*n*)) and VarSet(Hi(*n*)) do not contain the same set of variables

as shown by lines 8 and 16 of Algorithm 1. In the smoothing process, new conjunction nodes (lcNode for Lo(*n*) and rcNode for Hi(*n*)) are created to respectively replace child nodes of the decision node *n*, and the replaced child nodes are set as child nodes of the new conjunction nodes, respectively. Additionally, for a missing variable *v* of each of the new conjunction nodes, a new decision node representing the missing variable *v* is created and set as a child node of the respective one of the new conjunction nodes. Each of the new decision nodes created during the smoothing has both its lo-child and hi-child set to a true node. To reduce a memory footprint, it is checked if there exists the same decision node before creating it, in a checkMakeTrueDecisionNode(·) function.

[0074] [Fig. 3B] shows a diagram of a non-smoothed $\mathrm{PROB}_{\psi 2}$ 305 including the 5 nodes *n*1, *n*5, *n*6, *n*8 and *n*9 and representing the Boolean formula $F = (x \vee y) \wedge (\neg x \vee \neg z)$, according to embodiments.

[0075] Referring to [Fig. 3B], the $\mathrm{PROB}_{\psi 2}$ 305 is not smooth, because VarSet(Lo(*n*1)) = {*y*} and VarSet(Hi(*n*1)) = {*z*}.

[0076] Referring to [Figs. 3A and 3B], in a smoothing process, Lo(*n*1) or the node *n*5 is replaced with a new conjunction node *n*2, and the replaced node *n*5 is set as a child node of the new conjunction node *n*2. Also, a new decision node *n*4 representing a missing variable *z* is set as a child node of the new conjunction node *n*2, and both child nodes of the new decision node *n*4 are set to true node *n*9.

[0077] Further, in the smoothing process, Hi(*n*1) of the node *n*6 is replaced with a new conjunction node *n*3, and the replaced node *n*6 is set as a child node of the new conjunction node *n*3. Also, a new decision node *n*7 representing a missing variable *y*

is set as a child node of the new conjunction node *n*3, and both child nodes of the new decision node *n*7 are set to true node *n*9 is added, to arrive at the $\mathrm{PROB}_{\psi 1}$ in [Fig. 3A].

[0078] Advantageously, the smoothing of the PROB may enable probabilities of all assignments to be calculated. If a decision node for a variable is missing, then a sampled assignment would not have the missing variable. The smoothing the PROB refers to adding back the decision node for the missing variable so that the sampled assignment has all variables. Thus, the smoothing of the PROB may improve an output of a sampling algorithm.

[0079] Referring again to [Fig. 2], in operation 230, the method 200 includes performing weighted sampling from the obtained or smoothed PROB to obtain at least one satisfying assignment corresponding to a traversal through the obtained or smoothed PROB to a true node included in the obtained or smoothed PROB, while performing annotation of the obtained or smoothed PROB. In detail, the method 200 may include drawing the at least one satisfying assignment in a bottom-up pass of the obtained or smoothed PROB_{ψ} representing the Boolean formula F (e.g., from a space of satisfying assignments of the Boolean formula F), based on specified weights. The weighted sampling may be performed from the obtained or smoothed PROB_{ψ}, using Algorithm 2 below:

Algorithm 2 INC - returns a satisfying assignment based on PROB ψ parameters

Input: PROB ψ
Output: a sampled satisfying assignment

1: cache $\omega \leftarrow$ initCache()
2: joint prob cache $\varphi \leftarrow$ initCache()
3: $\psi' \leftarrow$ hideFalseNode(ψ)
4: **for** node n of ψ' in bottom-up order **do**
5: **if** n is *true* node **then**
6: $\omega[n] \leftarrow \emptyset$
7: $\varphi[n] \leftarrow \log(1)$
8: **else if** n is $\wedge$-node **then**
9: $\omega[n] \leftarrow \mathsf{unionChild}(\mathsf{Child}(n), \omega)$
10: $\varphi[n] \leftarrow \sum_{c \in \mathsf{Child}(n)} \varphi[c]$
11: **else**
12: $p_{lo} \leftarrow \log(\theta_{\mathsf{Lo}(n)}) + \varphi[\mathsf{Lo}(n)]$
13: $p_{hi} \leftarrow \log(\theta_{\mathsf{Hi}(n)}) + \varphi[\mathsf{Hi}(n)]$
14: $p_{joint} \leftarrow \mathsf{Logsumexp}(p_{lo}, p_{hi})$
15: $\varphi[n] \leftarrow p_{joint}$
16: $r \leftarrow x \sim \mathsf{binomial}(1, \exp(p_{hi} - p_{joint}))$
17: **if** r is 1 **then**
18: $\omega[n] \leftarrow \omega[\mathsf{Hi}(n)] \cup \mathsf{Var}(n)$
19: **else**
20: $\omega[n] \leftarrow \omega[\mathsf{Lo}(n)] \cup \neg\mathsf{Var}(n)$
21: **return** ω[rootnode(ψ)]

[0080] Algorithm 2 includes the weighted sampling in the bottom-up pass of the obtained or smoothed PROB_{ψ}, while integrating an annotation (e.g., joint probability calculations) of the obtained or smoothed PROB_{ψ} in the same bottom-up pass. In detail, to sample from the space of satisfying assignments, false nodes are entirely ignored in the PROB_{ψ} as shown by line 3 of Algorithm 2. For example, referring to [Fig. 3A], when a function hideFalseNode(·) is applied to the $\mathrm{PROB}_{\psi 1}$ 300, node *n*8 and edges immediately leading to it are hidden.

[0081] Next, each of remaining nodes are processed in a bottom-up order, while keeping two caches: ω to store partial samples from each node; and φ to store a joint probability of each node. At each true node, ω takes a value of $\emptyset$ since there is no associated variable. At each conjunction node ($\wedge$-node), ω takes a union of child nodes unionChild(Child(n), ω) as shown in line 9 of Algorithm 2. Referring again to [Fig. 3A], using node $n2$ as an example, if $\omega[n4] = \{\neg z\}$ and $\omega[n5] = \{y\}$, then a union of child nodes unionChild(Child($n2$),ω) = $\{y,\neg z\}$.

[0082] At each decision node n, a decision on a Boolean variable Var(n) is sampled from lines 12 to 20 of Algorithm 2. First, joint probabilities p_{lo} and p_{hi} of respectively choosing ¬Var(n) and choosing Var(n) are calculated. Subsequently, the decision on the Boolean variable Var(n) is sampled using a binomial distribution in line 16 with a probability of success being the joint probability of choosing the Boolean variable Var(n). Finally, the sampled satisfying assignment is output once the root node (e.g., the last node) of the PROB_{ψ} is processed.

[0083] Algorithm 2 may be extended from drawing a single sample to drawing k samples in a single pass, where k is a user-specified number. At each node, in the cache ω, a list of k independent copies of drawn partial assignments is stored. At each conjunction node, the same union process in line 9 of Algorithm 2 is performed for child node outputs in the same indices of respective lists in the cache ω. More specifically, if the conjunction node has child nodes cx and cy, outputs of index i are combined to get an output of the conjunction node at the index i. This is performed for all indices from 1 to k. At each decision node n_d, the k independent samples are drawn, instead of the single sample, from the binomial distribution shown in line 16. The sampling in lines 12 to 20 are performed independently for the k random numbers. There is no

change necessary for calculation of joint probabilities in Algorithm 2 as there is no change in literal weights.

[0084] To sample according to the specified weights, branch parameters of decision nodes in a corresponding PROB are set according to Equations (1) above. Under this setting of the branch parameters, one can view the branch parameters as a probability of choosing between positive and negative literals of a decision node. As a result of structural properties of the PROB, one can calculate joint probabilities of (or annotate) nodes as follows:

[0085]
$$\begin{aligned} P' \text{ of } \wedge\text{-node } n_c &= \prod_{c \in \mathsf{Child}(n_c)} P'(c) \\ P' \text{ of decision-node } n_d &= \theta_{\mathsf{Lo}(n_d)} \times P'(\mathsf{Lo}(n_d)) \\ &\quad + \theta_{\mathsf{Hi}(n_d)} \times P'(\mathsf{Hi}(n_d)) \end{aligned}, \quad (2)$$

[0086] where *P'* refers to a joint probability. That is, a joint probability at a conjunction node is a product of joint probabilities of its immediate child nodes. A joint probability at a decision node is a sum of joint probabilities of its child nodes, the joint probabilities being weighted by the corresponding branch parameters. A joint probability of a true node is 1, and a joint probability of a false node is 0.

[0087] Equation (2) involves computing weighted sums at each decision node. However, multiplying many small numbers (in this case, probabilities between 0 and 1) may lead to very small numbers, which can lead to underflow whereby a product at some point becomes 0 because there is not enough bits or digits after a decimal point to represent it. Afterwards, multiplying anything with 0 would also be 0, which may be problematic.

[0088] Thus, the joint probabilities may be computed in a log-space instead. In detail, a LogSumExp function, as detailed in Equation (3) below, may be used to prevent numerical underflow when performing a weighted sum at each decision node:

$$\begin{aligned}\log(a+b) &= \log(a) + \log(1+\frac{b}{a}) \\ &= \log(a) + \log(1+\exp(\log(b)-\log(a)))\end{aligned}$$

[0089] .

(3)

[0090] In detail, referring to Equation (2), a joint probability at a decision node n_d may be given by $\theta_{\mathrm{Lo}(n_d)}$ × joint probability of Lo(n_d) + $\theta_{\mathrm{Hi}(n_d)}$ × joint probability of Hi(n_d). If this calculation were to performed the log-space, two weighted log joint probabilities, which are termed p_{lo} and p_{hi} in Algorithm 2, are added using the LogSumExp function. Because the LogSumExp function is used, the weighted log probabilities p_{lo} and p_{hi} do not need to be independently exponentiated, which risks running into numerical underflow. Instead, based on Equation (3), a difference of the weighted log probabilities p_{lo} and p_{hi} may be exponentiated, which is more numerically stable. Furthermore, a high computational cost of using arbitrary precision math libraries may be avoided.

[0091] Advantageously, by performing the weighted sampling from the obtained or smoothed PROB, which is much smaller than prior art diagrams, processing efficiency of the current weighted sampling may be improved over prior art. Further, the weighted sampling from the obtained or smoothed PROB uses only a single bottom-up pass of the obtained or smoothed PROB, while the prior art uses two passes of a prior art diagram, e.g., a bottom-up pass for annotation and a top-down pass for

sampling. This may further improve the processing efficiency of the current weighted sampling over prior art.

[0092] In operation 235, the method 200 includes determining whether the method 200 is complete. Based on the method 200 being determined to be complete, the method 200 ends. Otherwise, the method 200 continues in operation 240.

[0093] In operation 240, the method 200 includes updating the weights for incremental weighted sampling.

[0094] In operation 245, the method 200 includes applying the updated weights to the obtained or smoothed PROB. The method 200 then returns to operation 205 so that the incremental weighted sampling is performed again until the method 200 is determined to be complete.

[0095] Advantageously, the method 200 described herein is faster in weighted sampling than prior art. Also, the method 200 is faster than the prior art in both general weighted sampling and incremental weighted sampling. Further, the method 200 is a functional replacement for the prior art in weighted sampling, as both processes draw from the same distribution in the same manner, verifying that the method 200 performs the weighted sampling with the same quality as the prior art.

[0096] The obtained at least one satisfying assignment may correspond to a suitable path for a transport task of a vehicle included in a transport system. The method 200 may further include controlling a device (e.g., the smartphone 105 of [Fig. 1]) to display the suitable path for the transport task via, e.g., a display such as a smartphone screen. Based on the user selecting the displayed suitable path, the device may control a server (e.g., the server 115 of [Fig. 1]) to control the transport system (e.g., the communication arrangement 100 of [Fig. 1]) to perform the transport

task along the suitable path (e.g., by controlling the delivery vehicles 111 of [Fig. 1] to travel along the suitable path to perform the transport task).

[0097] The method 200 of [Fig. 2] is, for example, carried out by the server 115 as illustrated in [Fig. 4].

[0098] [Fig. 4] shows a block diagram of the server 115 of [Fig. 1], implementing the method 200 of [Fig. 2].

[0099] Referring to [Fig. 4], the server 115 may be a server computer that includes a communication interface 405, a processor 410 and a memory 415.

[00100] The communication interface 405 may serve as a hardware and/or software interface that may, for example, transfer commands and/or data between a user and/or external devices and other components of the server 115. The communication interface 405 may further set up communication between the server 115 and the external devices, such as the smartphone 105 of [Fig. 1]. The communication interface 405 may be connected with a network through wireless or wired communication architecture to communicate with the external devices. The communication interface 405 may be a wired or wireless transceiver or any other component for transmitting and receiving signals.

[00101] The processor 410 may include one or more of a CPU, a graphics processor unit (GPU), an accelerated processing unit (APU), a many integrated core (MIC), a field-programmable gate array (FPGA), and/or a digital signal processor (DSP). The processor 410 may be a general-purpose controller that performs control of any one or any combination of the other components of the server 115, and/or performs an operation or data processing relating to communication. The processor 410 may execute one or more programs stored in the memory 415.

[00102] The memory 415 may include a volatile and/or non-volatile memory. The memory 415 stores information, such as one or more of commands, data, programs (one or more instructions), applications, etc., which are related to at least one other component of the server 115 and for driving and controlling the server 115. For example, commands and/or data may formulate an operating system (OS). Information stored in the memory 415 may be executed by the processor 410. The memory 415 may further store information that is executed by the processor 410 to perform functions and operations described with respect to [Figs. 1-3B] above.

[00103] The methods described herein may be performed and the various processing or computation units and the devices and computing entities described herein may be implemented by one or more circuits. In an embodiment, a "circuit" may be understood as any kind of a logic implementing entity, which may be hardware, software, firmware, or any combination thereof. Thus, in an embodiment, a "circuit" may be a hard-wired logic circuit or a programmable logic circuit such as a programmable processor, e.g., a microprocessor. A "circuit" may also be software being implemented or executed by a processor, e.g., any kind of computer program, e.g., a computer program using a virtual machine code. Any other kind of implementation of the respective functions that are described herein may also be understood as a "circuit" in accordance with an alternative embodiment.

[00104] While the disclosure has been particularly shown and described with reference to specific embodiments, it should be understood by those skilled in the art that various changes in form and detail may be made therein without departing from the spirit and scope of the invention as defined by the appended claims. The scope of the invention is thus indicated by the appended claims and all changes that come within

the meaning and range of equivalency of the claims are therefore intended to be embraced.

CLAIMS

[Claim 1] A method for scalable incremental weighted sampling, the method comprising:

compiling an input Boolean formula representing a space to be sampled from, to obtain an ordered binary decision diagram[∧] (OBDD[∧]);

parameterizing the obtained OBDD[∧] to obtain a probabilistic OBDD[∧] (PROB);

smoothing the obtained PROB by adding conjunction nodes to the obtained PROB;

performing the weighted sampling from the smoothed PROB to obtain at least one assignment corresponding to a traversal through the smoothed PROB to a true node comprised in the smoothed PROB, while performing annotation of the smoothed PROB;

updating a plurality of weights for the weighted sampling; and

applying the updated plurality of weights to the smoothed PROB.

[Claim 2] The method of claim 1, wherein the parameterizing the obtained OBDD[∧] comprises determining a lo normalized weight of a false literal of an input Boolean variable associated with a decision in the input Boolean formula.

[Claim 3] The method of claim 2, wherein the parameterizing the obtained OBDD[∧] comprises determining a hi normalized weight of a true literal of the input Boolean variable associated with the decision in the input Boolean formula.

[Claim 4] The method of claim 3, wherein the lo normalized weight $\theta_{\mathrm{Lo}(n_d)}$ and the hi normalized weight $\theta_{\mathrm{Hi}(n_d)}$ are determined based on following equations:

$$\theta_{\mathrm{Lo}(n_d)} = \frac{W(\neg x_i)}{W(\neg x_i) + W(x_i)} \quad \theta_{\mathrm{Hi}(n_d)} = \frac{W(x_i)}{W(\neg x_i) + W(x_i)},$$

wherein *W*() is a weight function that returns a weight for an input Boolean variable, $\neg x_i$ is a false literal of an input Boolean variable x_i, and x_i is a true literal of the input Boolean variable x_i.

[Claim 5] The method of any one of claims 1 to 4, further comprising determining whether the obtained PROB is smooth,

wherein the smoothing the obtained PROB comprises, based on the obtained PROB being determined to not be smooth, smoothing the obtained PROB.

[Claim 6] The method of claim 5, wherein the obtained PROB is determined to not be smooth when a first variable set of a first child node for a decision node comprised in the PROB is different from a second variable set of a second child node for the decision node.

[Claim 7] The method of claim 6, wherein the smoothing the obtained PROB comprises replacing the first child node and the second child node of the decision node with the conjunction nodes, respectively.

[Claim 8] The method of claim 7, wherein the smoothing the obtained PROB further comprises setting the replaced first child node and the replaced second child node as child nodes of the conjunction nodes, respectively.

[Claim 9] The method of claim 8, wherein the smoothing the obtained PROB further comprises, for a missing variable of each of the conjunction nodes, setting a new decision node representing the missing variable of a respective one of the conjunction modes as a child node of the respective one of the conjunction nodes.

[Claim 10] The method of claim 9, wherein the smoothing the obtained PROB further comprises, for each of the conjunction nodes, setting both child nodes of the new decision node to a true node.

[Claim 11] The method of any one of claims 1 to 10, wherein the weighted sampling from the smoothed PROB and the annotation of the smoothed PROB are performed in a bottom-up pass of the smoothed PROB, starting at true and false nodes of the smoothed PROB and ending at a root node of the smoothed PROB.

[Claim 12] The method of any one of claims 1 to 11, wherein the performing the weighted sampling from the smoothed PROB comprises ignoring one or more false nodes comprised in the smoothed PROB.

[Claim 13] The method of any one of claims 1 to 12, wherein the performing the weighted sampling from the smoothed PROB comprises obtaining, as a portion of

the at least one assignment, a union of variables for child nodes of each conjunction node comprised in the smoothed PROB.

[Claim 14] The method of any one of claims 1 to 13, wherein the performing the weighted sampling from the smoothed PROB comprises obtaining, as a portion of the at least one assignment, a decision on a Boolean variable associated with each decision node comprised in the smoothed PROB.

[Claim 15] The method of claim 14, wherein the performing the annotation of the smoothed PROB comprises:

calculating, for each decision node, a lo joint probability of choosing a false literal of a Boolean variable, and a hi joint probability of choosing a true literal of the Boolean variable, the lo joint probability and the hi joint probability being weighted and being in a log space; and

calculating, for each decision node, a sum of the calculated lo joint probability and the calculated hi joint probability, using a LogSumExp function.

[Claim 16] The method of claim 15, wherein the performing the weighted sampling from the smoothed PROB further comprises determining the decision on the Boolean variable associated with each decision node, based on a binomial distribution that is calculated based on the calculated sum of the lo joint probability and the hi joint probability.

[Claim 17] The method of any one of claims 1 to 16, wherein

the method further comprises controlling the transport system to perform the transport task along the suitable path.

[Claim 18] A server configured to perform the method of any one of claims 1 to 17.

[Claim 19] A computer program element comprising program instructions, which, when executed by one or more processors, cause the one or more processors to perform the method of any one of claims 1 to 17.

[Claim 20] A computer-readable medium comprising program instructions, which, when executed by one or more processors, cause the one or more processors to perform the method of any one of claims 1 to 17.