Appendix A. (Anytime Learning of Sum-Product and Sum-Product-Max Networks)

A.1 Proof of Proposition 1: SPMN Validity

The expansion of an SPMN S can be given using the expression $\max_{\sigma} \sum_{k} s_{k,\sigma} \Pi_k(...) U(k, \sigma)$, where $\Pi_k(...)$ is the monomial over the indicator variables and $s_{k,\sigma} \geq 0$ is its coefficient. For evaluating SPMN to get the expansion, the indicator variables that are consistent with the evidence are set to 1 and the remaining to 0. A bottom-up pass is performed by applying the operators at each node on the values of its children. Thus, an SPMN is valid if it is able to correctly compute the MEU for any given evidence. For this purpose, the states x and the monomials given by expansion must be in one-to-one correspondence. Thus, each monomial is non-zero in exactly one state (condition 1), and each state has exactly one non-zero monomial in it (condition 2). So, from condition 2, $S(x)$ would be equal to $\max_{\sigma} s_{x,\sigma} U(x, \sigma)$ corresponding to the monomial that is non-zero. Therefore, $\sum_{x \sim e} S(x) = \sum_{x \sim e} \max_{\sigma} s_{x,\sigma} U(x, \sigma) = \max_{\sigma} \sum_{k} s_{k,\sigma} U(k, \sigma) n_k(e)$, where $n_k(e)$ is the number of states x consistent with e for which $\Pi_k(x) = 1$. But $n_k(e) = 1$ from condition 1 if the state x for which $\Pi_k(x) = 1$ is consistent with the evidence and $n_k(e) = 0$ in all other cases. Thus, we get $\sum_{x \sim e} S(x) = \max_{\sigma} \sum_{k:\Pi_k(e)=1} s_{k,\sigma} U(k, \sigma) = S(e)$ which is the condition for validity. The coefficients $s_{x,\sigma}$ represent the probability distribution $\Phi(x, \sigma)$. As stated previously, $\sum_{x \sim e} S(x) = \sum_{x \sim e} \max_{\sigma} s_{x,\sigma} U(x, \sigma) = \max_{\sigma} \sum_{x \sim e} \Phi(x, \sigma) U(x, \sigma) = \max_{\sigma} EU(e, \sigma)$. Thus, we also have $S(e) = \max_{\sigma} EU(e, \sigma)$ that shows that the expansion of the network maximizes the expected utility.

A.2 Proof of Theorem 2: Validity of ANYTIMESPN Algorithm

The ANYTIMESPN algorithm makes use of the LEARNSPN* algorithm that utilizes the two parameters k and n. We need to show that for any combination of the parameters k and n in the given ranges, the SPN remains complete and decomposable, thus ensuring validity. So, we prove by induction from leaves to the root of the network that the SPNs at each iteration are valid.

Base Case: Validity is trivially true in the case of the leaf nodes since they only represent the univariate distributions and are not affected by the parameters of the algorithm.

Induction Hypothesis: Let n^0 be an internal node having children $n^1, ..., n^l$. Following the notations in Poon and Domingos (Poon and Domingos, 2011), the scope of n^0 is given as V^0, a state of V^0 as x^0, the expansion of the sub-network rooted at n^0 as S^0, and the unnormalized probability of x^0 under S^0 as $\Phi^0(x^0)$. The same notations apply for the other nodes as well. Thus, by induction hypothesis, the expansion of SPN rooted at n^l is $S^l = \sum_{x^l} \Phi^l(x^l) \Pi(x^l)$.

Induction Step: If the node n^0 is the product node in the case of naïve-factorization, then its children $n^1, ..., n^l$ consists of leaf nodes representing univariate distributions of each distinct variable in V. Thus, $V^i \cap V^j = \phi$, where V^i and V^j are the scopes of distinct children of n^0. This ensures decomposability of the node. Also, $S^0 = (\sum_{x^1} \Phi^1(x^1) \Pi(x^1)) \times ... \times (\sum_{x^l} \Phi^l(x^l) \Pi(x^l))$ which is its network polynomial.

Consider the sub-SPN rooted at the sum node n^0. This node could have 2 to k number of children. All these clusters would definitely share the same scope since they are discovered within the same dataset which would ensure completeness irrespective of the parameter k. Thus, if it has

k children, then the expansion $S^0 = w_{01}\sum_{x^1}\Phi^1(x^1)\Pi(x^1) + ... + w_{0k}\sum_{x^k}\Phi^k(x^k)\Pi(x^k)$. If the scopes of all the children are same, $V^0 = V^1 = ... = V^k$, then a one-to-one correspondence is established between the monomials of the expansions S^0 and the states of V^0. Therefore, $S^0 = \sum_{x^0}(w_{01}\Phi^1(x^0) + ... + w_{0k}\Phi^k(x^0))\Pi(x^0)$. This represents the network polynomial and maintains validity.

Now consider the case of n^0 being a product node. Let the independence testing performed on the subset of first n variables in V^0 find l independent subsets having scopes $V^1, ..., V^l$ where $l \leq n$. Since the subsets are independent of each other, $V^i \cap V^j = \phi$, for all distinct subset pairs V^i and V^j. The distribution of remaining $|V| - n$ variables ensures that the subsets are still disjoint and the product node n^0 having children of scopes $V^1, ..., V^l$ is decomposable. Thus, the expansion of sub-SPN rooted at n^0, $S^0 = (\sum_{x^1}\Phi^1(x^1)\Pi(x^1)) \times ... \times (\sum_{x^l}\Phi^l(x^l)\Pi(x^l))$, is its network polynomial since $V^i \cap V^j = \phi$ for all distinct subset pairs V^i and V^j.

Thus, the introduction of the parameters k and n to the structure learning algorithm for SPNs do not interfere with the completeness and decomposability of the internal nodes irrespective of the parameter values. Since the network remains complete and decomposable at each iteration of the ANYTIMESPN algorithm, the algorithm always generates a valid network structure.

A.3 Proof of Theorem 3: Validity of ANYTIMESPMN Algorithm

The ANYTIMESPMN algorithm makes use of the LEARNSPMN* algorithm that requires the parameters k, n, d and d_{max} to learn SPMNs. We need to show that for any combination of the parameters in the given ranges, the SPMN structures hold the properties that ensure validity. So, we prove that the SPMNs at each iteration are valid by using induction from leaves to the root of the network.

Base Case: Validity is trivial in the case of the leaf nodes since they only represent the univariate distributions or the utility values and are not affect by the parameters of the algorithm. For the base case of validity, consider the case of a SPMN having partial order $P^{\prec} = [[D], [U]]$. This case only has one decision node connected to d utility nodes that are leaves. The evaluation of this SPMN gives $S = max_{v_g}U(v_g)$ which maximizes the utility for the decision groups.

Induction Hypothesis: Let n^0 be an internal node having children $n^1, ..., n^p$. Let the scope of n^0 be V^0, σ^0 be a policy using decisions in the scope, a state of V^0 be x^0, let the expansion of the sub-network rooted at n^0 be S^0, and the unnormalized probability of x^0 under S^0 be $\Phi^0(x^0, \sigma^0)$. The same notations apply for the other nodes as well. Thus, by induction hypothesis, the scope of sub-SPN rooted at n^p is $S^p = \max_{\sigma^p}\sum_{x^p}\Phi^p(x^p, \sigma^p)U(x^p, \sigma^p)$.

Induction Step: Thus, if n^0 is a max node for decision variable D^0 having d children corresponding to the decision value groups in v_g, the expansion $S^0 = max_{v_g}\{S^1, ..., S^d\}$. By induction hypothesis, $S^1, ..., S^d$ return the MEUs yielded by sub-SPMNs rooted at nodes $n^1, ..., n^d$ that are learned from data having the same scope V_R. Thus, maximization over these values returns the MEU at the max node and the node n^0 is max-complete. The decision variable D^0 is processed at the node n^0 and $D^0 \notin V_R$. Since the children $n^1, ..., n^d$ have the scope V_R, the decision variable D^0 appears at the most once in the paths from the root to the leaves. Hence, max-uniqueness is also ensured.

If the node n^0 is a sum node, it could have 2 to k number of children. The dataset D is clustered into k clusters by considering the variables in the current information set $P^{\prec}[i]$, but the scope of the clusters remains identical. Thus, sum-completeness is guaranteed irrespective of the value of k. For the sub-SPMN rooted at sum node n^0 with k children, the expansion is $S^0 = w_{01}S^1 + ... + w_{0k}S^k$. Therefore,

$$S^0 = w_{01} \times \max_{\sigma^1} \sum_{x^1} \Phi^1(x^1, \sigma^1)U(x^1, \sigma^1) + ... + w_{0k} \times \max_{\sigma^k} \sum_{x^k} \Phi^k(x^k, \sigma^k)U(x^k, \sigma^k)$$

If the scopes of all the children are same, $V^0 = V^1 = ... = V^k$, then a one-to-one correspondence is established between the monomials of the expansions S^0 and the states of V^0. Therefore,

$$S^0 = \max_{\sigma^0} \sum_{x^0} [w_{01}\Phi^1(x^0, \sigma^0) + ... + w_{0k}\Phi^k(x^0, \sigma^0)]U(x^0, \sigma^0))$$

This expression too maximizes the expected utility at the sum node n^0 and so the sub-SPMN rooted at this node remains valid.

Now consider the case of n^0 being a product node. Let the variable splitting performed on the subset of first n variables in V^0 find p independent subsets having scopes $V^1, ..., V^p$ where $p \leq n$. Since the subsets are independent of each other, $V^i \cap V^j = \phi$ for all distinct subset pairs V^i and V^j. The subsets having variables in V_R are merged together and the remaining variables are distributed as mentioned. Due to this, only one branch of the product node has decision or utility variables in its scope and the product node n^0 having children of scopes $V^1, ..., V^q$ is decomposable. Hence, the expansion of V^0 is,

$$\begin{aligned} S^0 &= S^1 \times ... \times S^{q-1} \times (\max_{\sigma^q} \sum_{x^q} \Phi^q(x^q, \sigma^q)U(x^q, \sigma^q)) \\ &= \sum_{x^1} \Phi^1(x^1) \times ... \times \sum_{x^{q-1}} \Phi^{q-1}(x^{q-1}) \times (\max_{\sigma^q} \sum_{x^q} \Phi^q(x^q, \sigma^q)U(x^q, \sigma^q)) \\ &= \max_{\sigma^0} \sum_{x^0} \Phi^0(x^0, \sigma^0)U(x^0, \sigma^0) \end{aligned}$$

This is because $V^i \cap V^j = \phi$ for all distinct subset pairs V^i & V^j and $V^0 = V^1 \cup ... \cup V^q$. Hence, the policy $\sigma^0 = \sigma^q$ and $\Phi^0(x^0, \sigma^0) = \Phi^1(x^1)...\Phi^{q-1}(x^{q-1})\Phi^q(x^q, \sigma^q)$. So, the sub-SPN rooted at the product node n^0 is valid.

If the depth of the branch from the mentioned point exceeds d_{max}, then the variables V are naïve-factorized. We had already proved validity for naïve factorization in Section A.2.

Thus, the parameters k, n, d and d_{max} for the flexible structure learning algorithm LEARN-SPMN* for SPMNs do not interfere with properties required for validity irrespective of the parameter values. Since the network remains sum-complete, decomposable, max-complete and max-unique at each iteration of the ANYTIMESPMN algorithm, the algorithm always generates a valid network structure.

A.4 Modifications to the SPMN Evaluation Testbed

Since most of the RDDLSim domains model a continuous interaction with the environment rather than having a finite goal state, a few modifications have been made to these domains for conducting our experiments. In the case of the *Elevators* domain with three floors, a person may randomly show up at the middle floor with a given probability and may wish to go to the top or the bottom floor. In the modified domain, the start state has the elevator at the first floor and a person on the second floor waiting to go up. Here, the goal is to take that person to the third floor which has a reward of 5.0.

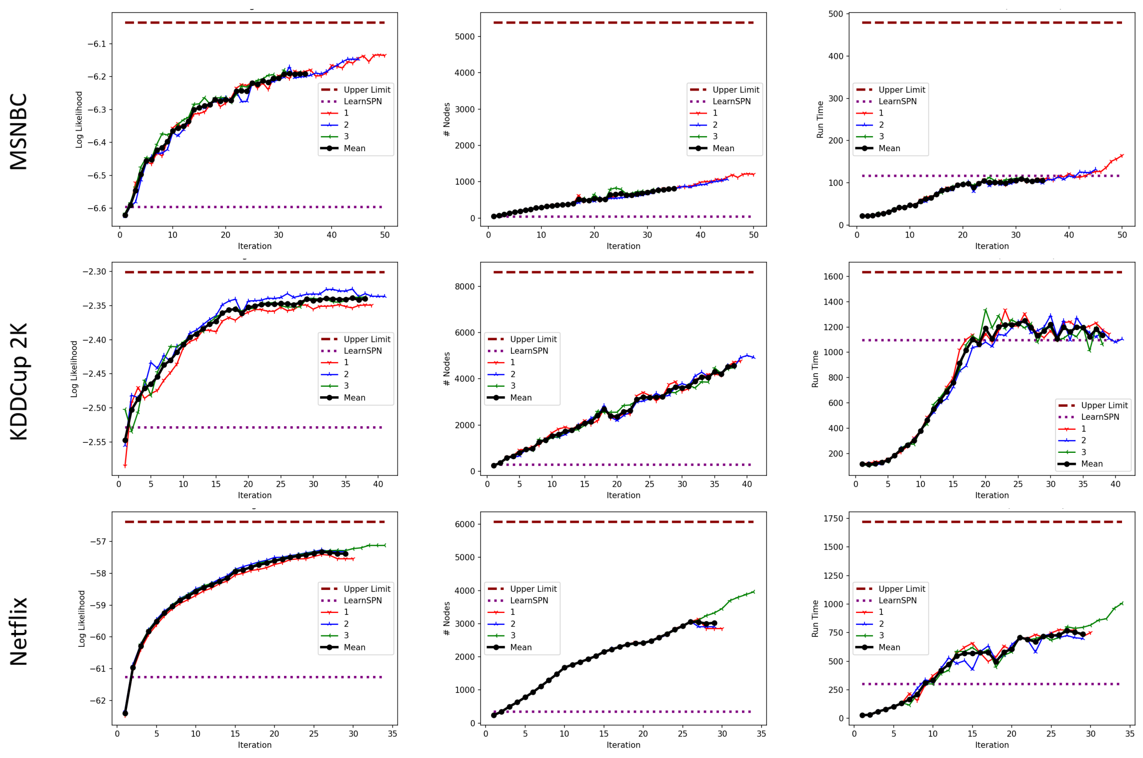

Figure 5: Performance profiles of ANYTIMESPN on the remaining 3 datasets. Left column shows the log likelihoods, middle column the number of nodes in the SPNs, and the right column presents the run times in seconds. Results for the remaining data sets are given in the supplementary file. All experiments were run on a Red Hat Linux 8 system with Intel Xeon 12 cores 1.63 GHz each and 16 GB of RAM.

The *Navigation* domain, originally having a 3×3 grid, is converted to a domain with 3×2 grid by removing the middle column. The initial state of the *CrossingTraffic* domain, having a 3×3 grid, is modified to have the robot starting from the cell that is diagonally opposite to the goal cell. The *GameOfLife* and the *SkillTeaching* domains are kept unchanged.

A.5 Performance of ANYTIMESPN on the Remaining Datasets

Figure 5 presents additional performance profiles generated by the ANYTIMESPN algorithm for the *MSNBC*, *KDDCup 2K*, *Netflix* datasets.

A.6 Performance of ANYTIMESPMN on the Remaining RDDLSim Datasets

Figure 6 presents the additional results generated by the ANYTIMESPMN algorithm for the *Crossing Traffic* and the *Skill Teaching* datasets.

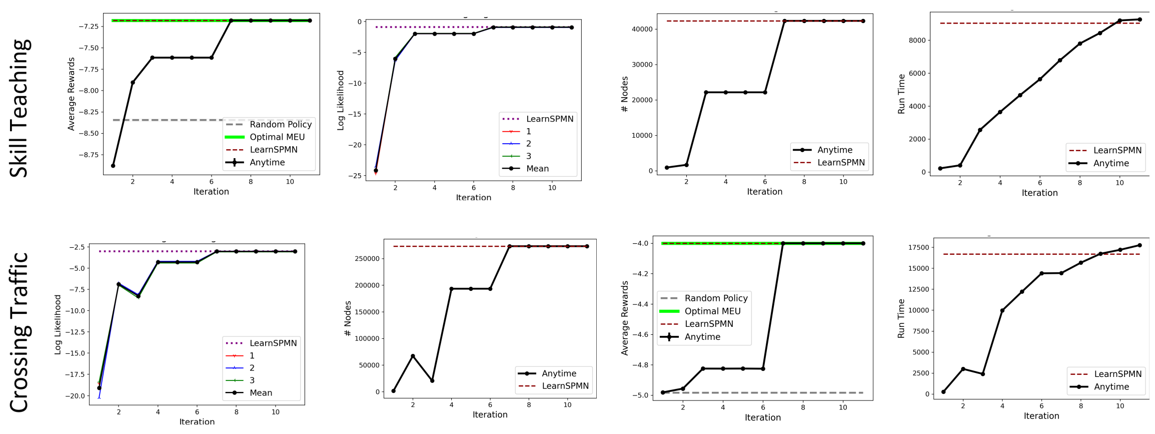

Figure 6: Performance profiles of the AnytimeSPMN algorithm on the remaining two decision-making data sets. We show, in order from left to right columns, the average rewards (with standard deviations), log likelihoods, number of nodes in the learned SPMNs, and the learning run time in seconds.