Nested Aggregates in Answer Sets
Application to a Priori Optimization

Emad Saad
emsaad@gmail.com

Abstract

We allow representing and reasoning in the presence of nested multiple aggregates over multiple variables and nested multiple aggregates over functions involving multiple variables in answer sets, precisely, in answer set optimization programming and in answer set programming. We show the applicability of the answer set optimization programming with nested multiple aggregates and the answer set programming with nested multiple aggregates to the Probabilistic Traveling Salesman Problem, a fundamental a priori optimization problem in Operations Research.

1 Introduction

A fruitful approach to aggregates in answer set programming has been presented in [Faber et al., 2010] that allows to represent and reason in the presence of aggregate function which are defined through aggregates atoms that are allowed to be recursive. The aggregate functions defined in [Faber et al., 2010], and hence the aggregate atoms defined over them, are allowed to be monotone, anti-monotone and non-monotone which is clearly appropriate for non-monotonic reasoning. The answer set semantics with aggregates defined in [Faber et al., 2010] have shown to be appropriate for many interesting problems in answer set programming that require aggregates over some numerical criteria imposed by the problems and arise in many domains.

On the other hand aggregates have been considered in answer set optimization programming described in [Saad and Brewka, 2011] that extended the answer set optimization programs [Brewka et al., 2003] to allow representing and declaratively solving multi-objective optimization problems in an answer set programming framework. The answer set optimization with aggregates framework of [Saad and Brewka, 2011] have shown to be able to find Nash equilibrium in strategic games with any number of players and with any number of strategies, which is a multi-objective optimization problem with multiple conflicting goals.

However the aggregates defined in the answer set programming framework in [Faber et al., 2010] allow a single aggregation over a single variable which limits its applicability to many interesting problems, especially the problems that require aggregations over a function with multiple variables. And hence, more generally limits its applicability to the problems that require nested multiple aggregations over multiple variables and the problems that require nested multiple aggregations over functions with multiple variables.

On the other hand, although the aggregates defined in answer set optimization programming [Saad and Brewka, 2011], is similar to the aggregates defined in the answer set programming [Faber et al., 2010], allows a single aggregation over a single variable, the answer set optimization programming in [Saad and Brewka, 2011] is still capable of solving some interesting multi-objective optimization problems like the one arises from finding Nash equilibrium in strategic games. However, the answer set optimization programming of [Saad and Brewka, 2011] is still incapable of representing and reasoning about many interesting optimization problems that require optimization over objective functions that involve nested multiple aggregation over functions that contain multiple variables. This implies minimization or maximization over nested multiple aggregations over functions with multiple variables or nested multiple aggregations over multiple variables.

In this paper we generalize both answer set optimization programming [Saad and Brewka, 2011] and answer set programming [Faber et al., 2010] with nested multiple aggregates to allow representing and reasoning in the presence of nested multiple aggregation over multiple variables and nested multiple aggregation over functions with multiple variables. We show the applicability of the nested multiple aggregates in answer sets to the Probabilistic Traveling Salesman Problem (PTSP), a fundamental stochastic optimization problem in Operations Research [Jaillet, 1988], whose objective function involving nested multiple aggregations over a function with multiple variables.

Probabilistic Traveling Salesman Problem is a priori optimization problem for which we want to find a priori a tour with minimum (expected) length through a set of
points, where only \( k \) points out of \( n \) points \((0 \leq k \leq n)\) at any given instance of the problem must be visited, where the number \( k \) is chosen at random with a known probability distribution, that is based on the probability distribution of visiting each point in the set of \( n \) points. In addition, the \( k \) points chosen to be visited, in a given instance of the problem, has to be visited in the same order as they appear in the a priori tour.

Formally, the Probabilistic Traveling Salesman Problem is defined as follows. A Probabilistic Traveling Salesman Problem (PTSP), \( G \), is a tuple of the form \( G = (V, E, D, P) \), where \((V, E)\) is a graph with a set of \( n \) nodes, \( V \), and a set of arcs, \( E \), connecting the nodes in \( V \). \( D \) is an \( n \times n \) (complete) matrix (the distance matrix) that specifies the distance (cost), \( d_{ij} \), incurred by traveling from a node (location) \( i \) to a node (location) \( j \). \( P = \{p_1, p_2, \ldots, p_n\} \) is a probability distribution over the nodes in \( V \), where \( p_i \) is the probability of a node \( i \) must be visited. In other words, the nodes in \( V \) represent the set of locations that need to be visited, arcs in \( E \) represent the roads connecting the locations, the values \( d_{ij} \) in \( D \) represent the distance traveling from location \( i \) to location \( j \), and the probabilities \( p_i \) in \( P \) represent the probability that a location \( i \) must be visited. The objective is to find a priori tour \( t \), a Hamiltonian circuit of \( G \), with the minimum expected length (cost). By indexing the nodes of a tour \( t \) by their order of appearance, we present the tour, \( t \), as the sequence of nodes \( t = (0, 1, 2, 3, 4, \ldots, n, 0) \), where 0 is the depot or the starting point from which the tour always starts and ends. It has the index 0 and with probability \( p_0 = 1 \), since it must be always visited. Therefore, the expected length, \( L_t \), of a tour \( t \), denoted by \( E[L_t] \), is given by:

\[
E[L_t] = \sum_{i=0}^{n} \sum_{j=i+1}^{n+1} d_{ij}p_ip_j \left( \prod_{k=i+1}^{j-1} (1 - p_k) \right) \tag{1}
\]

where \( i, j, k \) are indices of nodes within a tout \( t \) (not the nodes themselves) and \( n + 1 = 0 \). Given that \( T \) is the set of all tours in \( G \), the objective is to find a tour \( t \) in \( T \) with the minimum expected length \( E[L_t] \), i.e., solve the a priori optimization problem

\[
\min_{t \in T} E[L_t].
\]

In this paper we develop syntax and semantics for both answer set optimization programs and answer set programs to allow representing and reasoning in the presence of nested multiple aggregates over multiple variables and nested multiple aggregates over functions involving multiple variables. We show the applicability of both the answer set optimization programs with nested multiple aggregates and the answer set programs with nested multiple aggregates to the fundamental stochastic optimization Probabilistic Traveling Salesman Problem, a priori optimization problem in Operation Research [Jaillet, 1988]. In addition, we show that with minor modification to the probabilistic traveling salesman problems representation in answer set optimization programs with nested multiple aggregates and in answer set programs with nested multiple aggregates, a corresponding classical traveling salesman problems can be intuitively represented and solved in both frameworks. We prove that the presented answer set optimization programs with nested multiple aggregates modify and generalize the answer set optimization programs with a single aggregate over a single variable proposed in [Saad and Brewka, 2011] as well as a generalization of the answer set optimization programs described in [Brewka et al., 2003]. In addition, we prove that the presented answer set programs with nested multiple aggregates generalize the answer set programs with a single aggregate over a single variable described in [Faber et al., 2010] as well as a generalization of the original answer set programs presented in [Gelfond and Lifschitz, 1991].

2 Nested Aggregates Disjunctive Logic Programs

In this section we present the syntax and the answer set semantics of disjunctive logic programs (a form of answer set programming) with nested multiple aggregates, denoted by NDLP. The syntax and the answer set semantics of NDLP generalize and modifies the syntax and semantics of disjunctive logic programs with a single aggregate over a single variable, DLP\(^A\), presented in [Faber et al., 2010].

2.1 NDLP Programs Syntax

Let \( \mathcal{L} \) be a first-order language with finitely many predicate symbols, function symbols, constants, and infinitely many variables. A term is a constant, a variable or a function. An atom is a predicate in \( B_\mathcal{L} \), where \( B_\mathcal{L} \) is the Herbrand base of \( \mathcal{L} \). The Herbrand universe of \( \mathcal{L} \) is denoted by \( U_\mathcal{L} \). Non-monotonic negation or the negation as failure is denoted by \( \not \).

An expression of the form \( \{ F \mid C \} \) is called a symbolic set, where \( F \) is a single variable or a function \( F = g(X_1, \ldots, X_n) \), where \( X_1, \ldots, X_n \) are variables or functions and \( C \) is a conjunction of atoms, negation of atoms, aggregate atoms, and negation of aggregate atoms (defined below). A ground set is a set of pairs of the form \( \{ F_G \mid C_G \} \) such that \( F_G \) is a constant and \( C_G \) is a conjunction of ground atoms, negated ground atoms, ground aggregate atoms, and negated ground aggregate atoms (defined below). A ground set or a symbolic set is called a set term.

We say \( f(S) \) is an aggregate function if \( f \) is an aggregate function symbol and \( S \) is a set term, where \( f \in \{ \min, \max, \text{count}, \text{sum}, \text{times} \} \). If \( f(S) \) is an aggregate function and \( T \) is a constant, a variable or a function term called guard, then we say \( f(S) \prec T \) is an aggregate atom, where \( \prec \in \{ =, \neq, <, >, \leq, \geq \} \).

Observe that, from the definition of aggregate functions and aggregate atoms, it can be seen that an aggregate function \( f(S) \) in an aggregate atom \( f(S) \prec T \) can be defined in terms of other aggregate atoms \( f'(S') \prec T' \) and hence in terms of other aggregate functions \( f'(S') \).
This means multiple level of nested aggregate functions calculation is achieved by allowing aggregate functions of the aggregate atoms to be defined in terms of other aggregate atoms. This is accomplished by allowing the conjunction in the definition of the symbolic set $S$ of the aggregate function $f(S)$ in the aggregate atom $f(S) \leftarrow T$ to contain other aggregate atoms as constituents. This can be illustrated by the following example.

**Example 1** Assume that we want to represent the double summation
\[
\sum_{i,j=1}^{n,m} (i+j)
\]
as an aggregate function, $f(S)$, and assign the result to a variable called $X$ to form the aggregate atom $f(S) = X$. Assume also that the possible values of $i, j$ are represented by the predicate $d(I,J)$. Therefore, the double summation (2) can be represented as the aggregate atom $f(S) = X$ that is defined as:

\[
\text{sum} \{ A \mid 1 \leq I \leq n, \text{sum} \{ I + J \mid d(I,J), 1 \leq J \leq m, \} = A \} = X
\]
where $S = \{ A \mid 1 \leq I \leq n, \text{sum} \{ I + J \mid d(I,J), 1 \leq J \leq m, \} = A \}$.

Notice that there are two instances of the aggregate function $\text{sum}$ in the aggregate atom representation of the double summation (2). The outer instance of the aggregate function $\text{sum}$ and the inner instance of the aggregate function $\text{sum}$. The inner instance of the aggregate function $\text{sum}$ is defined inside the outer instance of the aggregate function $\text{sum}$.

Moreover, notice that the variable $J$ occurs only in the inner instance of the aggregate function $\text{sum}$. Whereas the variable $I$ occurs in both the outer and the inner instances of the aggregate function $\text{sum}$. This means that the variable $J$ is seen only by the inner aggregate function $\text{sum}$, but not seen by the outer aggregate function $\text{sum}$, and $I$ is seen by both the outer and the inner aggregate function $\text{sum}$. This implies that the occurrence of $I$ in the outer $\text{sum}$ hides the occurrence of $I$ in the inner $\text{sum}$, which means that any substitution of $I$ to a constant in both the inner and the outer $\text{sum}$ is made by the outer $\text{sum}$. This also implies that the scope of the variable $J$ is the inner $\text{sum}$ while the scope of the variable $I$ is both the outer and the inner $\text{sum}$, i.e., the entire symbolic set of the outer $\text{sum}$. This motivates the following definition of local variables to aggregate functions.

**Definition 1** Let $f(S)$ be an aggregate function. A variable, $X$, is a local variable to $f(S)$ if and only if $X$ appears in $S$ and $X$ does not appear in any aggregate function that is outer to $f(S)$ or in the NDLP rule that contains $f(S)$.

Definition (11) specifies that every aggregate function $f(S)$ has its own set of local variables. For example, for the aggregate atom representation of the double summation described in Example (11), the variable $I$ is local variable to the outer aggregate function $\text{sum}$ while the variable $J$ is local variable to the inner aggregate function $\text{sum}$.

**Definition 2** A global variable is a variable that is not a local variable.

**Definition 3** An NDLP program, $P$, is a set of NDLP rules of the form
\[
a_1 \lor a_2 \lor \ldots \lor a_k \leftarrow a_{k+1}, \ldots, a_m, a_m, a_m, \ldots, a_n
\]
where $a_1, a_2, \ldots, a_k$ are atoms and $a_{k+1}, \ldots, a_m$ are atoms or aggregate atoms.

Let $r$ be an NDLP rule of the form (4). We use $\text{head}(r) = a_1 \lor a_2 \lor \ldots \lor a_k$ and $\text{body}(r) = a_{k+1}, \ldots, a_m, a_m, \ldots, a_n$.

### 2.2 NDLP Programs Semantics

**Definition 4** The local ground instantiation of a symbolic set $S = \{ F \mid C \}$ is the set of all local ground pairs of the form $\langle F(\theta) \rangle$ or $\langle C(\theta) \rangle$, where $\theta$ is a substitution of every local variable appearing in $S$ to a constant from $U_C$.

**Definition 5** Let $S$ be a symbolic set. Then, the ground instantiation of $S$ is the local ground instantiation of $S$ followed by the local ground instantiation of every symbolic set, $S'$, appearing in $S$, then followed by the local ground instantiation of every symbolic set, $S''$, appearing in $S'$, then etc.

**Example 2** Consider the grounding of the aggregate function representation of the double summation presented in Example (11) and described by (3), where $n = m = 2$ and with the following facts added.

\[
d(1,1), \ d(1,2), \ d(2,1), \ d(2,2) \quad e(3,1)
\]

Since the aggregate function in (3) contains two levels of nesting of the aggregate function $\text{sum}$, then the grounding is achieved in two steps. The first step is described as follows, where $A_1$, $A_2$, and $A_3$ are variables act as place holders that are replaceable by constants.

\[
\text{sum}\{ \langle A_1 \mid 1 \leq I \leq 2, \rangle, \text{sum}\{ 1 + J \mid d(1,J), 1 \leq J \leq 2 \rangle = A_1 \}, \langle A_2 \mid 1 \leq 2 \leq 2, \rangle, \text{sum}\{ 2 + J \mid d(2,J), 1 \leq J \leq 2 \rangle = A_2 \}, \langle A_3 \mid 1 \leq 3 \leq 2, \rangle, \text{sum}\{ 3 + J \mid d(3,J), 1 \leq J \leq 2 \rangle = A_3 \}
\]

followed by the second step which is described as:

\[
\text{sum}\{ \langle A_1 \mid 1 \leq 2 \leq 2, \rangle, \text{sum}\{ (1 + 1)d(1,1), 1 \leq I \leq 2, \}, (1 + 3)d(1,3), 1 \leq 3 \leq 2, \) \} = A_1 \),
\[
\langle A_2 \mid 1 \leq 2 \leq 2, \), \text{sum}\{ (2 + 1)d(2,1), 1 \leq I \leq 2, \}, (2 + 3)d(2,3), 1 \leq 3 \leq 2, \) \} = A_2 \),
\[
\langle A_3 \mid 1 \leq 3 \leq 2, \), \text{sum}\{ (3 + 1)d(3,1), 1 \leq I \leq 2, \}, (3 + 3)d(3,3), 1 \leq 3 \leq 2, \) \} = A_3 \)
\]
Definition 6 A ground instantiation of an NDLP rule, \( r \), is the replacement of each global variable appearing in \( r \) to a constant from \( U_L \), then followed by the ground instantiation of every symbolic set, \( S \), appearing in \( r \).

The ground instantiation of an NDLP program, \( \Pi \), is the set of all possible ground instantiations of every NDLP rule, \( r \), in \( \Pi \).

Let \( X \) be a set of objects. Then, we use \( \mathcal{X} \) to denote the set of all multisets over elements in \( X \). The semantics of the aggregate functions \( \min, \max, \text{count}, \text{sum} \), and \( \text{times} \) are defined by the mappings: \( \min, \max : (\mathcal{X} - \emptyset) \to \mathbb{R} \); \( \text{count} : \mathcal{X}^U_L \to \mathbb{N} \); \( \text{sum} : \mathcal{X}^k \to \mathbb{R} \), and \( \text{times} : \mathcal{X}^m \to \mathbb{R} \), where \( \mathbb{R} \) is the set of all real numbers, and \( U_L \) is the Herbrand universe. The application of \( \text{sum} \) and \( \text{times} \) on the empty multiset returns zero and one respectively. The application of \( \text{count} \) on the empty multiset returns zero. However, the application of \( \max \) and \( \min \) on the empty multiset is undefined. Let \( \perp \) be a symbol that does not occur in any NDLP program.

Definition 7 An interpretation is a subset of the Herbrand base \( B_L \).

An atom, \( a \), is true (satisfied) with respect to an interpretation, \( I \), if \( a \) belongs to \( I \); but it is false (unsatisfied) otherwise. The negation of an atom, not \( a \), is true (satisfied) with respect to \( I \) if \( a \) does not belong to \( I \); but it is false (unsatisfied) otherwise. Similarly, the evaluation of an aggregate function, and hence the truth valuation of an aggregate atom, are established with respect to a given interpretation, \( I \), as described by the following definitions.

Definition 8 Let \( f(S) \) be a ground aggregate function and \( I \) be an interpretation. Then, we define \( S_I \) to be the multiset constructed from elements in \( S \), where \( S_I = \{ F_C \mid \{ F_C \mid C_G \} \in S \wedge C_G \text{ is true in } I \} \).

Definition 9 Let \( f(S) \) be a ground aggregate function and \( I \) be an interpretation. Then, the evaluation of \( f(S) \) with respect to \( I \) is \( f(S_I) \), the result of the application of \( f \) to \( S_I \). \( f(S_I) = \perp \) if \( S_I \) is not in the domain of \( f \).

Example 3 Let \( I = \{ d(1, 1), d(1, 2), d(2, 1), d(2, 2), e(3, 1) \} \) be an interpretation. Thus, the evaluation of the aggregate function representing the double summation in Example 2 is evaluated w.r.t. \( I \) as follows in two steps. Considering only the relevant possible values of the variables \( A_1, A_2, \) and \( A_3 \) in Example 2, the first step is to evaluate the inner instance of the aggregate function \( \text{sum} \) as follows:

\[
\text{sum} \{ 5 \mid (1 \leq i \leq 2), \text{sum} \{ 2, 3 \} = 5 \}, \text{sum} \{ 7 \mid (1 \leq i \leq 2), \text{sum} \{ 3, 4 \} = 7 \}
\]

followed by evaluating the second instance of the aggregate function \( \text{sum} \) as:

\[
\text{sum} \{ 5, 7 \} = 12 = \sum_{i=1}^{2} \sum_{j=1}^{2} (i + j)
\]

Definition 10 Let \( f(S) \prec T \) be a ground aggregate atom and \( I \) be an interpretation. Then, \( f(S) \prec T \) is true (satisfied) with respect to \( I \) if and only if \( f(S_I) \neq \perp \) and \( f(S_I) \prec T \). Furthermore, not \( f(S) \prec T \) is true (satisfied) with respect to \( I \) if and only if \( f(S_I) = \perp \) or \( f(S_I) \neq \perp \) and \( f(S_I) \prec T \).

Definition 11 Let \( \Pi \) be a ground NDLP program, \( r \) be a ground NDLP rule of the form (4), and \( I \) be an interpretation. Then,

- \( I \) satisfies \( \text{head}(r) \) iff \( \exists i \ (1 \leq i \leq k) \) such that \( I \) satisfies \( a_i \).
- \( I \) satisfies \( \text{body}(r) \) iff \( \forall (k + 1 \leq i \leq m) \ I \) satisfies \( a_i \) and \( \forall (m + 1 \leq j \leq n) \ I \) satisfies not \( a_j \).
- \( I \) satisfies \( r \) if \( I \) satisfies \( \text{head}(r) \) whenever \( I \) satisfies \( \text{body}(r) \) or \( I \) does not satisfy \( \text{body}(r) \).
- \( I \) satisfies \( \Pi \) iff \( I \) satisfies every NDLP rule, \( r \), in \( \Pi \).

2.3 Answer Sets

A model for an NDLP program, \( \Pi \), is an interpretation that satisfies \( \Pi \). A model \( I \) of \( \Pi \) is \( \subseteq \)-minimal if and only if there does not exist a model \( I' \) of \( \Pi \) such that \( I' \subset I \).

Definition 12 Let \( \Pi \) be a ground NDLP program, \( r \) be an NDLP rule in \( \Pi \), and \( I \) be an interpretation. Let \( I \models \text{body}(r) \) denotes that \( I \) satisfies \( \text{body}(r) \). Then, the reduct, \( \Pi' \), of \( \Pi \) w.r.t. \( I \) is the ground NDLP program \( \Pi' \) where

\[
\Pi' = \{ \text{head}(r) \leftarrow \text{body}(r) \mid r \in \Pi \wedge I \models \text{body}(r) \}
\]

The reduct \( \Pi' \) of \( \Pi \) w.r.t. \( I \) excludes all rules \( r \in \Pi \) whose \( \text{body}(r) \), is not satisfied by \( I \). The satisfaction of \( \text{body}(r) \) in the definition of the reduct does not distinguish between atoms or aggregate atoms or the negation of atoms or the negation aggregate atoms. This means that if dissatisfaction of \( \text{body}(r) \) is due to unsatisfied atom or aggregate atom or unsatisfied negated atom or negated aggregate atom the consequence is the same, which is the exclusion of \( r \) from the reduct of \( \Pi \).

Definition 13 An interpretation, \( I \), for a ground NDLP program, \( \Pi \), is an answer set for \( \Pi \) if \( I \) is \( \subseteq \)-minimal model for \( \Pi' \).

Observe that the definitions of the reduct and the answer sets semantics for NDLP programs are generalizations of the definitions of the reduct and the answer sets semantics for a single aggregate over a variable disjunctive logic programs, DLP-\( \delta \), described in [Faber et al., 2010], and hence, generalizations of the definitions of the reduct and the answer sets semantics for the original disjunctive logic programs, DLP, presented in [Gelfond and Lifschitz, 1991].

2.4 Semantics Properties

In this section we study the semantics properties of NDLP and its relationship to the answer set semantics of
a single variable and a single aggregate disjunctive logic programs, DLP\(^4\) \cite{Faberetal2010}, and its relationship to the answer set semantics of the original disjunctive logic programs, DLP \cite{GelfondLifsitz1991}. Theorem 1 Let \(\Pi\) be an NDLP program. The answer sets of \(\Pi\) are \(\preceq\)-minimal models for \(\Pi\).

The following theorem shows that the answer set semantics of NDLP subsumes the answer set semantics of DLP\(^4\) \cite{Faberetal2010}, and consequently subsumes the original answer set semantics of the original disjunctive logic programs DLP \cite{GelfondLifsitz1991}. DLP\(^4\) programs are NDLP programs with a single aggregation over a single variable which, unlike NDLP programs, do not allow aggregations over function terms. DLP programs are NDLP programs without any aggregate atoms.

Theorem 2 Let \(\Pi\) be a DLP\(^4\) program and \(I\) be an interpretation. Then, \(I\) is an answer set for \(\Pi\) iff \(I\) is an answer set for \(\Pi\) according to the answer set semantics of \cite{Faberetal2010}.

Proposition 1 Let \(\Pi\) be a DLP program and \(I\) be an interpretation. Then, \(I\) is an answer set for \(\Pi\) iff \(I\) is an answer set for \(\Pi\) according to the answer set semantics of \cite{GelfondLifsitz1991}.

3 Nested Aggregates Answer Set Optimization

In this section, we introduce the syntax and semantics of the answer set optimization programs with preferences that involve nested multiple aggregates, called nested aggregates preferences, and denoted by NASO programs, that modify and generalize the syntax and semantics of the answer set optimization with aggregate preferences presented in \cite{SaadBrewka2011}, from a single aggregate over a single variable preferences to nested multiple aggregates over multiple variables and nested multiple aggregates over functions that involve multiple variables preferences. An NASO program is a logic program under the answer set semantics whose answer sets are ranked according to preference relations represented in the program.

An NASO program, \(\Pi\), is a union of two sets of logic rules \(\Pi = R_{\text{gen}} \cup R_{\text{pref}}\). The first set of logic rules, \(R_{\text{gen}}\), is called the generator rules that generate the answer sets that satisfy every rule in \(R_{\text{gen}}\). \(R_{\text{gen}}\) is any set of logic rules with well-defined answer set semantics including normal, extended, and disjunctive sets of rules \cite{GelfondLifsitz1988, GelfondLifsitz1991, Faberetal2010}, as well as disjunctive logic with nested multiple aggregates sets of rules presented in the first part of this paper. The second set of logic rules, \(R_{\text{pref}}\), is a set of logic rules that represent the user preferences over the answer sets generated by \(R_{\text{gen}}\), called the preference rules. The preference rules in \(R_{\text{pref}}\) are used to rank the generated answer sets from the most preferred answer set to the least preferred one. An advantage of NASO is that \(R_{\text{gen}}\) and \(R_{\text{pref}}\) are independent. This makes preference elicitation easier and the whole approach is more intuitive and easy to use in practice. We focus on the syntax and semantics of the preference rules \(R_{\text{pref}}\) of the NASO programs \(\Pi = R_{\text{gen}} \cup R_{\text{pref}}\), since the syntax and semantics of \(R_{\text{gen}}\) is the same as syntax and semantics of any set of logic rules with answer set semantics \cite{GelfondLifsitz1991, Faberetal2010}.

3.1 NASO Programs Syntax

The language of NASO programs is the same as the language NDLP programs, presented in the first part of this paper, except that in the language of NASO programs classical negation is allowed. Let \(L\) be a first-order language with finitely many predicate symbols, function symbols, constants, and infinitely many variables. A term is a constant, a variable or a function. A literal is either an atom \(a\) in \(B_L\) or the negation of \(\neg a\), where \(B_L\) is the Herbrand base of \(L\) and \(\neg\) is the classical negation. The Herbrand universe of \(L\) is denoted by \(U_L\). Non-monotonic negation or the negation as failure is denoted by \(\text{not}\). Let \(Lit\) be the set of all literals in \(L\), where \(Lit = \{a|a \in B_L\} \cup \{-\neg a | a \in B_L\}\).

An expression of the form \(\{F | C\}\) is called a symbolic set, where \(F\) is a single variable or a function \(F = g(X_1, \ldots, X_n)\), where \(X_1, \ldots, X_n\) are variables or function terms and \(C\) is a conjunction of literals, non-monotonic negation of literals, aggregate atoms, and non-monotonic negation of aggregate atoms (defined below). However, a ground set is a set of pairs of the form \(\{F_G | C_G\}\) such that \(F_G\) is a constant and \(C_G\) is a conjunction of ground literals, non-monotonic negation of ground literals, ground aggregate atoms, and non-monotonic negotiation of ground aggregate atoms (defined below). A ground set or a symbolic set is called a set term. We say \(f(S)\) is an aggregate function if \(f\) is an aggregate function symbol and \(S\) is a set term, where \(f \in \{\min, \max, \text{count, sum, times}\}\). If \(f(S)\) is an aggregate function and \(T\) is a constant, variable or function term called guard, then we say \(f(S) \prec T\) is an aggregate atom, where \(\prec \in \{\text{=} \), \(\text{\neq, <, >, \leq, \geq}\}\). An optimization aggregate is an expression of the form \(\text{max}(f(S))\) or \(\text{min}(f(S))\), where \(S\) is a set term and \(f\) is an aggregate function symbol. Let \(A\) be a set of literals, aggregate atoms, and optimization aggregates. A boolean combination over \(A\) is a boolean formula over literals, aggregate atoms, and optimization aggregates in \(A\) constructed by conjunction, disjunction, and non-monotonic negation (\(\text{not}\)), where non-monotonic negation is combined only with literals and aggregate atoms.

Similar to NDLP, an aggregate function \(f(S)\) in an aggregate atom \(f(S) \prec T\) can be defined in terms of other aggregate atoms \(f'(S') \prec T'\) and hence in terms of other aggregate functions \(f'(S')\). This is because the conjunction in the definition of the symbolic set \(S\) of the aggregate function \(f(S)\) in the aggregate atom \(f(S) \prec T\) can contain other aggregate atoms as constituents. This implies that multiple level of nested aggregate functions
calculation is achieved by allowing aggregate functions of the aggregate atoms to be defined in terms of other aggregate atoms.

Let \( f(S) \) be an aggregate function. A variable, \( X \), is a local variable to \( f(S) \) if and only if \( X \) appears in \( S \) and \( X \) does not appear in any aggregate function that is outer to \( f(S) \) or in the preference rule the contains \( f(S) \). A global variable is a variable that is not a local variable.

**Definition 14** A preference rule, \( r \), over a set of literals, aggregate atoms, and optimization aggregates, \( A \), is an expression of the form

\[
C_1 > C_2 > \ldots > C_k \leftarrow l_{k+1}, \ldots, l_m, \text{not } l_{m+1}, \ldots, \text{not } l_n
\]

where \( l_{k+1}, \ldots, l_n \) are literals or aggregate atoms and \( C_1, C_2, \ldots, C_k \) are boolean combinations over \( A \).

Let \( r \) be a preference rule, body\( (r) = l_{k+1}, \ldots, l_m, \text{not } l_{m+1}, \ldots, \text{not } l_n, \) and head\( (r) = C_1 > C_2 > \ldots > C_k \). Intuitively, a preference rule, \( r \), says that any answer set that satisfies body\( (r) \) and \( C_1 \) is preferred over answer sets that satisfy body\( (r) \), some \( C_i \), (2 ≤ \( i \) ≤ \( k \)), not \( C_1 \), and any answer set that satisfies body\( (r) \) and \( C_2 \) is preferred over answer sets that satisfy body, some \( C_i \), (3 ≤ \( i \) ≤ \( k \)), but neither \( C_1 \) nor \( C_2 \), etc.

Recalling, an NASO program is a union of two sets of logic rules \( \Pi = R_{\text{gen}} \cup R_{\text{pref}} \), where \( R_{\text{gen}} \) is a set of logic rules with answer set semantics, called the generator rules, and \( R_{\text{pref}} \) is a set of preference rules.

### 3.2 NASO Programs Semantics

In defining the semantics of aggregate functions and aggregate atoms for NASO programs as they syntactically defined in the previous section, we follow the same semantics of aggregate functions and aggregate atoms as they defined for NDLP programs. We use for NASO programs the same notions of local ground instantiation and ground instantiation of symbollic sets as they defined for NDLP programs. Similarly, the semantics of the aggregate functions min, max, count, sum, and times are defined by the same mappings as defined for NDLP programs. Let \( \perp \) be a symbol that does not occur in any NASO program.

A ground instantiation of a preference rule, \( r \), is the replacement of each global variable appearing in \( r \) to a constant from \( U_r \), then followed by the ground instantiation of every symbolic set, \( S \), appearing in \( r \). The ground instantiation of an NASO program, \( \Pi = R_{\text{gen}} \cup R_{\text{pref}} \), is the set of all possible ground instantiations of every rule, \( r \), in \( \Pi \). Let \( I \) be an answer set for \( R_{\text{gen}} \) in an NASO program, \( \Pi = R_{\text{gen}} \cup R_{\text{pref}} \), \( f(S) \prec T \) be a ground aggregate atom, and \( S_t = \{ F_C \mid (F_G \mid C_G) \in S \text{ and } C_G \text{ is true in } f \} \) be the multiset constructed from elements in \( S \). Then, the evaluation of \( f(S) \) with respect to the answer set \( I \) is \( f(S_t) \), where \( f(S_t) = \perp \) if \( S_t \) is not in the domain of \( f \).

**Definition 15** Let \( \Pi = R_{\text{gen}} \cup R_{\text{pref}} \) be a ground NASO program, \( I \) be an answer set for \( R_{\text{gen}} \), and \( r \) be a preference rule in \( R_{\text{pref}} \). Then the satisfaction of a boolean combination, \( C_r \) appearing in head\( (r) \), by \( I \), denoted by \( I \models C_r \), is defined inductively as follows:

- \( I \models l \) iff \( l \in I \).
- \( I \models \neg l \) iff \( l \notin I \).
- \( I \models f(S) \prec T \) iff \( f(S_t) \neq \perp \) and \( f(S_t) \prec T \).
- \( I \models \neg f(S) \prec T \) iff \( f(S_t) = \perp \) or \( f(S_t) \neq \perp \) and \( f(S_t) \prec T \).
- \( I \models \max(f(S)) \) iff \( f(S_t) \neq \perp \), and for any answer set \( I' \), \( f(S_{t'}) \neq \perp \) and \( f(S_{t'}) \leq f(S_t) \) or \( f(S_t) \neq \perp \) and \( f(S_t) = \perp \).
- \( I \models \min(f(S)) \) iff \( f(S_t) \neq \perp \), and for any answer set \( I' \), \( f(S_{t'}) \neq \perp \) and \( f(S_{t'}) \leq f(S_t) \) or \( f(S_t) \neq \perp \) and \( f(S_t) = \perp \).
- \( I \models C_1 \wedge C_2 \) iff \( I \models C_1 \) and \( I \models C_2 \).
- \( I \models C_1 \vee C_2 \) iff \( I \models C_1 \) or \( I \models C_2 \).

The application of any aggregate function, \( f \), except count, on a singleton \( \{x\} \), returns \( x \), i.e., \( f(\{x\}) = x \). Therefore, we use \( \max(S) \) and \( \min(S) \) as abbreviations for the optimization aggregates \( \max(f(S)) \) and \( \min(f(S)) \) respectively, where \( S \) is a singleton and \( f \) is any arbitrary aggregate function except count.

**Definition 16** Let \( \Pi = R_{\text{gen}} \cup R_{\text{pref}} \) be a ground NASO program, \( I \) be an answer set for \( R_{\text{gen}} \), and \( r \) be a preference rule in \( R_{\text{pref}} \). Then the satisfaction of the body of \( r \) by \( I \), denoted by \( I \models \text{body}(r) \), is defined inductively as follows:

- \( I \models l \) iff \( l \in I \).
- \( I \models \neg l \) iff \( l \notin I \).
- \( I \models f(S) \prec T \) iff \( f(S_t) \neq \perp \) and \( f(S_t) \prec T \).
- \( I \models \neg f(S) \prec T \) iff \( f(S_t) = \perp \) or \( f(S_t) \neq \perp \) and \( f(S_t) \prec T \).
- \( I \models \text{body}(r) \) iff \( \forall(k + 1 \leq i \leq m), I \models l_i \), and \( \forall(m + 1 \leq j \leq n), I \models \neg l_j \).

The following definition specifies the satisfaction of the preference rules.

**Definition 17** Let \( \Pi = R_{\text{gen}} \cup R_{\text{pref}} \) be a ground NASO program, \( I \) be an answer set for \( R_{\text{gen}} \), \( r \) be a preference rule in \( R_{\text{pref}} \), and \( C_r \) be in head\( (r) \). Then, we define the following notions of satisfaction of \( r \) by \( I \):

- \( I \models r \) iff \( I \models \text{body}(r) \) and \( I \models C_r \).
- \( I \models \text{irr} r \) iff \( I \models \text{body}(r) \) and \( I \) does not satisfy any \( C_r \) in head\( (r) \).
- \( I \models \text{irr} r \) iff \( I \) does not satisfy body\( (r) \).

**Definition 18** Let \( \Pi = R_{\text{gen}} \cup R_{\text{pref}} \) be a ground NASO program, \( I_1, I_2 \) be two answer sets of \( R_{\text{gen}} \), \( r \) be a preference rule in \( R_{\text{pref}} \), and \( C_r \) be boolean combination appearing in head\( (r) \). Then, \( I_1 \) is strictly preferred over \( I_2 \) w.r.t. \( r \), denoted by \( I_1 \succ I_2 \), iff one of the following holds:
Definition 19 (Pareto Preference) Let $\Pi = R_{\text{gen}} \cup R_{\text{pref}}$ be an NASO program and $I_1, I_2$ be answer sets of $R_{\text{gen}}$. Then, $I_1$ is (Pareto) preferred over $I_2$ w.r.t. $R_{\text{pref}}$, denoted by $I_1 \succ_{R_{\text{pref}}} I_2$, if for every other rule $r' \in R_{\text{pref}}$, $I_1 \succ_{r'} I_2$. We say, $I_1$ and $I_2$ are equally (Pareto) preferred w.r.t. $R_{\text{pref}}$, denoted by $I_1 \equiv_{R_{\text{pref}}} I_2$, iff for all $r \in R_{\text{pref}}$, $I_1 \equiv_{r} I_2$.

Definition 20 (Maximal Preference) Let $\Pi = R_{\text{gen}} \cup R_{\text{pref}}$ be an NASO program and $I_1, I_2$ be answer sets of $R_{\text{gen}}$. Then, $I_1$ is (Maximal) preferred over $I_2$ w.r.t. $R_{\text{pref}}$, denoted by $I_1 \succ_{R_{\text{pref}}} I_2$, iff

$$\{r \in R_{\text{pref}} | I_1 \succ_{r} I_2\} > \{r \in R_{\text{pref}} | I_2 \succ_{r} I_1\}.$$

We say, $I_1$ and $I_2$ are equally (Maximal) preferred w.r.t. $R_{\text{pref}}$, denoted by $I_1 \equiv_{R_{\text{pref}}} I_2$, iff

$$\{r \in R_{\text{pref}} | I_1 \succ_{r} I_2\} = \{r \in R_{\text{pref}} | I_2 \succ_{r} I_1\}.$$

Observe that the Maximal preference relation is more general than the Pareto preference relation, since the Maximal preference definition subsumes the Pareto preference relation. Under the Pareto preference relation, the following result shows that the syntax and semantics of NASO programs subsume the syntax and semantics of the answer set optimization programs of [Brewka et al., 2003], since there is no notion of Maximal preference relation was introduced in [Brewka et al., 2003]. This is assuming that the answer set optimization programs of [Brewka et al., 2003] assign the lowest rank to the answer sets that do not satisfy neither the body nor the head of preference rules.

Theorem 3 Let $\Pi = R_{\text{gen}} \cup R_{\text{pref}}$ be an NASO program without either aggregate atoms or optimization aggregates and $I_1, I_2$ be answer sets of $R_{\text{gen}}$. Then, $I_1$ is Pareto preferred over $I_2$ w.r.t. $R_{\text{pref}}$ iff $I_1$ is Pareto preferred over $I_2$ w.r.t. $R_{\text{pref}}$ according to [Brewka et al., 2003].

However, the following result shows that the syntax and semantics of NASO programs subsume the syntax and semantics of a single aggregate over a single variable answer set optimization programs of [Saad and Brewka, 2011], under both the Pareto and the Maximal preference relations, since the notion of Maximal preference relation introduced in [Saad and Brewka, 2011] is a special case of the Maximal preference relation presented in this paper.

Theorem 4 Let $\Pi = R_{\text{gen}} \cup R_{\text{pref}}$ be a single aggregate over a single variable NASO program and $I_1, I_2$ be answer sets of $R_{\text{gen}}$. Then, $I_1$ is Pareto (Maximal) preferred over $I_2$ w.r.t. $R_{\text{pref}}$ iff $I_1$ is Pareto (Maximal) preferred over $I_2$ w.r.t. $R_{\text{pref}}$ according to [Saad and Brewka, 2011].

4 Probabilistic Traveling Salesman Problem

In this section we show that any instance of a priori optimization probabilistic traveling salesman problem (PTSP) [Jaillet, 1988] can be intuitively and easily represented and solved by the framework of nested multiple aggregates answer set optimization programs. In addition, we show that with a minor modification to our representation of any instance of PTSP problem in NASO, we can intuitively solve a corresponding classical traveling salesman problem (TSP) in our framework in particular, and in answer set programming in general, since finding the optimal tour for TSPs has not been considered before in answer set programming literature. This is because the emphasis in solving TSPs in answer set programming literature was on generating the possible tours of a given TSP rather than finding the optimal tour for that TSP, which is the tour with the minimum length. The reason for that is that a TSP are represented in answer set programming as answer sets and answer set programming is incapable of reasoning across answer sets to find the answer set that represents the tour with the minimum length. Therefore, a different framework that is capable of reasoning across answer sets is required, this framework is NASO programs.

Observe that our representation of PTSPs and consequently of TSPs is built on top of the existing answer set programming representation of TSPs, whose aim is to find all possible tours. This makes our NASO program representation further intuitive and keeps inline with the existing body of work in answer set programming. In addition, it shows and gives insight that many optimization problems can be solved by NASO framework in the same way by similar intuitive modifications to the existing answer set programming representation of corresponding similar problems.

4.1 Probabilistic Traveling Salesman Problem in NASO

Recalling, a Probabilistic Traveling Salesman Problem (PTSP), $G$, is a tuple of the form $G = (V, E, D, P)$, where $V$ is a set of $n$ vertices, $E$ is a set of edges, $D$ is an $n \times n$ distance matrix, where $d_{ij}$ is distance from
a vertex \( i \) to a vertex \( j \), and \( P = \{p_1, p_2, \ldots, p_n\} \) is a probability distribution over vertices in \( V \), where \( p_i \) is the probability a vertex \( i \) must be visited. The aim is to find a priori a tour, \( t = (0, 1, 2, 3, 4, \ldots, n, 0) \), of \( G \), with the minimum expected length, where 0 is the starting point from which the tour always starts and ends and whose probability \( p_0 = 1 \). The expected length, \( E[L_t] \), of a tour \( t \) is given by formula (I).

Any Probabilistic Traveling Salesman Problem (PTSP), \( G = (V,E,D,P) \), is represented as an NASO program \( \Pi = R_{gen} \cup R_{pref} \), where the generator rules in \( R_{gen} \) generate the answer sets the represents all the possible tours in \( G \) and the preference rules in \( R_{pref} \) rank the answer sets that correspond to all the possible tours in \( G \) from the tour with the minimum expected length to the tour with the maximum expected length. The representation proceeds as follows.

- Every vertex \( x \) in \( V \) whose probability is \( p \) in \( P \) is represented in \( R_{gen} \) as a fact of the form
  \[
  \text{vertex}(x,p) \leftarrow \tag{6}
  \]
  In addition to the starting vertex \( y \in V \) is represented in \( R_{gen} \) as a fact of the form
  \[
  \text{start}(y) \leftarrow \tag{7}
  \]

- Every edge \((x,y)\) in \( E \) is represented in \( R_{gen} \) as a fact of the form
  \[
  \text{edge}(x,y) \leftarrow \tag{8}
  \]

- Every element \( d_{xy} \) in the distance matrix \( D \) is represented in \( R_{gen} \) as a fact of the form
  \[
  \text{distance}(x,y,d_{xy}) \leftarrow \tag{9}
  \]

- The logic rules that generate all the possible tours from \( G \), are represented in \( R_{gen} \) by the disjunctive logic rules:
  \[
  \text{rinTour}(X,Y) \vee \text{outTour}(X,Y) \leftarrow \text{start}(X), \text{edge}(X,Y). \tag{10}
  \]

- Every distance \((x,y)\) is a distance \( d_{xy} \), where \( d_{xy}(x,y) \) is a distance \( d \), which represents the distance from a vertex \( i \) to a vertex \( j \).
  \[
  \text{inTour}(X,Y) \vee \text{outTour}(X,Y) \leftarrow \text{reached}(X), \text{edge}(X,Y). \tag{11}
  \]

- The indexing of vertices in any given tour, \( t \), by their order of appearance in the tour \( t \) is represented in \( R_{gen} \) by the logic rules:
  \[
  \text{index}(0,X) \leftarrow \text{start}(X). \tag{17}
  \]

- The computation of the expected length of each tour and the ranking of the answer sets corresponding to the tours in \( G \) from the minimum expected length tour to the maximum expected length tour is represented in \( R_{pref} \) by the preference rule (denoted by the preference rule opt):
  \[
  \min(\sum A | \text{index}(I,X), \text{vertex}(X,P_X), 0 \leq I \leq n, \sum P_X + P_Y + D \times M | \text{index}(J,Y), \text{vertex}(Y,P_Y), \text{distance}(X,Y,D), I + 1 \leq J \leq n + 1, \text{times} \{ (1 - P_Z) | \text{index}(K,Z), \text{vertex}(Z,P_Z), I + 1 \leq K \leq J - 1 \} = M \} = A \).
  \]

- Observe that the logic rules (10)–(12) are exactly a typical set of logic rules that are used in answer set programming literature [Eiter et al., 2000] to find Hamiltonian cycles of a given graph adapted to deal with graphs with probabilistic vertices. It can be easily seen that the nested aggregate function within the optimization aggregate \( \min \) in the above preference rule is exactly the representation of the expected length of a tour described by formula (I).

**Theorem 5** Let \( G = (V,E,D,P) \) be a Probabilistic Traveling Salesman Problem and \( \Pi = R_{gen} \cup R_{pref} \) be the NASO program representation of \( G \). Then, a tour, \( t \), is the optimal tour for \( G \) with the minimum expected length iff an answer set, \( I \), of \( R_{gen} \) is the top preferred answer set w.r.t. \( R_{pref} \).

### 4.2 Traveling Salesman Problem in NASO

In this section, we show that with a minor modification to the NASO program representation of a probabilistic traveling salesman problem, we can intuitively solve a corresponding classical traveling salesman problem (TSP). A classical Traveling Salesman Problem (TSP), \( G \), is a tuple of the form \( G = (V,E,D) \), where \( V \) is a set of \( n \) vertices, \( E \) is a set of edges, and \( D \) is an \( n \times n \) distance matrix, where \( d(i,j) \) is an element in \( D \) that represents the distance from a vertex \( i \) to a vertex \( j \). The aim is to find a tour, \( t = (0,1,2,3,4,\ldots,n,0) \), of \( G \), with the minimum length, where 0 is the starting point from which the tour always starts and ends and the length, \( L_t \), of a tour \( t \) is calculated by

\[
L_t = \sum_{i=0}^{n} d(i,i+1) \tag{19}
\]

where \( n+1 = 0 \). A Traveling Salesman Problem (TSP), \( G = (V,E,D) \), is represented as an NASO program, \( \Pi = R_{gen} \cup R_{pref} \), where the generator rules in \( R_{gen} \) generate the answer sets the represents all the possible tours in \( G \) and the preference rules in \( R_{pref} \) rank the answer sets that correspond to all the possible tours in \( G \) from the tour with the minimum length to the tour with the maximum length. The generator rules in \( R_{gen} \) consists of the logic rules from the logic rule (10) through the logic rule (13) after replacing \( \text{vertex}(x,p) \) in the logic rule (10)
by vertex($x$) and replacing vertex($X, P$) in the logic rule 15 by vertex($X$). The computation of the length of each tour and the ranking of the answer sets corresponding to the tours in $G$ from the minimum length tour to the maximum length tour is represented in $R_{pref}$ by the preference rule:

$$\min(\sum \{ D \mid \text{distance}(X, Y, D), \text{index}(I, X), \text{index}(I + 1, Y), 0 \leq I \leq n \}) \leftarrow$$

**Theorem 6** Let $G = \langle V, E, D \rangle$ be a classical Traveling Salesman Problem and $\Pi = R_{gen} \cup R_{pref}$ be the NASO program representation of $G$. Then, a tour, $t$, is the optimal tour for $G$ with the minimum length if an answer set, $I$, of $R_{gen}$ is the top preferred answer set w.r.t. $R_{pref}$.

4.3 Probabilistic Traveling Salesman Problem in NDLP

We also show that a Probabilistic Traveling Salesman Problem, $G = \langle V, E, D, P \rangle$, can be represented as an NDLP program, $\Pi$, whose answer sets correspond to tours in $G$. Although, NDLP programs framework is still capable of finding all possible tours in $G$, represented as answer sets, and computing the expected length of each tour, the semantics of NDLP programs is not able to locate the answer set that corresponds to the optimal tour of $G$, since it does not have the capability of reasoning across the answer sets. However, PTSPs can be represented and solved in NDLP programs framework in two steps. The first step is to represent a probabilistic traveling salesman problem, $G$, by an NDLP program whose answer sets correspond to all possible tours in $G$ along with their expected length. The second step is to determine the minimum expected length optimal tour, $t$, in $G$ represented by an answer set of the NDLP program representation of $G$ by means of any appropriate procedure internal or external to NDLP programs framework.

The NDLP program, $\Pi$, representation of a Probabilistic Travelling Salesman Problem, $G$, consists of the NDLP rules from the logic rule 15 through the logic rule 18, in addition to the following NDLP rule, where $X$ in length($X$) in the NDLP rule below represents the value of the expected length of a tour.

$$\text{length}(X) \leftarrow \sum \{ A \mid \text{index}(I, X), \text{vertex}(X, P_X), 0 \leq I \leq n, \sum \{ P_X * P_Y * D * M \mid \text{index}(J, Y), \text{vertex}(Y, P_Y), \text{distance}(X, Y, D), I + 1 \leq J \leq n + 1, \text{times} \{ (1 - P_Z) \mid \text{index}(K, Z), \text{vertex}(Z, P_Z), I + 1 \leq K \leq J - 1 \} = M \} = X.$$

**Theorem 7** Let $G = \langle V, E, D, P \rangle$ be a Probabilistic Traveling Salesman Problem and $\Pi$ be the NDLP program representation of $G$. Then, $t$ is a tour for $G$ iff $I$ is an answer set for $\Pi$ corresponds to $t$, where the expected length, $E[I_t]$, of $t$ is equal to the value of $X$ in length($X$) that is satisfied by $I$.

One possible way to find the optimal tour with the minimum expected length is to add the following two NDLP rules to the NDLP program representation, $\Pi$, of a probabilistic traveling salesman problem, $G$, where $X$ in min($X$) in the NDLP rules below represents the minimum expected length value of a tour

$$\min(X) \leftarrow \text{length}(X), \text{not maximal}(X). \ (20)$$

$$\text{maximal}(X) \leftarrow \text{length}(X), \text{length}(X), X_1 < X. \ (21)$$

**Theorem 8** Let $G = \langle V, E, D, P \rangle$ be a Probabilistic Traveling Salesman Problem and $\Pi$ be the NDLP program representation of $G$. Then, a tour, $t$, is the optimal tour for $G$ with the minimum expected length iff an answer set $I$ of $\Pi$ is the only answer set of $\Pi$ that satisfies min($X$).

However, the addition of the NDLP rules (20) and (21) to the NDLP program representation of a probabilistic traveling salesman problem leads when grounded to large number of ground rules, especially with the graphs that involve sufficiently large number of vertices and large number of edges, which is always the case with PTSPs. Therefore, a simple linear search for the minimum value of $X$ in length($X$) over all the answer sets of the NDLP program representation of a probabilistic traveling salesman problem is likely to be more efficient than performing the same task using logic rules.

4.4 Working Example

This section shows that a probabilistic traveling salesman problem instance can be intuitively and easily represented and solved by the NASO programs framework. Consider this instance of the probabilistic traveling salesman problem instance, $G = \langle V, E, D, P \rangle$, where $\langle V, E \rangle$ is complete undirected graph where $V = \{ a, b, c, d \}$ and $E = \{ (a, b), (a, c), (a, d), (b, c), (b, d), (c, d), (b, a), (c, a), (d, a), (c, b), (d, b), (d, c) \}$, the probability distribution, $P$, is given as $p_a = 1$, $p_b = 0.3$, $p_c = 0.7$, and $p_d = 0.4$, and the distance matrix, $D$, is given by $d_{ab} = 40$, $d_{ac} = 40$, $d_{ad} = 22$, $d_{bc} = 40$, $d_{bd} = 25$, $d_{cd} = 22$, $d_{da} = 40$, $d_{db} = 40$, $d_{dc} = 22$. Let $\Pi = R_{gen} \cup R_{pref}$ be the NASO program representation of this probabilistic traveling salesman problem instance, $G$, where $R_{pref}$ consists of the preference rule opt, with $n = 3$, and $R_{gen}$ consists of the logic rules (10) through (18), in addition to the facts:

vertex($a, 1$). vertex($b, 0.3$). vertex($c, 0.7$). vertex($d, 0.4$).
edge($a, b$). edge($a, c$). edge($a, d$). edge($b, d$).
edge($c, d$). edge($b, a$). edge($c, a$). edge($d, a$).
edge($d, b$). edge($d, c$). distance($a, c, 40$). distance($a, b, 40$).
distance($a, d, 22$). distance($b, c, 40$). distance($b, d, 25$).
distance($c, d, 22$). distance($c, a, 40$). distance($b, a, 40$).
distance($c, b, 40$). distance($d, b, 25$). distance($d, a, 22$).
distance($d, c, 22$). start($a$).

By considering only the relevant atoms, $R_{gen}$, has six answer sets which correspond to the six available tours in $G$. These answer sets are:

$I_1 = \{ \text{start}(a), \text{inTour}(a, d), \text{inTour}(d, b), \text{inTour}(b, c), \text{inTour}(c, a), \text{index}(0, a), \text{index}(1, d), \text{index}(2, b), \text{index}(3, c), \text{index}(4, a) \}$
by considering only the relevant atoms, it can be easily expected length tour of min aggregates answer set programming to the showed the applicability of both the nested aggregates multiple variables and to nested multiple aggregates over answer set programming from a single aggregate over a metrics of both answer set optimization programming and answer set optimization programs with a single aggregate over a single variable. The existing answer set programming literature considered a single aggregate over a single variable answer set programming approaches. A comprehensive comparisons of the existing a single aggregate over a single variable answer set programming approaches is found in [Faber et al., 2010].

The major difference in the development presented in this paper is that we allow the ability to reasoning in the presence of nested multiple aggregates in answer sets in general and in both answer set optimization programming and in answer set programming in particular. The only answer set optimization framework that allows aggregates is [Saad and Brewka, 2011], which is answer set optimization programs with a single aggregate over a single variable. The existing answer set programming literature considered a single aggregate over a single variable answer set programming approaches. A comprehensive comparisons of the existing a single aggregate over a single variable answer set programming approaches is found in [Faber et al., 2010].

5 Conclusions and Related Work

We developed a generalization for the syntax and semantics of both answer set optimization programming and answer set programming from a single aggregate over a single variable to nested multiple aggregates over multiple variables and to nested multiple aggregates over functions involving multiple variables. In addition, we showed the applicability of the nested aggregates answer set optimization programming and the nested aggregates answer set programming to the a priori optimization Probabilistic Traveling Salesman Problem, a fundamental stochastic optimization problem in Operation Research. Moreover, we showed that with minor modification to the probabilistic traveling salesman problems representation in nested aggregate answer set optimization programming and answer set optimization programming, a corresponding classical traveling salesman problems can be intuitively represented and solved in both frameworks.

We showed that the nested aggregates answer set optimization programming framework presented in this paper modifies and generalizes a single aggregate over a single variable answer set optimization programming framework of [Saad and Brewka, 2011] as well as a generalization of the answer set optimization programming framework of [Brewka et al., 2003]. We showed that the nested aggregates answer set programming framework presented in this paper subsumes a single aggregate over a single variable answer set programming framework of [Faber et al., 2010] as well as a generalization of the original answer set programming framework of [Gelfond and Lifschitz, 1991].

There are six tours for, which are $t_1$, $t_2$, $t_3$, $t_4$, $t_5$, and $t_6$ that correspond to the six answer sets $I_1$, $I_2$, $I_3$, $I_4$, $I_5$, and $I_6$ respectively. The expected length, $E[L_{t_i}]$, of each tour, $t_i$, in $G$, calculated by formula (1), is given as:

$$E[L_{t_1}] = 75.16, \quad E[L_{t_2}] = 76.67, \quad E[L_{t_3}] = 76.92, \quad E[L_{t_4}] = 76.92, \quad E[L_{t_5}] = 76.92, \quad E[L_{t_6}] = 76.67$$

that exactly corresponds to the evaluation of the nested aggregate function within the optimization aggregate, $\text{min}$, of the preference rule $\text{opt}$ contained in the set of the preference rules $R_{\text{pref}}$ of the NASO program representation, $\Pi = R_{\text{gen}} \cup R_{\text{pref}}$, of the probabilistic traveling salesman problem instance $G$. It is clear that $t_1$ is the tour with minimum expected length, $E[L_{t_1}] = 75.16$, and hence the optimal tour for $G$, as well as, $I_1$ is the top answer set of $R_{\text{gen}}$ with respect to $R_{\text{pref}}$. This is because

$$I_1 \models \text{opt}, \quad I_2 \models \text{irr}, \quad I_3 \models \text{irr}, \quad I_4 \models \text{irr}, \quad I_5 \models \text{irr}, \quad I_6 \models \text{irr}$$

Similarly, this probabilistic traveling salesman problem instance can be represented by an NDLP program, $\Pi'$, that consists of the NDLP rules (10) through (21), as well as the NDLP rule representation of the expected length (the definition of the $\text{length}(X)$ predicate), and by considering only the relevant atoms, it can be easily seen that $I_1$ is the only answer set of $\Pi'$ that satisfies $\text{min}(X)$ and coincides with the optimal minimum expected length tour of $G$.

References

[Brewka et al., 2003] G. Brewka, I. Niemelä, and M. Truszczynski. Answer set optimization. In IJCAI, 2003.

[Eiter et al., 2000] T. Eiter, W. Faber, N. Leone, and G. Pfeifer. Declarative problem-solving using the dlv system. In Logic-Based AI, 2000.

[Faber et al., 2010] W. Faber, N. Leone, and G. Pfeifer. Semantics and complexity of recursive aggregates in answer set programming. Artificial Intelligence, 2010.

[Gelfond and Lifschitz, 1988] M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In IJCAI, 1988.

[Gelfond and Lifschitz, 1991] M. Gelfond and V. Lifschitz. Classical negation in logic programs and disjunctive databases. New Generation Computing, 9(3-4):363–385, 1991.

[Jaillet, 1988] P. Jaillet. A priori solution of a travelling salesman problem in which a random subset of the customers are visited. Operation Research, 36(6):929–936, 1988.
[Saad and Brewka, 2011] E. Saad and G. Brewka. Aggregates in answer set optimization. In 11th International Conference on Logic Programming and Non-monotonic Reasoning, 2011.