MILPHelper

PARTITION : bool
PRINT_LABELS : bool
PRINT_VARIABLES : bool
cardinalities : list[SigmaCount]
constraints : list[Inequation]
crisp_concepts : set[str]
crisp_roles : set[str]
nominal_variables : bool
number_of_variables : dict[str, int]
show_vars : ShowVariablesHelper
string_features : set[str]
string_values : dict[int, str]
variables : list[Variable]

__add_new_constraint_1(expr: Expression, constraint_type: InequalityType): None
__add_new_constraint_2(x: Variable, n: float): None
__add_new_constraint_3(ass: Assertion, n: float): None
__add_new_constraint_4(x: Variable, d: Degree): None
__add_new_constraint_5(ass: Assertion): None
__add_new_constraint_6(expr: Expression, constraint_type: InequalityType, degree: Degree): None
__add_new_constraint_7(expr: Expression, constraint_type: InequalityType, n: float): None
__bfs(graph: nx.Graph, solution: dict[int, int]): int
__common_partition_part(objective: Expression): tuple[list[Variable], dict[int, int], int, list[int], int, int]
__compute_partition(queue: list[int], solution: dict[int, int], p: int, graph: nx.Graph): None
__get_graph(): nx.Graph
__get_nominal_variable_1(i1: str): Variable
__get_nominal_variable_2(i1: str, i2: str): Variable
__get_ordered_permutation_1(x: list[Variable]): list[Variable]
__get_ordered_permutation_2(x: list[Variable], z: list[list[Variable]]): list[Variable]
__get_variable_1(var_name: str): Variable
__get_variable_10(a: str, b: str, role: str, v_type: VariableType): Variable
__get_variable_11(ind: CreatedIndividual): Variable
__get_variable_12(ind: CreatedIndividual, v_type: VariableType): None
__get_variable_2(var_name: str, v_type: VariableType): Variable
__get_variable_3(ass: Assertion): Variable
__get_variable_4(rel: Relation): Variable
__get_variable_5(ind: Individual, restrict: Restriction): Variable
__get_variable_6(ind: Individual, c: Concept): Variable
__get_variable_7(ind: Individual, concept_name: str): Variable
__get_variable_8(a: Individual, b: Individual, role: str): Variable
__get_variable_9(a: Individual, b: Individual, role: str, v_type: VariableType): Variable
__has_variable_1(name: str): bool
__has_variable_2(ass: Assertion): bool
__init__(): None
__print_instance_of_labels_1(f_name: str, ind_name: str, value: float): None
__print_instance_of_labels_2(name: str, value: float): None
__remove_nominal_variables(): None
__solve_gurobi_using_partitions(objective: Expression): typing.Optional[Solution]
add_cardinality_list(sc: SigmaCount): None
add_contradiction(): None
add_crisp_concept(concept_name: str): None
add_crisp_role(role_name: str): None
add_equality(var1: Variable, var2: Variable): None
add_new_constraint(expr: Expression, constraint_type: InequalityType): None
add_string_feature(role: str): None
add_string_value(value: str, int_value: int): None
change_variable_names(old_name: str, new_name: str, old_is_created_individual: bool): None
check_if_replacement_is_needed(v1: Variable, s1: str, v2: Variable, s2: str): bool
clone(): typing.Self
exists_nominal_variable(i: str): bool
exists_variable(a: Individual, b: Individual, role: str): bool
get_name_for_integer(i: int): typing.Optional[str]
get_negated_nominal_variable(i1: str, i2: str): Variable
get_new_variable(v_type: VariableType): Variable
get_nominal_variable(i1: str): Variable
get_number_for_assertion(ass: Assertion): int
get_ordered_permutation(x: list[Variable]): list[Variable]
get_variable(var_name: str): Variable
has_nominal_variable(terms: list[Term]): bool
has_variable(name: str): bool
is_crisp_concept(concept_name: str): bool
is_crisp_role(role_name: str): bool
is_nominal_variable(i: str): bool
optimize(objective: Expression): typing.Optional[Solution]
print_instance_of_labels(f_name: str, ind_name: str, value: float): None
set_binary_variables(): None
set_nominal_variables(value: bool): None
solve_gurobi(objective: Expression): typing.Optional[Solution]
solve_mip(objective: Expression): typing.Optional[Solution]
solve_pulp(objective: Expression): typing.Optional[Solution]