IMPL

MODULES

IMPL

UtilityFunctions
[AgentId, Action, Config <: Configuration[...]]
depends on a Configuration

VertexColoringUtility
computeUtility(c: Config): Double

ConflictBasedVertexColoringUtility
computeUtility(c: Config): Double

TargetFunctions
[AgentId, Action, Config <: Configuration[..], UtilityType]
depends on a Configuration

MemoryLessTargetFunction
computeExpectedUtilities(c: Config):
Map[Action, UtilityType]

RankWeightedTargetFunction
[... Config <: RankedConfig[...] ...]
computeExpectedUtilities(c: Config):
Map[Action, Double]

etc.

Configurations
all with types [Id, Action]

SimpleConfig
neighborhood: Map[Id, Action]
domain: Set[Action]
centralVariableAssignment: (Id, Action)
final withCentralVariableAssignment(value: Action): Config
computeExpectedNumberOfConflicts: Int

RankedConfig
neighborhood: Map[Id, Action]
ranks: Map[Id, Double]
domain: Set[Action]
centralVariableAssignment: (Id, Action)
final withCentralVariableAssignment(value: Action): Config
computeExpectedNumberOfConflicts: Int

DecisionRulesWithTargetFunctions
all with types [AgentId, Action, Config <: Configuration[..]]
depend on a Configuration

ArgmaxADecisionRule
computeMove(c: Config): Action

etc.

ConvergenceRules
all with types [AgentId, Action, Config <: Configuration[..]]
depend on a Configuration

NashEquilibriumConvergence
[AgentId, Action, Config <: Configuration[...]]
isConverged(c: Config): Boolean
isConvergedGivenUtilitiesAndMaxUtility(...): Boolean

ZeroConflictConvergence
[AgentId, Action, Config <: Configuration[...]]
isConverged(c: Config): Boolean

ZeroUtilityConvergence
[AgentId, Action, Config <: Configuration[...]]
isConverged(c: Config): Boolean
isConvergedGivenUtilitiesAndMaxUtility(...): Boolean

AdjustmentSchedules
all with types [AgentId, Action, Config <: Configuration[AgentId, Action]]
depend on a Configuration

ParallelRandomAdjustmentSchedule
shouldConsiderMove(c: Config): Boolean

FloodAdjustmentSchedule
shouldConsiderMove(c: Config): Boolean

RankedAdjustmentSchedules
all with types [AgentId, Action]
depend on RankedConfig

RankedBasedAdjustmentSchedule
shouldConsiderMove(c: Config): Boolean

etc.
shouldConsiderMove(c: Config): Boolean

TargetFunction
[AgentId, Action, Config, UtilityType]
computeExpectedUtilities(c: Config):
Map[Action, UtilityType]

DecisionRule
[AgentId, Action, Config <: Configuration[AgentId, Action], UtilityType]
computeMove(c: Config): Action
isConverged(c: Config): Boolean
isConvergedGivenUtilitiesAndMaxUtility(c: Config,
expectedUtilities: Map[Action, UtilityType],
maxUtility: UtilityType): Boolean

Configuration
[AgentId, Action]
neighborhood: Map[AgentId, Action]
domain: Set[Action],
centralVariableAssignment: (AgentId, Action)
withCentralVariableAssignment(value: Action): this.type
centralVariableValue
computeExpectedNumberOfConflicts: Int

UtilityFunction
[AgentId, Action, Config <:
Configuration[AgentId, Action], UtilityType]
computeUtility(c: Config): UtilityType

AdjustmentSchedule
[AgentId, Action, Config <: Configuration[AgentId, Action]]
shouldConsiderMove(c: Config): Boolean

Optimizer
[AgentId, Action, Config <: Configuration[AgentId, Action], UtilityType]
schedule: AdjustmentSchedule[AgentId, Action, Config]
rule: DecisionRule[AgentId, Action, Config, UtilityType]
with TargetFunction[AgentId, Action, Config, UtilityType]
with UtilityFunction[AgentId, Action, Config, UtilityType]
shouldConsiderMove(c: Config): Boolean = schedule.shouldConsiderMove(c)
computeMove(c: Config): Action = rule.computeMove(c)
isConverged(c: Config): Boolean = rule.isConverged(c)

...Optimizer
[AgentId, Action]
schedule: IMPL.AdjustmentSchedules
rule: Mix and Match from IMPL. :
DecisionRules, ConvergenceRules,
TargetFunctions, Utilities

trait
abstract class
class
case class
object

EXTENDS

USES

group sharing dependencies