Yolo Knowledgebase

An Overview

Outline

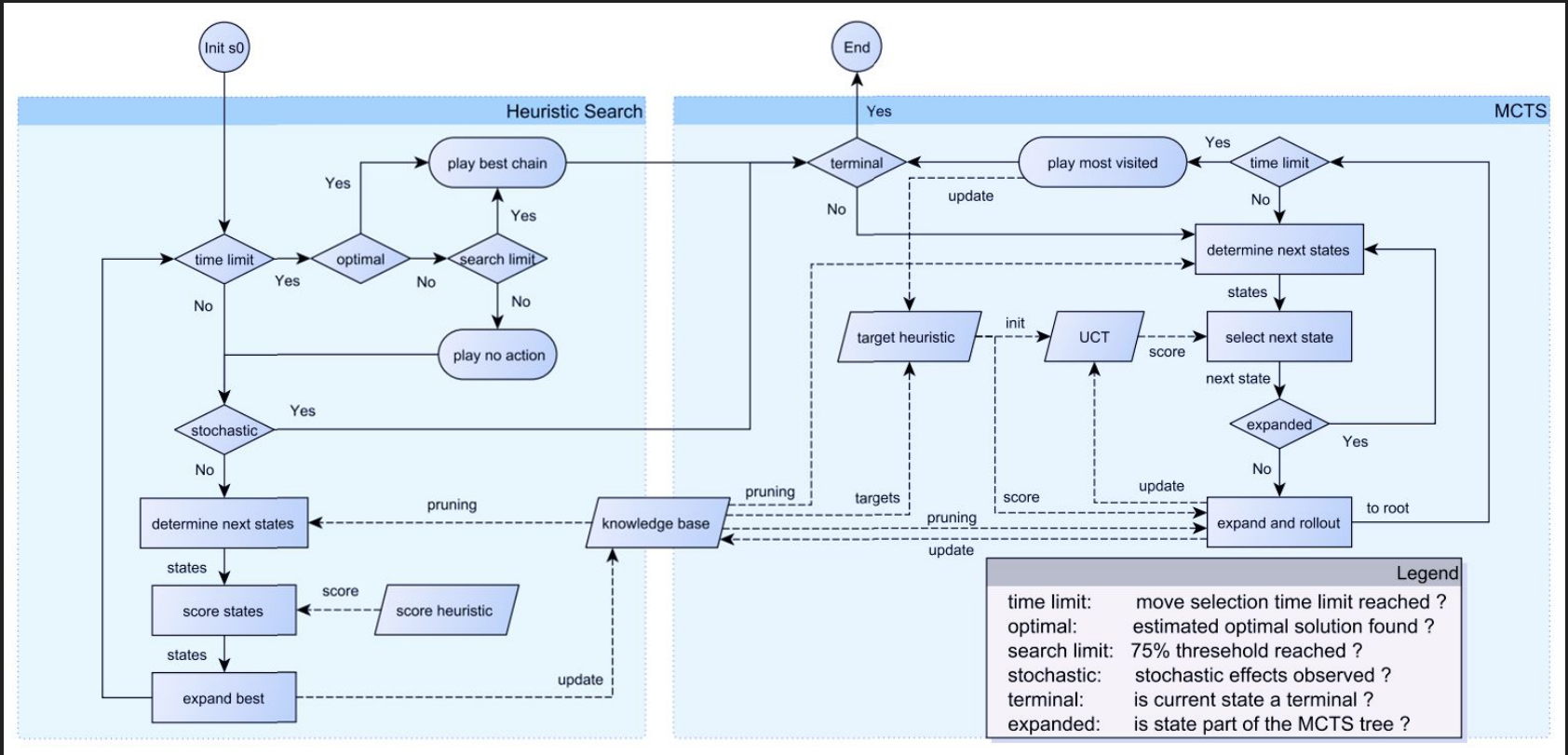
1. Core Concept
    a. Yolo Agent
    b. KnowledgeBase: Provided Tasks
2. Information provided by the Yolo Knowledgebase
    a. Avatar related functions
    b. NPC related functions
    c. Passive object related functions
    d. General game related functions
    e. Masks
3. Collision Classifiers
4. Current Issues
5. Planned additions/expansions

1. Core Concept

YoloAgent

- Start using an enhanced heuristic search
- Switches to MCTS if stochastic effects were observed or if a game-tick limit is reached
- Uses an evaluation heuristic for UCT selection and guided rollouts
- Utilizes the KnowledgeBase to prune the possible game states

Core Concept



1. Core Concept

KnowledgeBase: Provided tasks

- Provide an approximation of the transition function
- Predict object tasks
- Prediction of collision effects
- Prediction of game specific effects
- Determination of interesting game states
- Pruning the search space

2. Information provided by the Yolo Knowledgebase

Avatar related functions

boolean agentHasControlOfMovement(YoloState state)

boolean avatarLooksOutOfGame(YoloState state)

boolean moveWillCancel(YoloState currentState, ACTIONS action, boolean killIsCancel, boolean ignoreStochasticEnemyKilling)

update expected position, by checking if specified action causes no position move because of orientation change
check if avatar can be killed by enemy or bad spawners

boolean canUseInteractWithSomethingAt(YoloState state)

check if avatar can interact with something at next grid w.r.t the avatar current orientatition

boolean canInteractWithUse(int avatarItype, int objectItype)

get event in useEvents specified by parameters, return true if not getWall() and there is score increase

boolean getIncreaseScoreIfInteractWith(int avatarItype, int objectItype)

check if there is score increase if the two specified objects can interact with each other

boolean hasEverBeenAliveAtFieldWithItypeIndex(int avatarIndex, int passiveIndex)

boolean playerItypeIsWellKnown(YoloState state)

LinkedList<Integer> getPossiblePlayerItypes()

int getInventoryMax(int slot)

2. Information provided by the Yolo Knowledgebase

NPC related functions

int getNpcMaxMovementX(int itype)
int getNpcMaxMovementY(int itype)
boolean isStochasticEnemy(int index)
boolean canBeKilledByStochasticEnemyAt(YoloState state, int xPos, int yPos, boolean ignoreTicks)
Observation getPossibleStochasticKillerAt(YoloState state, int xPos, int yPos, boolean ignoreTicks)
boolean movesAtTick(int obsIndex, int gameTick)
boolean movesAtTickOrDirectFollowing(int obsIndex, int gameTick)
byte getNpcMovesEveryXTicks(int npcIndex)

2. Information provided by the Yolo Knowledgebase

Passive object related functions

int getObjectCategory(int objectIndex, YoloState state)
LinkedList<Integer> getPushableITypes()
int getPushTargetIndex(int pushObjectIndex)
int getPortalExitEntryIType(int portalExitIndex)
boolean isSpawner(int itype)
boolean isSpawnable(int itype)
int getSpawnIndexOfSpawner(int itype)
int getSpawnerIndexOfSpawned(int itype)

2. Information provided by the Yolo Knowledgebase

General game related functions

boolean positionAufSpielfeld(int x, int y)

Long getPropablyHash(YoloState currentState, ACTIONS action, boolean ignoreNPCs)

boolean actionsLeadsOutOfBattlefield(YoloState state, ACTIONS action)

boolean canIncreaseScoreWithoutWinning(YoloState state)

boolean haveEverGotScoreWithoutWinning()

boolean isMinusScoreBad()

2. Information provided by the Yolo Knowledgebase

Masks

int getBlockingMask(int index)

int getPlayerIndexMask()

int getDynamicMask()

int getFromAvatarMask()

3. Collision Classifiers

- Uses movement prediction subsystem to predict possible collisions
- Collisions are then classified into three classes:
  - Blocked Movement
  - Class A dependent on avatar/object pair
  - Class B dependent on avatar/object pair

  → Ignores additional effect groups (only the first two are used)

- Two Classifiers:
  - Blocked Movement Classifier: Blocked ←→ Effect Type
  - Effect Type Classifier: Class A ←→ Class B
- Classifier learn single conjunctive rule (e.g. I_1>5 && I_2 <10 → Blocked)

3. Collision Classifiers

Blocked Movement Classifier

- If a movement action does not change the position of the avatar:
  → the current inventory is used as positive example
  → all other inventories are negative examples
  → the negative examples are forwared to the effect type classifier

Effect Type Classifier

- First effects of non blocking collision are labeled as ”Class A”
- If effects of a later collision do not fit to effects of “Class A”
  → effects are labeled “Class B”

4. Current Issues

- Unimplemented methods can irritate the user
- Bad documentation is a problem for using the knowledgebase the right way
- Issues from game analysis: Continuous non grid based movement enemies
- Issues from game analysis: Spawner with treasures on it are not blocked
- Issues from game analysis: Use actions not correctly “tried” to find the target

4. Current Issues

Unimplemented methods: Do not use these methods:

- ***void lernActionResult(YoloState currentState, YoloState lastState)***
  - Better: ***learnFrom(YoloState currentState, YoloState lastState, ACTIONS* actionDone)**
  - Automatic learning when advancing: ***YoloState copyAdvanceLearn(ACTIONS action)***

- ***boolean canIndexMoveTo(int itypeIndex, int x, int y, Vector2d moveDirection)***
  - Try if your avatar will be blocked: ***boolean moveWillCancel(YoloState curre***
    ACTIONS action, boolean killIsCancel, boolean ignoreStochasticEnemyK
  - Try if your avatar can be killed by enemy: ***boolean***
    canBeKilledByStochasticEnemyAt(YoloState state, int xPos, int yPos)

4. Current Issues

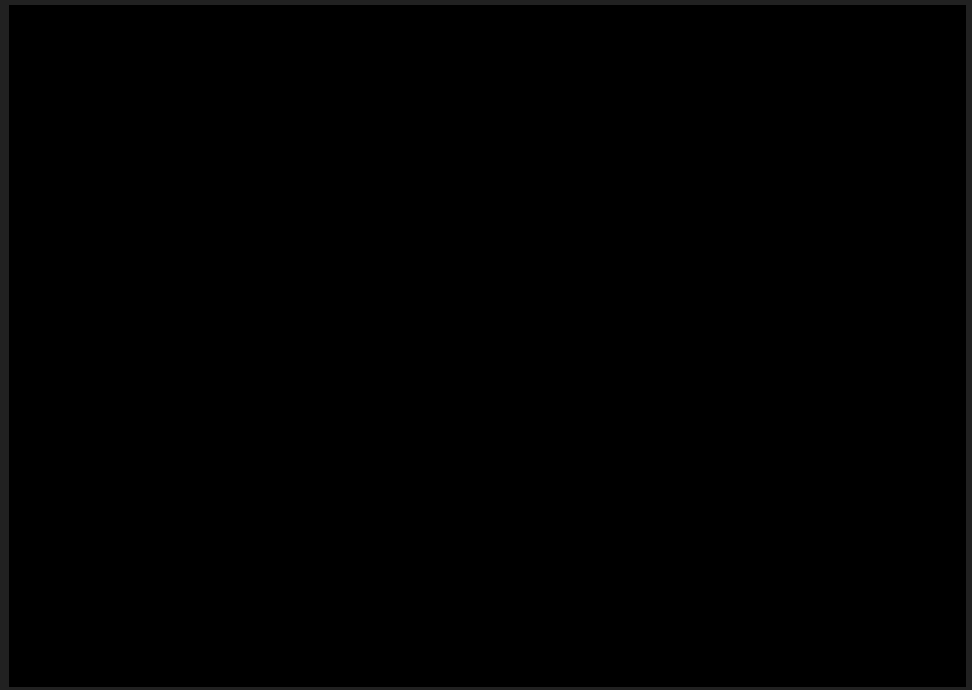
Problems with current Games: Continuous non grid based moving enemies

- Normally Yoloknowledge knows where enemies could move to, the YoloAgent does not move to such positions to avoid losing the game

- The Yoloknowledgebase fails to determine the movement of continuous non grid based moving enemies => Sometimes the YoloAgent loses the game

4. Current Issues

Problems with current Games: Continuous non grid based moving enemies, example Jaws



4. Current Issues

Problems with current Games: Knowledgebase does not reliable mark spawners as blocked field, e.g. when a treasure is on the spawner

- Knowledgebase ignores the spawner when a treasure is on the field

- Sometimes game is lost because of this issue

4. Current Issues

Problems with current Games: Knowledgebase does not reliable mark spawners as blocked field, e.g. when a treasure is on the spawner

5. Planned additions/expansions

New Collision Classifiers

-> Getting possibility to deal with more than two classes

Better trade-off between scoring and winning the game

Expansion of the calculation of the blocking mask

-> Recognition of non grid based moving enemies is necessary to calculate a more accurate blocking mask

Improve recognition from use action with range effects