1. Принцип отбора аналогов.

В отбираемые аналоги вошли существующие алгоритмы искусственного интеллекта для игр жанра RTS (стратегий реального времени), преимущественно для StarCraft. Большинство аналогов были взяты из статьи 2014 года «A review of real-time strategy game AI» [1]. В ней авторы сгруппировали используемые методы на три группы: Tactical-decision making (принятие тактических решений, микроменеджмент), Strategic-decision making (принятие стратегических решений, макроменеджмент) и Plan recognition and Learning (распознавание плана и обучение).

2. Аналоги

1) Tactical-decision making

Тактические решения принимаются при краткосрочном контроле над отдельными подразделениями.

1.1) Sarsa [2]

Алгоритм Sarsa использует Reinforcement Learning - область машинного обучения, в которой агент должен методом проб и ошибок научиться оптимальным действиям в конкретных ситуациях, чтобы максимизировать общую ценность вознаграждения [3]. Sarsa используется для обучения управлению юнитами в небольших перестрелках. Авторы использовали искусственные нейронные сети, чтобы узнать ожидаемую награду за атаку или бегство с конкретным подразделением в данном состоянии, и выбрали действие с наибольшей ожидаемой наградой в игре. Система научилась побеждать встроенные сценарии искусственного интеллекта StarCraft в среднем только в небольших перестрелках из трех единиц, при этом ни одна из вариаций не научилась побеждать встроенные сценарии в среднем в перестрелках из шести единиц.

1.2) ABCD (Alpha-Beta Considering Durations) [4]

Алгоритм ABCD заключается в поиске по дереву решений, но с ограничением глубины поиска. В конечных узлах оценивается максимальная полезность путем вызова функции оценки, и это значение распространяется вверх по дереву, на основе чего и создаётся последовательность действий.

Алгоритм разрабатывался для SparCraft - симуляции игры StarCraft c возможностью произвольно манипулировать состоянием (https://code.google.com/archive/p/sparcraft/).

Основным ограничением этой системы является то, что из-за комбинаторного взрыва возможных действий и состояний по мере увеличения количества юнитов, количества возможных действий в StarCraft и ограничения по времени в 5 мс на игровой кадр поиск позволит использовать только до восьми юнитов с каждой стороны в битве двух игроков. С другой стороны, лучшие результаты могут быть достигнуты с помощью моделирования противника, поскольку поиск может включать известные действия противника вместо поиска по всем возможным действиям противника. Когда это было протестировано на стратегиях с идеальной моделью каждого противника, поиск смог достичь по крайней мере 95%-ного коэффициента выигрыша против каждого из сценариев в симуляции.

1.3) Bayesian model (Байесовская модель) [5]

Бот, основанный на Байесовской модели, учитывает цели, возможности и угрозы подразделений, чтобы решить, в каком направлении перемещать подразделения в бою. Модель рассматривает каждый из своих сенсорных входных сигналов как часть уравнения вероятности, которое может быть решено, учитывая данные (потенциально полученные с помощью RL) о распределении входных сигналов относительно направления перемещения, чтобы определить вероятность того, что единица должна двигаться в каждом возможном направлении. Можно выбрать наилучшее направление или перебрать вероятности направления, чтобы избежать перемещения двух единиц в одно и то же место. Эта байесовская модель работает в паре с иерархическим конечным автоматом для выбора различных наборов поведения, когда подразделения вступают в бой с вражескими силами или избегают их, или ведут разведку. Созданный бот был очень эффективен против встроенного искусственного интеллекта StarCraft.

1.4) rtNEAT [6]

Нейроэволюция — это метод, который использует эволюционный алгоритм для создания или обучения искусственной нейронной сети. rtNEAT - нейроэволюционный алгоритм для разработки как топологии, так и весов соединений нейронных сетей для управления отдельными подразделениями в StarCraft. В этом алгоритме каждый юнит имеет свою собственную нейронную сеть, которая получает входные данные из источников окружающей среды (таких как близлежащие юниты или препятствия) и абстракции, определенные вручную (такие как количество, тип и “качество” близлежащих юнитов), и выдает решение атаковать, отступать или двигаться влево или вправо. Во время игры производительность юнитов оценивается с помощью разработанной вручную функции пригодности, а плохо работающие агенты юнитов заменяются комбинациями наиболее эффективных агентов. rtNEAT тестировался в очень простых сценариях 12 против 12 юнитов на квадратной арене, где все юниты с каждой стороны являются либо единицами рукопашного боя, либо единицами дальнего боя. В таких ситуациях он учится побеждать встроенный в StarCraft искусственный интеллект и некоторых других ботов. Однако остается неясным, насколько хорошо он справится с большим количеством единиц или набором различных типов единиц.

2) Strategic-decision making

Принятие стратегических решений происходит на высоком уровне с учётом долгосрочных целей. Для совершения разумных действий на стратегическом уровне многие исследователи создали системы планирования. Эти системы способны определять последовательность действий, которые необходимо предпринять в конкретной ситуации для достижения определенных целей. Это сложная проблема из–за неполной доступной информации – “туман войны” скрывает области поля боя, которые находятся вне поля зрения дружественных подразделений, - а также огромные пространства состояний и действий и множество одновременных неиерархических целей. С помощью систем планирования исследователи надеются позволить ИИ играть на человеческом уровне, одновременно сокращая затраты на разработку по сравнению со сценариями, обычно используемыми в промышленности. Основными методами, используемыми для систем планирования, являются планирование на основе конкретных случаев (CBP), автономия, основанная на достижении целей (GDA), и иерархическое планирование.

2.1) Darmok [7]

Darmok был разработан для игры в WARGUS, реализацию знаменитой Warcraft II с открытым исходным кодом. Использует метод Case-Based Planning (CBP, планирование на основе конкретных случаев). Случаи изучаются из игровых записей с комментариями человека, причём в каждом случае подробно описываются цели, которых человек пытался достичь с помощью определенных последовательностей действий в определенном состоянии. Эти случаи затем могут быть адаптированы и применены в игре, чтобы попытаться изменить игровое состояние. Рассуждая о дереве целей и подцелей, которые необходимо выполнить, можно выбрать кейсы и связать их вместе в план для достижения общей цели - победы в игре.

Системы CBP могут проявлять низкую реактивность на стратегическом уровне и чрезмерную реактивность на уровне действий, не реагируя на изменения ситуации на высоком уровне до тех пор, пока действие на низком уровне не завершится неудачей, или отказываясь от всего плана из-за неудачи одного действия [8].

2.2) Generalized Sequential Patterns [9]

Generalized Sequential Patterns - алгоритм последовательного анализа шаблонов. GSP работает, выполняя серию сканирований последовательностей данных, каждый раз совершая поиск частых шаблонов на один элемент длиннее, чем при предыдущем сканировании. Таким образом, обучаясь на игровых записях, система извлекает шаблоны поведения в разных игровых ситуациях. Генерируемые закономерности создаются как на микро-, так и на макроуровне. Полученные шаблоны в последствии можно использовать для генерации сложных задач для модели Hierarchical Task Network (HTN, Иерархическая сеть задач), которую никогда напрямую не использовали в области RTS AI.

2.3) Behavior Trees from Observations

Behavior Trees (BTs, деревья поведения) - это иерархические целевые структуры, которые выглядят несколько похожими на иерархические сети задач (HTN), но вместо того, чтобы использоваться для динамической генерации планов, BTs используются для хранения и выполнения планов [10]. Behavior Trees from Observations (на основе наблюдений) - система, способная автоматически извлекать знания о предметной области из примеров поведения экспертов, с небольшим количеством предположений о предметной области, и иметь возможность быстро реагировать на изменения состояния во время выполнения. Полученные знания представляются BTs, которая идеально подходит для выполнения одной задачи в пределах предметной области. Кроме того, результирующее BT может быть настроен вручную, поэтому этот подход может быть использован в качестве начального шага, за которым следует доработка человеком, в процессе определения нового поведения для агента[11].

2.4) EISBot [12]

EISBot использует модель Goal-Driven Autonomy (GDA) для одновременного анализа в нескольких масштабах. Используемый язык поведения способен формировать планы с ожиданиями относительно результата. Если возникает неожиданная ситуация или событие, система может записать это как несоответствие, сгенерировать объяснение, почему это произошло, и сформировать новую цель для пересмотра плана, что позволяет системе соответствующим образом реагировать на непредвиденные события. Изначально было невозможно изучить цели, ожидания или стратегии, поэтому эти знания приходилось вводить и обновлять вручную, но более поздние усовершенствования позволили извлечь их из демонстрации [13].

2.5) Soar [14]

Soar - когнитивная архитектура, использующая пространственную визуальную систему для ведения разведки и определения пути, а также рабочую память для хранения воспринимаемой и аргументированной информации о состоянии. Однако в настоящее время она ограничена частичной игрой в StarCraft, используя только базовые казармы и подразделения пехоты для ведения боя и используя жестко заданные локации для размещения зданий.

2.6) Build order optimization [15]

Алгоритм оптимизации сборки использует автоматизированный подход поиска в пространстве состояний для планирования заказов на сборку в играх RTS. Алгоритм сосредоточен на одной цели: найти план строительства желаемого набора единиц и зданий за минимальное время. Домен RTS упрощается за счет абстрагирования сбора ресурсов от нормы дохода на одного работника, предполагая, что размещение зданий и перемещение юнитов занимает постоянное количество времени, и полностью игнорируя противников. Игнорирование оппонентов довольно разумно для начала игры, поскольку, как правило, взаимодействие с оппонентами невелико, и это означает, что планировщику не приходится иметь дело с неопределенностью и внешними воздействиями на состояние. Данный метод требует экспертных знаний, чтобы обеспечить целевое состояние для их достижения.

Из-за вычислительных затрат на планирование позже в игре планирование было ограничено 120 секундами вперед, с перепланировкой каждые 30 секунд. Это привело к созданию более коротких или равных по длине планов для игроков-людей в начале игры и планов аналогичной длины в среднем (с большей дисперсией) позже в игре. Еще предстоит выяснить, насколько хорошо этот метод будет работать на более поздних этапах игры, поскольку оценивались только первые 500 секунд, а во второй половине поиск занял значительно больше времени. Тем не менее, это, по-видимому, эффективный способ получения почти оптимальных заказов на сборку, по крайней мере, для раннего и среднего этапов игры StarCraft.

3) PlanRecognition

Некоторые исследователи сосредоточились на подзадаче определения стратегии противника, которая особенно сложна в играх RTS из-за неполной информации о действиях противника, скрытой “туманом войны”. Большинство методов распознавания планов используют существующую библиотеку планов для сопоставления при попытке распознавания стратегии, но некоторые методы позволяют распознавать планы без каких-либо предопределенных планов. Часто данные извлекаются из широкодоступных файлов повторов опытных игроков-людей.

Одним из примеров решения задачи распознавания плана является Дедуктивное распознавание плана, идентифицирующее план путем сравнения ситуации с гипотезами ожидаемого поведения для различных известных планов. Наблюдая за конкретным поведением, можно сделать вывод о предпринимаемом плане, даже если полное знание недоступно. Система HICOR[16] выполняет распознавание плана в StarCraft путем сопоставления наблюдений за своим противником со всеми известными стратегиями, которые могли бы создать данную ситуацию. Затем она моделирует возможные планы, чтобы определить ожидаемые будущие действия своего противника, оценивая вероятность планов на основе новых наблюдений и отбрасывая планы, которые не совпадают. Используемый метод требует значительных человеческих усилий для описания всех возможных планов в структуре типа дерева решений.

3. Критерии сравнения аналогов.

1) Существует ли бот, реализовывающий описанный метод ИИ или представлен только алгоритм.

2) Способен ли бот провести полную игру от начала до конца, или же обеспечивает только ограниченный функционал для определённой области игры.

3) Обучаемость. Может ли рассматриваемый алгоритм обучаться на игровых записях.

4) Зависимость от экспертных знаний и многократных повторов.

Может ли предложенный метод искусственного интеллекта быть использован при отсутствии игровых записей. Например, можно ли будет применить эту технику при создании искусственного интеллекта для новой игры или не такой популярной, как StarCraft.

5) Учёт долгосрочных целей.

Учитывает ли метод ИИ цели на стратегическом уровне при выборе действий или планировании.

6) Превосходство над встроенным ИИ.

Насколько реализация ИИ превосходит встроенный ИИ игры.

7) Участие в соревнованиях.

Участвовал ли и какое место занял бот в соревнованиях среди ботов игры StarCraft (SSCAIT, AIIDE, CIG или других).

4. Таблица сравнения аналогов.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Sarsa | ABCD | Bayesian model | Darmok | GSP |
| Реализован бот | + | + | + | + | - |
| Бот полноценный | - | - | - | + | - |
| Обучаемость | + | + | + | + | + |
| Отсутствие зависимости от экспертных знаний | + | + | + | - | - |
| Учёт долгосрочных целей | - | - | - | + | + |
| Превосходство над встроенным ИИ | +-1 | +2 | +2 | + | - |
| Участие в соревнованиях | - | - | - | - | - |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | BTfO | EISBot | Soar | BOO | HICOR |
| Реализован бот | - | + | + | + | + |
| Бот полноценный | - | + | - | - | + |
| Обучаемость | + | + | + | + | + |
| Отсутствие зависимости от экспертных знаний | - | + | + | + | - |
| Учёт долгосрочных целей | + | + | + | + | + |
| Превосходство над встроенным ИИ | - | + (73%)3 | - | + | + |
| Участие в соревнованиях | - | - | - | - | - |

1 Только при определённом кол-ве юнитов.

2 Только на тактическом уровне.

3 До использования обучения.

Как видно из сравнения разных методов искусственного интеллекта, мало кто из исследователей создал на их основе полноценного бота, показавшего превосходство над встроенным ИИ или участвовавшего в соревнованиях.

Структура большинства современных ботов возникает в результате попыток разложить игру на иерархию более мелких подзадач, таких как стратегия более высокого уровня, тактика, управление боевыми единицами, анализ местности и сбор разведданных. Боты различаются по сложности, и, хотя многие из них основаны на правилах, наиболее эффективные боты в настоящее время используют более сложные методы искусственного интеллекта, такие как поиск/планирование в реальном времени, предварительно обученные контроллеры нейронных сетей и онлайн-обучение во время соревновательных игр. Ниже представлен обзор на ботов, принимавших участие в различных соревнованиях. Данные взяты из статей «StarCraft AI Competitions, Bots and Tournament Manager Software» (2018) [17] и « A Survey of Real-Time Strategy Game AI Research and Competition in StarCraft» (2013) [18].

1. CherryPi.

Реализован как набор разнородных модулей, которые могут быть добавлены, удалены или настроены с помощью командной строки. Такая конструкция позволяет легко заменять отдельные модули на обучающие модули или проводить узкие эксперименты, используя только их подмножество. Модули взаимодействуют путем обмена двумя типами элементов данных через архитектуру blackboard: парами ключ-значение и так называемыми объектами UPC (единица измерения, позиция, команда). В общем случае UPC представляет собой вероятностное распределение подразделений, вероятностное распределение позиций и единую команду.

Заказы на сборку представлены в виде набора приоритетных запросов, помещенных в очередь, которая заполняет требования и применяет оптимизации (например, выделяет ресурсы для строительства точно в срок).

Решения о сражении или бегстве принимаются путем объединения юнитов в группы и запуска боевой симуляции. Боевое моделирование CherryPi работает с наивной политикой «атаковать ближайшую цель». Поиск путей с учетом угроз используется для поиска наименее опасных маршрутов в безопасное место. Выбор стратегии высокого уровня в нескольких играх основан на алгоритме UCB1, выбирающем из фиксированного списка стратегий против каждой расы.

2. cpac.

Сочетает жестко закодированные правила с многослойной сетью персептронов для производства единиц. Сеть обучается на парах состояние-действие, извлеченных из большого набора данных BroodWar games 8.

3. ForceBot.

Написан на агентном языке программирования GOAL, разработанном поверх BWAPI для программирования когнитивных агентов. Поскольку язык GOAL предназначен для реализации многоагентных систем, все подразделения ForceBot имеют своего собственного соответствующего агента с конкретными убеждениями и целями. Каждый агент более или менее следует шаблону искусственного интеллекта, основанному на правилах.

4. Iron bot

Представляет собой децентрализованную мультиагентную систему, где каждое устройство управляется высокоавтономным индивидуальным агентом, способным переключаться между 25 вариантами поведения. Все его подразделения преследуют одну простую цель: добраться до главной вражеской базы и уничтожить ее. Он часто кажется ботом-преследователем, что связано с тем, что его юниты ведут себя в основном индивидуально. Существуют также так называемые “экспертные” агенты, которые автономно рекомендуют, как следует расходовать ресурсы и какие подразделения следует обучать, основываясь на эвристике.

5. KillAll

Большая часть его функциональности основана на правилах. Однако его производственный модуль использует Q-learning для выбора типов единиц для производства на основе текущей ситуации.

6. Krasi0bot

Участвует в соревнованиях каждый год с 2010 года и все еще активно развивается. По словам автора, изначально он начинался как бот, основанный на правилах, но в настоящее время в некоторой степени использует генетические алгоритмы и нейронные сети. Поскольку бот не является открытым исходным кодом, эти данные не могут быть проверены. Krasi0bot играет за расу землян и известен своими сильными оборонительными возможностями и широким разнообразием реализованных стратегий.

7. LetaBot

Использует поиск по дереву Монте-Карло (MCTS) для планирования перемещения групп юнитов по карте. Использует совместный поиск путей для сбора ресурсов и интеллектуальный анализ текста для извлечения заказов на сборку из статей Liquипедии[19].

8. McRave

Все решения бота основаны на текущем составе вражеских юнитов – нет жестко запрограммированных технических вариантов. Бот также создает модель противника и использует ее для выбора заказов на построение.

9. MegaBot

Для каждой игры MegaBot выбирает один из трех подходов, каждый из которых реализован как отдельный бот (Skynet, Xelnaga или NUSBot). В начале игры выбирается алгоритм, использующий эпсилон-жадный алгоритм. После игры определяется награда (+1, 0, -1 за победу, ничью и проигрыш соответственно), а значение выбранного алгоритма обновляется с помощью инкрементной версии взвешенного по времени экспоненциального среднего (правило обновления Q-learning).

10. PurpleWave

Принятие решений в основном основано на иерархических сетях задач. Для микроменеджмента он использует гибридный подход команды/мультиагента и кластеризацию ближайших соседей. Затем бот моделирует результаты сражений и предлагает тактику для отрядов, минимизируя тактические подходы каждой стороны (например, “атаковать”, “убегать” или “сражаться с рабочими”). В конце концов, каждое подразделение принимает тактическое предложение к сведению, но ведет себя независимо. Устройства выбирают примерно из двух десятков простых, многократно используемых вариантов поведения. Эвристика бота включает в себя использование потенциальных полей для перемещения юнитов. Стратегии выбираются на основе результатов предыдущих игр, расы, карты и количества стартовых позиций. В нем есть график выбора стратегии, например, порядок открытия сборки в сочетании с переходами в середине игры и составами в конце игры.

11. StarcraftGP

Это первая программа StarCraft meta-bota, которая автономно создает программу, которая автономно играет в StarCraft. В настоящее время StarcraftGP v0.1 использует (линейное) генетическое программирование и может напрямую писать код на C++. Его первые творения: Salsa и Tequila, были первыми ботами, не написанными непосредственно человеком, для участия в международных конкурсах.

12. Steamhammer

Использует сложную боевую симуляцию для прогнозирования исхода сражений. Бот также использует иерархическое реактивное управление для юнитов. Для производства протоссов и терран Steamhammer использует поиск по ветвям и границам, в то время как зерги в настоящее время основан на правилах.

13. tscmoo

Бот не использует внешних библиотек: у него есть свой собственный код моделирования боя для прогнозирования исхода сражений, он не использует BWTA для анализа местности и даже имеет свой собственный поиск путей с учетом угроз для отдельных юнитов. Бот является одним из самых стратегически разнообразных и выбирает среди своих многочисленных стратегий, основываясь на их успехе в предыдущих играх. Последние версии бота экспериментировали с рекуррентными нейронными сетями для высокоуровневой стратегии и принятия решений о порядке сборки.

14. UAlbertaBot

UAlbertaBot использует динамическую систему поиска заказов на сборку на основе эвристического поиска (BOSS) для планирования всех своих заказов на сборку в режиме реального времени, а также систему моделирования боевых действий StarCraft под названием SparCraft для оценки результатов внутриигровых сражений. Бот использует результаты предыдущих игр против конкретных противников, чтобы выбрать стратегию для реализации в начале каждой игры, причем каждая стратегия определяется во внешнем файле конфигурации JSON. Его разработка была сосредоточена на простоте использования и модификации, и как таковая стала основой для более чем 10 других ботов, включая LetaBot, Overkill и Steamhammer. В 2017 году UAlbertaBot стал CommandCenter, первым ботом, способным играть как в BroodWar, так и в StarCraft 2.

15. ZZZKbot

Его общая стратегия реализует 4 простые стратегии: 4-pool, Speedlings, Hydralisks и Mutalisks. Если первоначальный бросок не завершает игру, бот переключается либо на Mutalisks, либо на Hydralisks для поздней игры, одновременно исследуя обновления для всех своих юнитов. Бот записывает информацию о выигрыше / проигрыше для каждого соперника и использует эту информацию для выбора наилучшей комбинации параметров стратегии для будущих игр на основе правил. Большинство правил ботов для управления подразделениями и микроменеджмента - это простые правила поведения, основанные на приоритизации экспертных знаний.

Критерии сравнения аналогов.

1) Обучаемость. Может ли рассматриваемый бот обучаться на игровых записях.

2) Учёт долгосрочных целей.

Учитывает ли бот цели на стратегическом уровне при выборе действий или планировании.

3) Эвристический поиск.

Используется ли эвристический поиск при выборе действий.

4) Боевая симуляция.

Используется ли боевая симуляция при выборе действий для боевых подразделений.

5) Призовые места.

Какие места занял бот в соревнованиях среди ботов игры StarCraft (SSCAIT, AIIDE, CIG).

Таблица сравнения аналогов по критериям.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | CherryPi | cpac | ForceBot | Iron | KillAll |
| Обучаемость | - | - | - | - | + |
| Учёт долгосрочных целей | + | ? | + | - | ? |
| Эвристический поиск | - | + | - | - | + |
| Боевая симуляция | + | + | - | - | + |
| Призовые места | SSCAIT2017(1), AIIDE2018(2) | - | - | SSCAIT2018(3), AIIDE2016(1), CIG2016(2), AIIDE2017(3) | - |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Krasi0bot | LetaBot | McRave | MegaBot |
| Обучаемость | + | - | - | - |
| Учёт долгосрочных целей | + | + | ? | + |
| Эвристический поиск | + | + | - | - |
| Боевая симуляция | + | + | + | + |
| Призовые места | AIIDE2010(2), SSCAIT2013(1) | SSCAIT2014(1), SSCAIT2015(1) SSCAIT2016(1), CIG2014(3), AIIDE2014(3), SSCAIT2017(2), CIG2016(3) | - | AIIDE2011(1), CIG2011(1), AIIDE2012(1), CIG2012(1), CIG2013(1), AIIDE2013(2) |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | PurpleWave | StarcraftGP | Steamhammer | tscmoo |
| Обучаемость | - | + | - | + |
| Учёт долгосрочных целей | + | ? | - | + |
| Эвристический поиск | + | - | + | + |
| Боевая симуляция | + | - | + | + |
| Призовые места | SSCAIT2018(2), AIIDE2017(2), CIG 2017(3) | - | - | AIIDE2015(1), CIG2016(1), CIG2015(2), CIG2017(2), AIIDE2016(3) |

|  |  |  |
| --- | --- | --- |
|  | UAlbertaBot | ZZZKbot |
| Обучаемость | - | + |
| Учёт долгосрочных целей | + | + |
| Эвристический поиск | + | - |
| Боевая симуляция | + | - |
| Призовые места | AIIDE2011(2), CIG2011-2013(2), AIIDE2012(3),  AIIDE2013(1), SSCAIT2012-2016(3) | CIG2015(1), AIIDE2015(2), AIIDE2016(2), AIIDE2017(1), CIG2017(1) |

**Список источников.**

1. Robertson G., Watson I. A review of real-time strategy game AI //Ai Magazine. – 2014. – Т. 35. – №. 4. – С. 75-104

2. Shantia, A.; Begue, E.; and Wiering, M. 2011. Connectionist reinforcement learning for intelligent unit micro managementinStarCraft. Presented at the International Joint Conference on Neural Networks

3. Sutton, R. S., and Barto, A. G. 1998. Reinforcement learning: An introduction. Cambridge Massachusetts: MIT Press

4. Churchill, D.; Saffidine, A.; and Buro, M. 2012. Fast heuristic search for RTS game combat scenarios. In Proceedings of the AIIDE Conference, 112–117

5. Synnaeve, G., and Bessiere, P. 2011b. A bayesian model for RTS units control applied to StarCraft. In Proceedings of the IEEE Conference on Computational Intelligence and Games, 190–196

6. Gabriel, I.; Negru, V.; and Zaharie, D. 2012. Neuroevolution based multi-agent system for micromanagement in real-time strategy games. In Proceedings of the Fifth Balkan Conference in Informatics, 32–39. ACM.

7. Ontanón S. et al. Case-based planning and execution for real-time strategy games //International Conference on Case-Based Reasoning. – Springer, Berlin, Heidelberg, 2007. – С. 164-178.

8. Palma R. et al. Combining expert knowledge and learning from demonstration in real-time strategy games //International Conference on Case-Based Reasoning. – Springer, Berlin, Heidelberg, 2011. – С. 181-195.

9. Leece M. A., Jhala A. Sequential pattern mining in Starcraft: Brood War for short and long-term goals //Tenth Artificial Intelligence and Interactive Digital Entertainment Conference. – 2014.

10. G. Florez-Puga, M. Gomez-Martin, P. Gomez-Martin, B. Diaz-Agudo, and P. Gonzalez-Calero, “Query-enabled behavior trees,” IEEE Trans. Computational Intelligence and AI in Games, vol. 1, no. 4, pp. 298–308, Dec 2009

11. Robertson G., Watson I. Building behavior trees from observations in real-time strategy games //2015 International symposium on innovations in intelligent systems and applications (INISTA). – IEEE, 2015. – С. 1-7.

12. Weber B., Mateas M., Jhala A. Applying goal-driven autonomy to starcraft //Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment. – 2010. – Т. 6. – №. 1. – С. 101-106.

13. Weber, B.; Mateas, M.; and Jhala, A. 2012. Learning from demonstration for goal-driven autonomy. In Proceedings of the AAAI Conference on AI, 1176–1182

14. Soar-SC [Электронный ресурс] URL: <https://github.com/bluechill/Soar-SC> (Дата обращения: 30.11.2022)

15. Churchill, D., and Buro, M. 2011. Build order optimization in StarCraft. In Proceedings of the AIIDE Conference, 14–19.

16. Kabanza, F.; Bellefeuille, P.; Bisson, F.; Benaskeur, A.; and Irandoust, H. 2010. Opponent behaviour recognition for real-time strategy games. In Proceedings of the AAAI Workshop on Plan, Activity, and Intent Recognition.

17. Čertický M. et al. Starcraft AI competitions, bots, and tournament manager software //IEEE Transactions on Games. – 2018. – Т. 11. – №. 3. – С. 227-237.

18. Ontanón S. et al. A survey of real-time strategy game AI research and competition in StarCraft //IEEE Transactions on Computational Intelligence and AI in games. – 2013. – Т. 5. – №. 4. – С. 293-311.

19. Liquipedia [Электронный ресурс] URL: <https://liquipedia.net/> (Дата обращения: 10.12.2022)