> Вырезанная техническая часть из Proposal к CV в лабу

Co-evolution

https://github.com/aimclub/GOLEM/issues/47

Высокоуровневое описание

Предлагается реализовать кооперативную ко-эволюцию: решение будет собираться по частям из нескольких популяций подграфов.

Разделение на подзадачи

Вероятно, будет несколько фиксированных слотов — каждый для своей популяции и по этой схеме будет собираться итоговый граф. О том, какая именно структура, знает только пользователь GOLEM, поэтому метод сборки передаётся в качестве параметра.

Теоретически, структура соединения слотов тоже может эволюционировать (как отдельная популяция), но это в большинстве случаев дискредитирует идею разделения ответственности ⟶ специализации популяций. Кроме того, при коэволюции разбиение на подзадачи зачастую осуществляют через анализ статистических взаимосвязей переменных, но в случае графа это не представляется возможным — его структура не фиксирована.

Применение для мультимодальных данных в FEDOT

Одно из полезных применений — работа с мультимодяльными данными в FEDOT: будет происходить эволюция `Pipiline`-ов, которые объединяются в общий `Pipiline`, оборачиваясь каждая в `AtomizedModel`.

Структура общего пайплайна, вероятно, должна быть такой: несколько популяций ответственны за перевод входных данных конкретного типа в промежуточное представление, а после этого ещё одна популяция аггрегирует их, получая ответ.

При этом на соединениях между разными популяциями выделяется «типизация»: нужно, чтобы данные, выдаваемые подграфами одной популяции соответствовали входным другой популяции, соединяющейся с ней в пайплайне. И эти внешние соединения (свои для каждой популяции) получают особенную поддержку в генетических операторах (как входы и выходы в pipeline — только модели с правильным типом входов/выходов могут вставать на это место).

Мотивация

Такое разделение ответственности (разные популяции для разных типов данных + аггрегатор) — максимально естественно и гарантирует высокую степерь *независимости задач, выполнямых разными популяцями*, а это в точности условие, требующееся для применимости кооперативной коэволюции, поэтому можно ожидать экспоненциального от количества популяций ускорения.

На предметом уровне это будет значить, что особи станут более «прозрачными» и гибкими (их структура будет более точно разложена перед оптимизатором). Соответственно, будет меньше появляться «комплексов» (термин позаимствован у Фрейда…): ситуации, при которой неудачная часть графа остаётся в популяции только потому, что когда-то зацепилась за удачную. Таким образом, зона ответственности будет менее расплывчатой.

Генетические операторы

Для оценки особи нужно протестировать, «как она себя ведёт в комбинации с особями из других популяций», пробуя разные комбинации.

Проводится несколько испытаний, каждое из которых — оценка фитнесса всего графа, собранного из каких-то обоей разных популяций. Выбирать испытания нужно, чтобы:

- Каждая особь поучаствовала в достаточном количестве испытаний
- Отдавалось предпочтение более точной оценки fitness-а у лучших особей

При этом нужно быть более чувствительным к лучшим значениям, чем усреднение. В пределе — максимум.

Классический подход к выбору collaborator-ов — k-fold. Однако практика показывает, что максимум для фиксированного индивида почти всегда достигается на одном из лучших членов каждой из остальных популяций, поэтому, вероятно, стоит использовать подход iCCEA: поддерживать архив этих лучших представителей (похожую задачу решает ~~GloryHall~~ HallOfFame, только ещё нужно учитывать расстояние между особями).

Когда fitness особей оценён, формирование новой популяции происходит одним из стандатных методов.

Для оценки хода оптимизации используется external fitness — в случае кооперативной коэволюции это fitness лучшей комбинации из текущих популяций.

Diversity Maintenance (Niching)

— подход, в котором отдельно наказывается за отсутствие разнообрзия (здесь описаны основные методы: https://www.intechopen.com/chapters/8533).

Измерение разнообразия

В случае графов измерять разнообразие можно с помощью аппроксимации Graph edit distance — сама задача NP полная, но есть аппроксимации за ≈линейное время (В целом, обычно бо'льшая часть времени — вычисление фитнесса, так что можем позволить) Кроме структуры, вероятно, нужно использовать пользовательскую информацию для нахождения расстояния.

Можно использовать информацию о хронологии эволюции: как напрямую учитывать родословеную при селекции, так и помогать GED находить оптимальное решение.

Как влиять на ход эволюции

Предлагается использовать один из классических подходов, описанных, например, в Essentials of metaheuristics. Глобально, есть два пути:

- Модификация fitness
- Учитывать на уровне селекции

Область применения

Как минимум — параметр разнообразия полезно мониторить, чтобы не действовать вслепую.

Кроме того, имея метрику и рычажок для её увеличения/уменьшения (в некотором смысле — exploration vs. exploitation), логично вручить их адаптирующему агенту.

Что уже реализовано

На данный момент ведётся работа над минимизацией дубликатов: https://github.com/aimclub/GOLEM/issues/89, что можно назвать частным случаем поддержки разнообразия, а для оценки разнообразия используются метрики, не использующие графовую структур до конца (сравниваются скалярные параметры графов).