0. Введение

Данная книга представляет конспект лекций по **метаэвристикам** (***metaheuristics***) для студентов бакалаврского этапа обучения (*undergraduate*). Она создана на базе курса, который я читал весной 2009 г., потому что сейчас недостаточно литературы для студентов младших курсов, посвященной этой теме. Т.к. это *конспект* для *бакалавриата* по специфичному курсу, то он содержит ряд особенностей. Во-первых, он написан неформальным языком и содержит множество моих суждений и предпочтений. Во-вторых, в них излагается мало теории и дается небольшое количество примеров: в основном приведены описания алгоритмов и наброски интуитивных объяснений почему и в каких случаях стоит их применять. В-третьих, здесь полно алгоритмов, больших и маленьких. На мой взгляд, этот конспект может служить хорошим дополнением к учебнику, но также может использоваться и отдельно, в качестве быстрого введения в рассматриваемое научное направление.

Я не могу дать **никаких гарантий**, касающихся корректности алгоритмов или текста данного конспекта. В действительности, в них содержится большое количество ошибок. *Пожалуйста*, сообщайте о найденных ошибках (с исправлениями!). Некоторые сложные алгоритмы представлены в упрощенном виде, такие случаи отмечаются в пояснениях в тексте.

0.1 Что такое метаэвристика?

Метаэвристика – достаточно неудачное название[1] для описания большого подраздела, в действительности основного, в **стохастической оптимизации** (***stochastic optimization***). Стохастическая оптимизация является большим классом алгоритмов и методов, которые в той или иной степени используют случайность для поиска оптимального (или достижимого оптимального) решения сложных задач. Метаэвристики – наиболее общие алгоритмы из этого класса и применяются для решения широкого спектра задач.

Какие задачи рассматриваются? В деле *Якобеллиса против штата Огайо* (*Jacobellis v. Ohio*) (1964г., дело о нецензурном содержании) судья Верховного суда США Поттер Стюарт написал свое знаменитое изречение:

> Сегодня я не буду более пытаться дать определение тем материалам, которые я понимаю, и которые нужно втиснуть в краткое определение. Вполне возможно, что у меня никогда не получится сделать это внятно. **Но я узнаю их, когда увижу**, и рассматриваемый в деле фильм к ним не относится.

Метаэвристики применяются к задачам типа *я узнаю их, когда увижу*. Это алгоритмы, которые используются для поиска решения на задачи, в которых доступно очень мало вспомогательной информации: вы не знаете, на что похоже оптимальное решение, неизвестно, каким способом необходимо искать это решение, имеющейся эвристической информации явно недостаточно, а последовательный перебор не подходит, т.к. пространство поиска слишком велико. *Но* если есть потенциальное решение, его можно проверить и установить, насколько оно хорошо. Т.е. можно узнать качество решения, когда оно доступно.

---

[0] Перевод раздела из книги Luke S. Essentials of Metaheuristics. A Set of Undergraduate Lecture Notes. Zeroth Edition. Online Version 1.2. July, 2011 (`http://cs.gmu.edu/~sean/book/metaheuristics/`). Перевел – Юрий Цой, 2011 г. Любые замечания, касающиеся перевода, просьба присылать по адресу yurytsoy@gmail.com

Данный текст доступен по адресу: `http://qai.narod.ru/GA/meta-heuristics_0.pdf`

[1] Обычно я называю этот подраздел *стохастической оптимизацией* (*stochastic optimization*). Однако это слишком обширный термин, который включает такие важные алгоритмы как **методы Монте-Карло с цепями Маркова** (***Markov Chain Monte Carlo, MCMC***) или **сэмплирование по Гиббсу** (***Gibbs Sampling***), не попадающие в рассматриваемый подраздел. В последнее время часто используется термин метаэвристика, но на мой взгляд, этот термин – глубоко ошибочный и странный. Если я слышу «метаобсуждение», то я понимаю это как: *обсуждение об обсуждении*. Аналогично, услышав «метаэвристика», я воспринимаю: *эвристика об (или для) эвристики*. Однако это не имеет ни малейшего отношения к самим алгоритмам! Возможно, реже используемый термин **оптимизация черного ящика** (***black box optimization***) подошло бы лучше, хотя и он также имеет дополнительный балласт. «**Слабые методы**» (***weak methods***) – тоже слишком широкий термин: он не означает использование стохастичности. Иногда используется термин **стохастический поиск** (***stochastic search***): однако для меня проблема «поиска» означает все или ничего: решение либо найдено, либо нет. А у нас не поиск, у нас – оптимизация.

Предположим, к примеру, что необходимо найти оптимальный набор действий для робота, играющего в футбол. Имеется симулятор действий робота и можно протестировать любой набор действий и присвоить этому набору оценку (хороший набор действий можно увидеть). Также имеется общее определение для набора действий, т.е. можно представить, что он из себя представляет. Однако нет ни малейшей идеи о том, каким должен быть оптимальный набор, и о том, как к нему прийти.

Простейший вариант решения задачи в подобной ситуации – **случайный поиск** (***Random Search***): будем просто генерировать случайные наборы поведений, пока не закончится отведенное на поиск время, и вернем наилучший найденный набор. Но прежде чем отчаяться и взяться за случайный поиск, рассмотрим альтернативу, называюмую **локальным поиском** (***Hill-Climbing***). Начнем работать со случайным набором действий. Затем применим к нему небольшое случайное изменение и протестируем полученную новую версию. Если она окажется лучше, то предыдущую версию откинем. В противном случае, отбрасывается новая версия. После сделаем еще одно небольшое случайное изменение текущей версии набора (т.е. той, которая не была отвергнута). Если новая версия лучше, то отбрасываем текущую, иначе не принимаем новую. Повторяем процесс насколько это возможно.

Локальный поиск – это простой метаэвристический алгоритм. Он использует эвристическое предположение о свойстве пространства поиска, которое обычно актуально для многих задач: схожие решение часто ведут себя подобным образом (и часто имеют близкое качество), поэтому небольшие модификации решений обычно влекут небольшие и объяснимые изменения в их качестве, что позволяет взбираться на «вершину горы качества» для получения хороших решений. Такое эвристическое предположение является одной из **ключевых особенностей метаэвристических методов**. И действительно, практически все метаэвристики объединяют локальный поиск со случайным.

0.2 Алгоритмы

Данный конспект включает множество алгоритмов, больших и маленьких. Есть все, начиная от целых алгоритмов эволюционных вычислений и заканчивая такими простыми решениями как перемешивание элементов массива. Приводятся алгоритмы даже для большинства тривиальных и очевидных задач. Я постарался быть как можно более педантичным, чтобы не возникало лишних вопросов.

Если у алгоритма есть аргументы, то они перечисляются вначале и отделяются от кода алгоритма пустой строкой. Если параметров нет, то описание алгоритма начинается сразу после заголовка. В некоторых случаях алгоритм представлен несколькими функциями, каждая из которых отмечается как **процедура**. Иногда будут использоваться общие статические глобальные переменные, которые вводятся вначале алгоритма и помечаются ключевым словом **global**. Вот пример простого алгоритма:

Алгоритм 1 Сортировка пузырьком

```
1: v ← ⟨v_1, ..., v_l⟩ сортируемый вектор {Здесь приводятся параметры алгоритма.}
                                          {Затем идет пустая строка.}

   {Алгоритм начинается здесь.}
2: repeat
3:    swapped ← false {← всегда обозначает присваивание.}
4:    for i от 1 до l − 1 do
5:       {Заметьте, что значение l определено через v_l в строке 1.}
6:       if v_i > v_{i+1} then
7:          Поменять местами v_i и v_{i+1}
8:          swapped ← true
9:       end if
10:   end for
11: until swapped = false {= означает «равняется»}
12: return v {Некоторые алгоритмы ничего не возвращают, в них оператор return отсутствует.}
```

Обратите внимание, что параметры функции определены очень обобщенно, поэтому в ряде случаев при вызове функций параметры явно не задаются, если их значение, необходимое для правильной работы, очевидно. Ну да, в некоторых местах мне следовало бы соблюдать формальности. Можете

подать на меня в суд[2].

0.3 Обозначения

Больших особенностей нет. Но чтобы поставить все точки над i и перечеркнуть все t:

- **Числовые** и **булевские** величины обозначены строчными буквами, греческими символами или словами ($n, \lambda, min, popsize$). «Пустой» или **«нулевой»** элемент по умолчанию обозначен как $\square$. Интервалы обозначаются так: от 1 до n, включая концы. Сами интервалы могут содержать как целые, так и вещественные значения. Символ $\leftarrow$ всегда обозначает «запись в», а символ $=$ – обычно значит «равняется».

- **Потенциальные решения** (иногда называемые **особями**, **частицами** или **следами**) обозначаются большими буквами или словами, начинающимися с заглавной буквы ($Best, S, P_i$). Некоторые потенциальные решения являются векторами и описываются как векторы (см. далее). Другие включают некоторое количество **компонент**, часто обозначаемых $C_1, \ldots, C_j$. Потенциальным решениям может быть поставлено в соответствие некое подобие **качества** (**приспособленности**), обычно посредством функции Качество(S) или Приспособленность(P_i). При этом значение качества может быть изменено. Обычно качество представлено числом, но в ряде случаев (для многокритериальной оптимизации) оно описывается набором чисел, называемых **целями**. Значение цели O_j, ассоциированное с особью P_i, вычисляется с помощью функции вроде ОценкаЦели(O_j, P_i). В ряде случаев особям или иным объектам могут быть приписаны другие атрибуты.

- **Наборы** (сумки, группы, пулы, списки, мультимножества) являются группами объектов, причем объекты не обязательно уникальны. Фактически в лекциях редко используются множества и часто встречаются наборы. Наборы обозначаются заглавной буквой, например, P, и содержать некоторое количество элементов, заключенных в скобки, $\{P_1, \ldots, P_n\}$. Размер P равен $\|P\|$ или (в данном случае) n. Принадлежность множеству (или ее отсутствие) обозначаются как $\in$ ($\notin$). Как правило в наборе присутствует внутренний порядок, поэтому элементы набора имеют уникальное обозначение (P_4 или P_1) и можно осуществлять их перебор, скажем, так (**for** каждый элемент $P_i \in P$ **do** $\ldots P_i$). Наборы обычно можно только читать, однако элементы, входящие в набор, можно подвергать внутренним изменениям.

  Оператор объединения ($\cup$) используется для конкатенации множеств (например, $P \leftarrow P \cup Q$). Этот прием часто применяется при добавлении элемента к набору: $P \leftarrow P \cup \{R_j\}$. Символ вычитания обозначает удаление элементов, входящих во второй набор, как в $P - M$, либо удаление конкретного элемента из текущего набора ($P \leftarrow P - \{P_2\}$). Во всех случаях подразумевается, что новый набор наследует внутренний порядок, присутствовавший в предыдущем наборе или наборах.

  Наиболее общими примерами набора являются **популяции**. Популяции чаще всего обозначены как P или Q. Иногда будет необходим набор популяций, который будет обозначен так: $P^{(1)}, \ldots, P^{(n)}$. Особь с номером j в популяции $P^{(i)}$ записывается как $P^{(i)}_j$.

  Потомки иногда обозначаются как C_a, C_b и т.д. Однако это не означает наличия набора с именем C (несмотря на то, что его можно ввести без какого-либо ущерба).

- **First-in First-out очереди** рассматриваются как наборы, обладающие дополнительной возможностью добавления элемента в конец очереди, изъятие элемента из ее начала или вставка в произвольную позицию.

- **Векторы** обозначаются посредством верхних стрелок ($\overrightarrow{x}, \overrightarrow{Best}$) и содержат некоторое количество компонент, включенных в угловые скобки $\langle x_1, \ldots, x_n \rangle$. В отличие от наборов, векторы можно модифицировать. Элемент вектора можно заменить другим элементом. Из вектора можно не только удалять слоты (позиции), но и расширять вектор, путем их добавления в конец вектора. Я буду использовать векторы вместо наборов, когда будет необходимость явного изменения элементов в конкретных позициях.

- **Кортежи** – это векторы в которых слоты поименованы, например, $\overrightarrow{r} \leftarrow \langle t_{lock}, t_{data} \rangle$, а не пронумерованы.

[2] Не подавайте на меня в суд. Спасибо.

- Двумерные **массивы** или **матрицы** обозначены заглавными буквами (A), а к их элементам можно обращаться обычным путем: $A_{i,j}$. Как и для векторов, элементы матриц могут быть заменены.

- **Распределения вероятностей** и другие **модели** обозначаются заглавными буквами, к примеру, T. Распределения и модули можно сконструировать и обновлять. Затем с их использованием генерируются случайные числа. Таже в тексте дисперсии обозначаются σ^2, стандартные отклонения – σ, а средние часто обозначают символом μ.

- При передаче вместе с данными имена **функций** приводятся в нижнем регистре, как f или f(узел).