Санкт - Петербургский Государственный Университет

Математико - механический факультет

Кафедра - системного программирования

Адаптивная оптимизация сервера , обрабатывающего очередь заданий

Дипломная работа студента 545 группы

Ле Чунг Хьеу

Научный руководитель :	О.Н. Граничин. /...../
Рецензент :	А.С. Лопатин. /...../
Зав. кафедры информатики :	А.Н. Терехов. /...../

Санкт - Петербург
2007

Содержание

1 Введение

В дипломной работе рассматривается - проблема эффективного обслуживания сервером очереди заданий. Эта задача - классическая для теории массового обслуживания (см. [2]). Существует множество методов для ее решения, были разработаны различные алгоритмы и проведены многочисленные исследования.

В последнее время развитие электронной техники приблизилось к порогу создания устройств с характеристиками искусственного интеллекта. При решении многих практических задач возникла необходимость эффективного использования новых математических алгоритмов оптимизации,оценивания неизвестных параметров динамических систем, оптимального и адаптивного управления объектами. В книге профессора Граничина и Поляка [1] предлагается использовать рандомизированный алгоритм стохастической аппроксимации. Другой подход к решению таких проблем рассматривается методами обучения с подкреплением (см. [3]). Цель моей работы изучить эти два алгоритма на примере задачи о эффективном обслуживании сервером очереди заданий и выявить преимущества и недостатки каждого из них.

Многомерная стохастическая оптимизация играет важную роль в анализе и управлении многими техническими системами. Практически во всех прикладных задачах оптимизации используется какой-нибудь математический алгоритм, который последовательно отыскивает нужное решение, так как аналитическое решение задачи редко доступно. Для решения трудных многомерных задач оптимизации и разрабатывались рандомизированные алгоритмы стохастической аппроксимации с возмущением на входе (см. [1]).

Эти алгоритмы в последнее время активно используются в таких областях, как статистическая оценка параметров, управление с обратной связью, основанная на моделировании оптимизация, обработка сигналов и изображений, планирование экспериментов. Их существенная особенность состоит в том, что для аппроксимации градиента функции потерь, лежащей в основе многих алгоритмов, требуется только одно или два измерения функции независимо от размерности задачи оптимизации. Эта особенность обеспечивает относительную легкость представления алгоритмов и позволяет добиться существенного уменьшения затрат на решение, особенно в задачах оптимизации по большому количеству переменных. Кроме того, в случае зашумленных измерений функции потерь для этих алгоритмов удается доказать состоятельность доставляемых оценок при очень незначительных ограничениях на помехи.

Обучение с подкреплением представляет класс задач, в которых агент, действуя в определенной среде, должен найти оптимальную стратегию взаимодействия с ней. Информация для обучения агента предоставляется в форме награды (простого скалярного платежа), имеющей определенное количественное значение для каждого перехода среды из одного состояния в другое. Задача агента, таким образом,сводится к максимизации суммарного платежа.

Агенту не говорят, какие действия предпринимать. Задача агента выяснить, какие действия приводят к большинству награды, пробуя каждое из них. В самых интересных случаях, действия могут затронуть не только непосредственную награду но также и следующую ситуацию и, соответственно, все последующие награды. Метод проб и ошибок поиск и отложенной награды - два самых важных отличительных признаке обучения с подкреплением.

Опишем три фундаментальных класса методов для решения задачи обучения с подкреплением (см. [3]): Динамическое программирование, методы Монте Карло, и Обучение на временных разностях.

У каждого класса методов есть сильные и слабые стороны. Методы Динамического программирования хорошо развиты математически, но требуют полной и точной модели окружающей среды. Методы Монте Карло не требуют точной модели и концептуально просты. Наконец, методы TD не требуют никакой модели, но более сложны для анализа.

2 Задача о сервере

2.1 Постановка задачи

- Рассмотрим задачу повышения эффективности сервера (см. [2]). Пусть он используется для обслуживания очереди заданий, процесс поступления которых является случайным. Будем считать, что вероятностное распределение времени обслуживания задания сервером зависит от вещественного параметра x, который требуется выбрать с целью минимизации среднего времени ожидания клиентами $L(x)$ вместе с некоторой стоимостью использования $q(x)$ параметра x

$$f(x) = q(x) + L(x) \equiv q(x) + \lim_N \frac{1}{N} \sum_{i=1}^{N} Ey_i(x) \to \min$$

  где $E\{.\}$ — символ математического ожидания, $y_i(x)$ — время, которое задание, законченное i-м по счету, ожидало ("простаивало") в сервере до момента своего завершения. Требуется найти параметр θ, минимизирующий $f(x)$ по x из некоторого компактного множества $\Theta \in \mathbb{R}$ (области определения).

- Вообще говоря, функцию $L(x)$ очень трудно вычислить. У экспериментатора нет возможности выбирать значения функции в некоторой окрестности точки x для формирования оценки производной в этой точке, т. е. эту задачу оптимизации не решить традиционными средствами.Более адекватными подходами к решению являются алгоритмы основанные на использовании подсчитываемых на определенных интервалах времени значений эмпирических функционалов качества $F(x, w)$, которые при определенных естественных предположениях о регулярности входного потока и эксплуатационных характеристик сервера являются несмещенными оценками значений функционала качества

$$f(x) = E\{F(x, w)\}$$

- Пусть $F_t(x, w)$ - случайная функция дискретного времени $t = 1, 2, \ldots$, векторного параметра x и вектора неконтролируемых возмущений w. Обозначим через f_t "текущий"функционал среднего риска:

$$f_t(x) = E_w\{F_t(x, w)\}$$

  Точку минимума функционала f_t обозначим :

$$\theta_t = arg \min_x f_t(x)$$

- Требуется по наблюдениям (может быть с дополнительными помехами) за случайными величинами $F_t(x_n, w_n), n = 1, 2, \ldots$ построить последовательность оценок $\{\widehat{\theta}_n\}$, отслеживающих изменения параметра θ_t, для которой $\|\widehat{\theta}_n - \theta_t\| \to \min$ в каком-нибудь смысле.

2.2 Алгоритм адаптации шага диспетчеризации сервера

- Вернемся к задаче эффективного обслуживания очереди заданий одним сервером. Будем предполагать, что процесс поступления этих заданий от клиентов носит случайный характер. Входные данные определяют последовательность пар чисел $\{(t_i^n, d_i)\}$, в которых первое значение соответствует времени прихода задания на сервер, а второе - времени, необходимому для его исполнения. Естественно, что для сервера значения d_i^{in} считаются неизвестными.

- Для простоты считаем, что сервер обрабатывает задания следующим образом:
  - задается шаг диспетчеризации $x \qquad (x \in \Theta = [a, b])$;
  - первое поступившее задание начинает выполняться сервером сразу после поступления;
  - поступающие задания попадают в естественную очередь;
  - в каждый момент времени сервер обрабатывает только одно текущее задание;
  - в моменты времени кратные выбранному шагу диспетчеризации x, если сервер свободен, то он переключается на выполнение первого в очереди задания, если он занят, то выполняющееся задание прерывается и отправляется в конец очереди, а сервер переключается на первое задание из очереди. На эту операцию загрузки-выгрузки сервер всегда затрачивает фиксированное время d_{load}.

- Выберем натуральное достаточно большое число N (например, $N = 10000$). Разобьем общее время работы сервера на определенные интервалы - такты: $t_0, t_1, \ldots, t_k, \ldots$, по правилу:

  t_0 - время начала работы сервера;

  t_1 - время окончания выполнения первых N заданий;

  $\ldots$

  t_k - время окончания выполнения k-ой серии из N заданий;

  $\ldots$

- Отметим, что вообще-то задавать такты можно различными способами: либо начиная новый интервал по завершении обработки группы из некоторого фиксированного числа заказов, либо после прихода на сервер фиксированного числа заданий, либо выбирая фиксированный интервал времени сервера, и т. п. Все разумные правила выбора тактов во многом эквивалентны.

- Опишем способ получения значений y_k для эмпирического функционала $F(x, w)$. Считаем, что для каждого k-го такта каким-то образом задано соответствующее значение оценки $\widehat{\theta}_k$ для регулируемого параметра x (интервала времени обслуживания задания на сервере). Для удобства перенумеруем задания, выполненные сервером, в порядке окончания их обработки. Каждому заданию наряду с парой чисел (t_i^{in}, d_i), определенных уже в момент его поступления, сопоставим еще одно - t_i^{out} - время окончания его обработки. Определим значения эмпирической функции качества как среднее время бесполезного ожидания заданий до их завершения с учетом стоимости использования загрузки-выгрузки обработчика

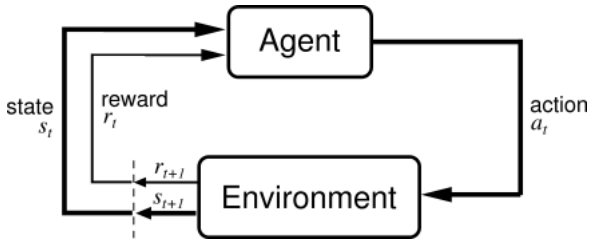
$$y_k = \frac{(t_k - t_{k-1}) * d_{load}}{\widehat{\theta}_k} + \frac{1}{N} \sum_{t_i^{out} \in [t_{k-1}, t_k]} (t_i^{out} - t_i^{in} - d_i).$$

3 Обучение с подкреплением

3.1 Метод обучения с подкреплением

- *Обучение с подкреплением* (Reinforcement Learning, RL) представляет класс задач, в которых *агент*, действуя в определенной среде, должен найти оптимальную стратегию взаимодействия с ней. Информация для обучения агента предоставляется в форме «награды» (простого скалярного платежа), имеющей определенное количественное значение для каждого перехода среды из одного состояния в другое. Задача агента, таким образом, сводится к максимизации суммарного платежа.

- *Обучение с подкреплением* происходит в результате получения агентом наград и наказаний, поступающих из внешней среды.



  В текущей состоянии $s_t \in \mathcal{S}$, где $\mathcal{S}$ - набор возможных состояний , агент выполняет *действие* $a_t \in \mathcal{A}(s_t)$, где $\mathcal{A}(s_t)$ - набор действий, доступных в состоянии s_t, получает подкрепление $r_t \in \mathcal{R}$ и попадает в следующий состояние $s_{t+1}, t = 1, 2, \ldots$

- На каждом шаге, агент осуществляет отображение из множества состояний в множество возможных действий. Это отображение называют *стратегией* и обозначают π_t, где $\pi_t(s, a)$ - вероятность того что $a_t = a$, если $s_t = s$.

- *Награда* , простой скалярный платеж, имеющии определенное количественное значение для каждого перехода среды из одного состояния в другое, $r_t \in \mathcal{R}$. Цель агента – максимизировать суммарную награду. Это означает максимизировать не непосредственную награду, но совокупную награду в конечном счете.

- Ожидаемая прибыль (*expected return*) :

$$R_t = r_{t+1} + r_{t+2} + r_{t+3} + \cdots + r_T,$$

где T - заключительный шаг времени, и $r_{t+1}, r_{t+2}, \ldots, r_T$ последовательность наград, полученных после шага времени t.

- Ожидаемая дисконтирующая привыль (*expected discounted return*) :

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1},$$

где γ – *коэффициент забывания* $(0 < \gamma < 1)$, учитывающий, что чем дальше агент "заглядывает" в будущее, тем меньше у него уверенность в оценке ожидаемой награды.

- Рассмотрим, как общая внешная среда могла бы ответить во время $t+1$ на действие, предпринятое на времени t. В самом общем, причинном случае этот ответ может зависеть от всего, что случилось ранее. В этом случае динамика может быть определена только, определяя полное распределение вероятности:

$$Pr\{s_{t+1} = s', r_{t+1} = r | s_t, a_t, r_t, s_{t-1}, a_{t-1}, \ldots r_1, s_0, a_0\}$$

Если выполняется свойство Маркова , то :

$$Pr\{s_{t+1} = s', r_{t+1} = r | s_t, a_t\}$$

- RL задачу, которая удовлетворяет свойству Маркова , называют *Markov decision process*, или MDP. Если набор состояния и набор действия конечны, то это называют *finite Markov decision process* (конечный MDP).

- Учитывая любое состояние и действие, s и a, вероятность каждого возможного следующего состояния, s',

$$\mathcal{P}^a_{s,s'} = Pr\{s_{t+1} = s' | s_t = s, a_t = a\}$$

Эти количества называют *вероятностями перехода.*

- Точно так же учитывая любое текущее состояние и действие, s and a, вместе с любым следующим состоянием, s', ожидаемая ценность следующей награды

$$\mathcal{R}^a_{s,s'} = E\{r_{t+1} | s_t = s, a_t = a, s_{t+1} = s'\}$$

3.1.1 Оценочная функция

- Почти все RL алгоритмы основано на оценке оценочной функцией (*value functions*) - функции состоян-ия (или пар состояния - действия), приближает ожидаемый возврат для текущего состояния (или для текущего состояния после выполнения каждого из действий).

- Вспомним, что стратегия, π, является отображение от каждого состояния, $s \in \mathcal{S}$, и действие, $a \in \mathcal{A}(s)$, к вероятности $\pi(s, a)$ принятия действия a когда в состоянии s. Неформально, оценочная функция состояния s под стратегией π, обозначенный $V^\pi(s)$, является средним ожидаемым возвратом по всем инвестициям, начинаясь в s и после π после того. Для MDPs

$$V^\pi(s) = E_\pi\{R_t | s_t = s\} = E_\pi\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s\}$$

Назовем функцию V^π оценочная функция состояния для стратегии π

$$Q^\pi(s, a) = E_\pi\{R_t | s_t = s, a_t = a\} = E_\pi\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s, a_t = a\}$$

Назовем Q^π оценочная функция состояний и действий для стратегии π.

- *Уравнение Белмана* для V^π

$$
\begin{aligned}
V^{\pi}(s) &= E_{\pi}\{R_t|s_t = s\} \\
&= E_{\pi}\{\sum_{k=0}^{\infty}\gamma^k r_{t+k+1}|s_t = s\} \\
&= E_{\pi}\{r_{t+1} + \gamma\sum_{k=0}^{\infty}\gamma^k r_{t+k+2}|s_t = s\} \\
&= \sum_a \pi(s,a)\sum_{s'}\mathcal{P}^a_{ss'}[\mathcal{R}^a_{ss'} + \gamma E_{\pi}\{\sum_{k=0}^{\infty}\gamma^k r_{t+k+2}|s_{t+1} = s'\}] \\
&= \sum_a \pi(s,a)\sum_{s'}\mathcal{P}^a_{ss'}[\mathcal{R}^a_{ss'} + \gamma V^{\pi}(s')],
\end{aligned}
$$

- Стратегия π определена, чтобы быть лучше чем или равным стратегией π', если его ожидаемое возвращение больше чем или равен тому из π' для всех состояний. Другими словами, $\pi \geqslant \pi'$, если и только если $V^{\pi}(s) \geqslant V^{\pi'}(s)$ для всех $s \in \mathcal{S}$.
- Есть всегда по крайней мере одна стратегия, которая лучше чем или равна всей другой стратегией. Это - *оптимальная стратегия.*
- *Оптимальная оценочная функция состояния*, обозначенный V^*, и определенный как

$$V^*(s) = \max_{\pi} V^{\pi}(s) \qquad \forall s \in \mathcal{S},$$

- *Оптимальная оценочная функция состояния и действия*, обозначенный Q^*, и определенный как

$$Q^*(s,a) = \max_{\pi} Q^{\pi}(s,a) \qquad \forall s \in \mathcal{S}, a \in \mathcal{A}(s),$$

- Таким образом, мы можем написать Q^* в терминах V^* следующим образом:

$$Q^*(s,a) = E\{r_{t+1} + \gamma V^*(s_{t+1})|s_t = s, a_t = a\}$$

- Уравнение Белмана для V^*, или уравнение оптимальности Белмана.

$$
\begin{aligned}
V^*(s) &= \max_{a\in\mathcal{A}(s)} Q^{\pi^*}(s,a) \\
&= \max_a E_{\pi^*}\{R_t|s_t = s, a_t = a\} \\
&= \max_a E_{\pi^*}\{\sum_{k=0}^{\infty}\gamma^k r_{t+k+1}|s_t = s, a_t = a\} \\
&= \max_a E_{\pi^*}\{r_{t+1} + \gamma\sum_{k=0}^{\infty}\gamma^k r_{t+k+2}|s_t = s, a_t = a\} \\
&= \max_a E\{r_{t+1} + \gamma V^*(s_{t+1})|s_t = s, a_t = a\} \\
&= \max_{a\in\mathcal{A}(s)}\sum_{s'}\mathcal{P}^a_{ss'}[\mathcal{R}^a_{ss'} + \gamma V^*(s')],
\end{aligned}
$$

- Уравнение оптимальности Белмана для Q^*

$$
\begin{aligned}
Q^*(s,a) &= E\{r_{t+1} + \gamma\max_{a'} Q^*(s_{t+1},a')|s_t = s, a_t = a\} \\
&= \sum_{s'}\mathcal{P}^a_{ss'}[\mathcal{R}^a_{ss'} + \gamma\max_{a'} Q^*(s',a')],
\end{aligned}
$$

3.2 Элементарные методы решения

- В этой части мы описываем три фундаментальных класса методов для решить RL задачу:
  - Динамическое программирование.
  - Метод Монте Карло.
  - Обучение на временных разностях.

3.2.1 Динамическое программирование

- Мы обычно предполагаем, что внешная среда - конечный MDP, его набор состояния и наборы действия, $\mathcal{S}$ и $\mathcal{A}(s)$, для $\forall s \in \mathcal{S}$, является конечным.
- Его динамика задается рядом вероятностей перехода,

$$\mathcal{P}^a_{ss'} = Pr\{s_{t+1} = s' | s_t = s, a_t = a\},$$

и ожидаемыми непосредственными наградами,

$$\mathcal{R}^a_{ss'} = E\{r_{t+1} | a_t = a, s_t = s, s_{t+1} = s'\}, \qquad \forall s \in \mathcal{S}, a \in \mathcal{A}(s), s' \in \mathcal{S}^+$$

($\mathcal{S}^+$ - $\mathcal{S}$ плюс предельное состояние, если задача является эпизодической).

- Ключевая идея относительно DP, и относительно RL вообще, является использованием оценочной функции , чтобы организовать и структурировать поиск хорошей стратегий.

$$\begin{aligned} V^*(s) &= \max_a E\{r_{t+1} + \gamma V^*(s_{t+1}) | s_t = s, a_t = a\} \\ &= \max_a \sum_{s'} \mathcal{P}^a_{ss'} [\mathcal{R}^a_{ss'} + \gamma V^*(s')] \end{aligned}$$

$$\begin{aligned} Q^*(s,a) &= E\{r_{t+1} + \gamma \max_{a'} Q^*(s_{t+1}, a') | s_t = s, a_t = a\} \\ &= \sum_{s'} \mathcal{P}^a_{ss'} [\mathcal{R}^a_{ss'} + \gamma \max_{a'} Q^*(s', a')] \end{aligned}$$

для всех $s \in \mathcal{S}$, $a \in (s)$, и $s' \in \mathcal{S}^+$

Оценка стратегии

- Сначала мы полагаем, как вычислить оценочную функцию состояния V^π для произвольной стратегии π. Это называют оценкой стратегии (*policy evaluation*) в литературе DP. Мы также именуем это как проблема предсказания .$\forall s \in \mathcal{S}$

$$\begin{aligned} V^\pi(s) &= E_\pi\{r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots | s_t = s\} \\ &= E_\pi\{r_{t+1} + \gamma V^\pi(s_{t+1}) | s_t = s\} \\ &= \sum_a \pi(s,a) \sum_{s'} \mathcal{P}^a_{ss'} [\mathcal{R}^a_{ss'} + \gamma V^\pi(s')] \end{aligned}$$

где $\pi(s,a)$ - вероятность принятия действия a в состоянии s под стратегией π. Существование и уникальность V^π гарантируются пока любой $0 < \gamma < 1$

- Рассмотрите последовательность приблизительных оценочных функций $V_0, V_1, V_2, \ldots$, и каждое последовательное приближение получено при использовании уравнения Белмана для V^π

$$\begin{aligned} V_{k+1}(s) &= E_\pi\{r_{t+1} + \gamma V_k(s_{t+1}) | s_t = s\} \\ &= \sum_a \pi(s,a) \sum_{s'} \mathcal{P}^a_{ss'} [\mathcal{R}^a_{ss'} + \gamma V_k(s')] \qquad \forall s \in \mathcal{S} \end{aligned}$$

- Этот алгоритм называют *iterative policy evaluation.*

Усовершенствование стратегии

- Предположим, что мы определили оценочную функцию V^π для произвольной детерминированной стратегии π.Мы знаем, как хороший это должно следовать за текущей стратегией от s - который является $V^\pi(s)$ - но это было бы лучше или хуже, чтобы измениться на новую стратегию?

$$\begin{aligned} Q^\pi(s,a) &= E_\pi\{r_{t+1} + \gamma V^\pi(s_{t+1}) | s_t = s, a_t = a\} \\ &= \sum_{s'} \mathcal{P}^a_{ss'}[\mathcal{R}^a_{ss'} + \gamma V^\pi(s')] \end{aligned}$$

  Ключевой критерий - больше ли это чем или меньше чем $V^\pi(s)$. Если это больше - то есть, если бы лучше выбрать a однажды в s и после того следовать π, чем это должно было бы следовать π все время - тогда можно было бы ожидать, что с этим, чтобы быть лучше все еще, чтобы выбрать a каждый раз s сталкиваются, и что новая стратегия фактически была бы лучшим повсюду.

- Позвольте π и π' быть любой парой детерминированной стратегии такой что, для всех $s \in \mathcal{S}$,

$$Q^\pi(s, \pi'(s)) \geqslant V^\pi(s)$$

  тогда стратегия π' должна быть лучше чем, π.

$$V^{\pi'}(s) \geqslant V^\pi(s) \qquad \forall s \in \mathcal{S}$$

$$\begin{aligned} V^\pi(s) &\leq Q^\pi(s, \pi'(s)) \\ &= E_{\pi'}\{r_{t+1} + \gamma V^\pi(s_{t+1}) | s_t = s\} \\ &\leq E_{\pi'}\{r_{t+1} + \gamma Q^\pi(s_{t+1}, \pi'(s_{t+1})) | s_t = s\} \\ &= E_{\pi'}\{r_{t+1} + \gamma E_{\pi'}\{r_{t+2} + \gamma V^\pi(s_{t+2})\} | s_t = s\} \\ &= E_{\pi'}\{r_{t+1} + \gamma r_{t+2} + \gamma^2 V^\pi(s_{t+2}) | s_t = s\} \\ &\leq E_{\pi'}\{r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \gamma^3 V^\pi(s_{t+3}) | s_t = s\} \\ &\vdots \\ &\leq E_{\pi'}\{r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \gamma^3 r_{t+4} + \ldots | s_t = s\} \\ &= V^{\pi'}(s) \end{aligned}$$

- Другими словами, чтобы рассмотреть новую стратегию, π' :

$$\begin{aligned} \pi'(s) &= arg\max_a Q^\pi(s,a) \\ &= arg\max_a E\{r_{t+1} + \gamma V^\pi(s_{t+1}) | s_t = s, a_t = a\} \\ &= arg\max_a \sum_{s'} \mathcal{P}^a_{ss'}[\mathcal{R}^a_{ss'} + \gamma V^\pi(s')] \end{aligned}$$

- Процесс создания новой стратегии, которая изменяет к лучшему оригинальную стратегию называют усовершенствованием стратегии (*policy improvement*).

- Предположим, что новая стратегия, π' столь же хороша как, но не лучше чем, старая стратегия π, тогда $V^\pi = V^{\pi'}$, и для всех $s \in \mathcal{S}$:

$$\begin{aligned} V^{\pi'}(s) &= \max_a E\{r_{t+1} + \gamma V^{\pi'}(s_{t+1}) | s_t = s, a_t = a\} \\ &= \max_a \sum_{s'} \mathcal{P}^a_{ss'}[\mathcal{R}^a_{ss'} + \gamma V^{\pi'}(s')] \end{aligned}$$

  $V^{\pi'}$ должен быть V^*, π и π' должен быть оптимальной стратегией.

Повторение стратегии

- Этот способ находить оптимальную стратегию называют повторением стратегии (*policy iteration*)

$$\pi_0 \longrightarrow^E V^{\pi_0} \longrightarrow^I \pi_1 \longrightarrow^E V^{\pi_1} \longrightarrow^I \pi_2 \longrightarrow^E \ldots \longrightarrow^I \pi^* \longrightarrow^E V^*$$

  где $\longrightarrow^E$ обозначает оценка стратегии (*evaluation*) и $\longrightarrow^I$, обозначает усовершенствование стратегии (*improvement*).

- Поскольку конечный MDP имеет только конечное число стратегий, этот процесс должен сходиться к оптимальной стратегией и оптимальной оценочной функцией в конечном числе повторений.

Повторение ценности

- Фактически, шаг оценка стратегии повторения стратегии может быть обрезанным несколькими способами, не теряя гарантии сходимости повторения стратегии. Один важный специальный случай - то, когда оценка стратегии остановлена после только одна зачистка . Этот алгоритм называют повторением ценности (*value iteration*).

$$\begin{aligned} V_{k+1}(s) &= \max_a E\{r_{t+1} + \gamma V_k(s_{t+1}) | s_t = s, a_t = a\} \\ &= \max_a \sum_{s'} \mathcal{P}^a_{ss'} [\mathcal{R}^a_{ss'} + \gamma V_k(s')] \end{aligned}$$

  для всех $s \in \mathcal{S}$, произвольного V_0, последовательности $\{V_k\}$ показывает, чтобы сходиться к V^* при тех же самых условиях, которые гарантируют существование V^*.

Обобщенное повторение стратегии

- Повторение стратегии состоит из двух одновременных, взаимодействующих процессов, один создание оценочной функции, совместимой с текущей стратегией (оценка стратегии), и другое создание стратегии, жадной относительно текущей оценочной функции (усовершенствование стратегии) .

- В повторении ценности, только единственное повторение оценки стратегии выполнено между каждым усовершенствованием стратегии.

- Мы используем обобщенное повторение стратегии (*generalized policy iteration*) (GPI), чтобы обратиться к общей идее позволить оценке стратегии и интермедии процессов усовершенствования стратегии, независимой от степени детализации и других деталей двух процессов.

- Почти все RL методы хорошо описаны как GPI.

Эффективность методы динамического программирования

- Методы DP фактически весьма эффективны. Если мы игнорируем несколько технических деталей (худший случай), то методы DP времени берут, чтобы найти, что оптимальная стратегия - полином от числа состояний и действий. Если n и m обозначают число состояний и действий, это означает, что метод DP берет множество вычислительных операций, который является меньше чем некоторая многочленная функция n и m. Метод DP, как гарантируют, найдет оптимальную стратегию в многочленное время даже при том, что общее количество стратегий - m^n.

- DP сравнительно лучше подходит для обработки больших наборов состояний чем конкурирующие методы, типа прямого поиска и линейного программирования. Практически, методы DP могут использоваться с компьютерами, чтобы решить MDPs с миллионами состояний.

3.2.2 Методы Монте Карло

- В отличие от методов DP, здесь мы не принимаем полное знание окружающей среды. Методы Монте Карло требуют только *опыт* - типовые последовательности состояний, действий, и наград от или моделируемого взаимодействия онлайн с окружающей средой.
- Методы Монте Карло - способы решить RL задачу, основанную на усреднении получаемых результатов. Чтобы гарантировать, что четкие возвращения являются доступными, мы определяем методы Монте Карло только для некоторых задач.

Оценка стратегии в методом Монте Карло

- *The every-visit* метод MC оценивает $V^{\pi}(s)$ как среднее число возвращений после всех посещений s в ряде эпизодов.
- *The first-visit* метод MC составляет в среднем только возвращения после первых посещений s.
- The first-visit MC и the every-visit MC сходятся к $V^{\pi}(s)$ как число посещений (или сначала посещает) к s, стремится к бесконечности.
- Проблема оценки стратегии за ценности действия состоит в том, чтобы оценить $Q^{\pi}(s,a)$, средний ожидаемый доход по всем инвестициям, начиная в состоянии s, совермая действие a, и после этого следуя стратегии π. Метод the every-visit MC оценивает ценность пары состояния - действия как среднее число возвращений, которые следовали за посещениями состояния, в котором было отобрано действие. Метод the first-visit MC составляет в среднем возвращения после первого раза в каждом эпизоде, что состояние было посещено, и действие было отобрано.
- Единственное осложнение состоит в том, что много возможных пар состояния - действия никогда не могут посещаться. Если π - детерминированная стратегия, то в следующем π каждый будет наблюдать возвращения только для одного из действий от каждого состояния. Без возвращений, чтобы составить в среднем, оценки Монте Карло других действий не будут улучшаться с опытом. Чтобы сравнивать альтернативы, мы должны оценить ценность всех действий от каждого состояния.
- Это гарантирует, что все пары состояния - действия будут посещены бесконечное количество раз в пределе бесконечного числа эпизодов. Мы называем это *exploring starts*.

Управление методом Монте Карло

- Мы теперь готовы рассмотреть, как оценка Монте Карло может использоваться в управлении, то есть, приближать оптимальную стратегию. Общая идея должна следовать согласно тому же самому образцу как в главе DP, то есть, согласно идее относительно обобщенного повторения политики (GPI).
- В этом методе, мы выполняем переменные полные шаги оценки стратегии и усовершенствования стратегии, начиная с произвольной стратегии π_0 и заканчиваясь оптимальной стратегией и оптимальной оценочной функцией состояний и действий:

$$\pi_0 \longrightarrow^{E} Q^{\pi_0} \longrightarrow^{I} \pi_1 \longrightarrow^{E} Q^{\pi_1} \longrightarrow^{I} \pi_2 \longrightarrow^{E} \ldots \longrightarrow^{I} \pi^* \longrightarrow^{E} Q^*$$

  где $\longrightarrow^{E}$ обозначает *оценка стратегии* и $\longrightarrow^{I}$ обозначает *усовершенствование стратегии*.
- Оценка стратегии сделана точно как описано в предыдущей часту.
- Усовершенствование стратегии производится жадно относительно текущей оценочной функции.

$$\pi(s) = arg\max_{a} Q(s,a).$$

  Усовершенствование стратегии тогда может быть сделано, построением каждого π_{k+1} как жадной стратегии относительно Q^{π_k}.

$$
\begin{aligned}
Q^{\pi_k}(s, \pi_{k+1}(s)) &= Q^{\pi_k}(s, arg\max_a Q^{\pi_k}(s,a)) \\
&= \max_a Q^{\pi_k}(s,a) \\
&\geq Q^{\pi_k}(s, \pi_k(s)) \\
&= V^{\pi_k}(s)
\end{aligned}
$$

- Мы сделали два маловероятных предположения выше того, чтобы легко получить эту гарантию сходимости для метода Монте Карло. Каждый был этим, эпизоды имеют запуски исследования, и другой был то, что оценка стратегии могла быть сделана с бесконечным числом эпизодов.

On-Policy - Управление методом Монте Карло

- Как мы можем избежать маловероятного предположения об исследовании запусков? Есть два подхода к обеспечению этого, которые мы называем *on-policy* и *off-policy* методами .
- В on-policy методах управлая стратегия вообще мягка, это означае что $\pi(s,a) > 0$ для всех $s \in \mathcal{S}$ и все $a \in \mathcal{A}(s)$.
- On-policy метод, который мы представляем в этой секции, использует стратегию, который является всеми нежадными действиями, даются минимальную вероятность выбора, $\frac{\epsilon}{|\mathcal{A}|}$, и остающаяся большая часть вероятности, $1 - \epsilon + \frac{\epsilon}{|\mathcal{A}|}$, дается жадному действию.

$$
\begin{aligned}
Q^{\pi}(s, \pi'(s)) &= \sum_a \pi'(s,a) Q^{\pi}(s,a) \\
&= \frac{\epsilon}{|\mathcal{A}|} \sum_a Q^{\pi}(s,a) + (1-\epsilon) \max_a Q^{\pi}(s,a) \\
&\geq \frac{\epsilon}{|\mathcal{A}|} \sum_a Q^{\pi}(s,a) + (1-\epsilon) \sum_a \frac{\pi(s,a) - \frac{\epsilon}{|\mathcal{A}|}}{1-\epsilon} Q^{\pi}(s,a) \\
&= \frac{\epsilon}{|\mathcal{A}|} \sum_a Q^{\pi}(s,a) - \frac{\epsilon}{|\mathcal{A}|} \sum_a Q^{\pi}(s,a) + \sum_a \pi(s,a) Q^{\pi}(s,a) \\
&= V^{\pi}(s)
\end{aligned}
$$

Таким образом, по теореме усовершенствования стратегии, $\pi' \geq \pi$ (то есть, $V^{\pi'}(s) \geq V^{\pi}(s)$, для всех $s \in \mathcal{S}$).

3.2.3 Обучение на временных разностях

- Обучение на временных разностях - сочетание идей Монте Карло и Динамического программирования.
  - Как и методы Монте Карло, методы TD могут учиться непосредственно из сырого опыта без модели динамической окружающей среды.
  - Как и методы DP, методы TD обновляют оценки, основанные частично на других ученых оценках, не ждущих заключительного результата (они улучшают),

Предсказание TD

- DP и методы Монте Карло использовали опыт чтобы решить задачу предсказания, обновить их оценку V ,V^{π}.
  - Методы Динамического Программирования :

$$
V^{\pi}(s) = \sum_a \pi(s,a) \sum_{s'} \mathcal{P}^a_{ss'} [\mathcal{R}^a_{ss'} + \gamma V^{\pi}(s')]
$$

– Методы Монте Карло:

$$V(s_t) \leftarrow V(s_t) + \alpha[R_t - V(s_t)]$$

Методы Монте Карло должны ждать до конца эпизода, чтобы определить приращение к $V(s_t)$ (только тогда - R_t известный).

– Обучение на временных разностях

$$V(s_t) \leftarrow V(s_t) + \alpha[r_{t+1} + \gamma V(s_{t+1}) - V(s_t)]$$

* Потребности методов TD ждут только до следующего шага.
* В момент времени $t+1$ они обновляют значение наблюдаемой награды r_{t+1} и оценку $V(s_{t+1})$.
* Методы TD комбинируют *sampling* Монте Карло с *bootstraping* из DP.

Преимущества методов предсказания TD

- Методы TD имеют преимущество перед методами DP, в которых они не требуют модели окружающей среды, его награды и распределений вероятности по состояниям - действиям.
- С методами Монте Карло нужно ждать до конца эпизода, потому что только тогда получается известное возвращение, тогда как с методами TD одна потребность ждет только один шаг времени.
- Для любой неподвижной стратегии π,была доказана сходимость алгоритма TD к V^π.
- TD и Монте Карло, какой из методов обучается быстрее? Какой делает более эффективное использование ограниченных данных? - никто не был в состоянии доказать математически, что один метод сходится быстрее чем другой. Практически, однако, методы TD, как обычно находили, сходились быстрее чем постоянные-α методы MC на стохастических задачах.

Sarsa: On-Policy - Управление TD

- В предыдущей части мы рассматривали переходы в зависимости от состояния и определяли ценности состояний. Теперь мы рассматриваем переходы от пары состояния действия паре состояния действия, и изучаем ценность пар состояния - действия.

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]$$

– Если s_{t+1} является предельным, то $Q(s_{t+1}, a_{t+1})$ определен как ноль.

– $(s_t, a_t, r_{t+1}, s_{t+1}, a_{t+1})$ - эта пятерка дает название *Sarsa* для алгоритма.

Q-обучение: Off-Policy - Управление TD

- Один Q-обучение шаг определяется как :

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)]$$

- В этом случае, изучаемая оценочная функция состояния и действия, Q, непосредственно приближается Q^*, оптимальная оценочная функция состояния и действия, независимо от сопровождаемой стратегии.
- Стратегия все еще имеет эффект, в котором это определяет, какие пары состояния - действия посещены и обновлены.

4 Метод обучения с подкреплением и задача о сервере

- Теперь попробуем применить метод обучения с подкреплением для решения задачи о сервере. Суть RL заключается в том, что у нас имеется набор состояний и агент должен для каждого из состояний выбрать такое действие, чтобы в сумме получить максимум наград. В задаче о сервере у нас есть множество состояний $\mathcal{S} = \{(t, d)\}$, где t - это математическое ожидания времени поступления на сервер $(k+1)$-ой и k-ой задачи, а d - это математическое ожидания времени выполнения k-ой задачи. В каждом состоянии имеется множество действий $\mathcal{A} = \{\theta\}$. Агент для каждого состояния будет выбирать какое-то действие и получать значение функции F_k по формуле:

$$F_k = \frac{(t_k - t_{k-1}) * d_{load}}{\widehat{\theta}_k} + \frac{1}{N} \sum_{t_i^{out} \in [t_{k-1}, t_k]} (t_i^{out} - t_i^{in} - d_i)$$

  Здесь значения функции выступают в форме наград.

- Цель моей программы построить стратегию выбора агентом действий из множества $\mathcal{A}$ такую, чтобы агент собрал в сумме минимум наград.

4.1 Метод Монте Карло и задача о сервере

- Алгоритм заключается в следующем:

  1 Сначала считаем значения наград для всех состояний (t, d) и для всех θ равным 50, $R(t, d, \theta) = 50$.

  2 На k-ом шаге попадая в какое-то состояние $s = (t, d)$, находим действие θ такое что $R(t, d, \theta)$ минимально и называем его θ_{min}.

  3 Теперь зная значение *epsilon* мы в $1 - epsilon$ случаев выбираем действие $\theta = \theta_{min}$, а в *epsilon* случаев произвольное.

  4 Считаем $F(t, d, \theta)$ по формуле:

$$F_k(t, d, \theta) = \frac{(t_k - t_{k-1}) * d_{load}}{\theta} + \frac{1}{N} \sum_{t_i^{out} \in [t_{k-1}, t_k]} (t_i^{out} - t_i^{in} - d_i)$$

  5 Находим значение $R(t, d, \theta)$ по формуле:

$$R(t, d, \theta) \leftarrow R(t, d, \theta) + \alpha[F_k(t, d, \theta) - R(t, d, \theta)]$$

  6 Находим значение *epsilon* для следующего шага .

  7 Переходим ко второму пункты алгоритма считая $k = k + 1$.

5 Рандомизированные алгоритмы стохастической аппроксимации

5.1 Формулировки и обоснования рандомизированных алгоритмов СА

- Рассмотрим для примера задачу о нахождении стационарной точки θ некоторой функции $f(.)$ (точки локального минимума или максимума) при условии, что для каждого значения $x \in \mathbb{R}$ - входа алгоритма (управляемой переменной) - наблюдается случайная величина $Y(x)$, являющаяся зашумленным значением функции $f(.)$ в точке x

$$Y(x) = f(x) + V$$

- Дж.Кифер и Дж.Вольфовиц для решения этой задачи при некоторых дополнительных ограничениях обосновали сходимость к точке θ в рекуррентной последовательности, определяемой по правилу (алгоритму)

$$\widehat{\theta}^n = \widehat{\theta}^{n-1} - \alpha_n \frac{Y(\widehat{\theta}^{n-1} + \beta_n) - Y(\widehat{\theta}^{n-1} - \beta_n)}{2\beta_n}$$

  где $\{\alpha_n\}$ и $\{\beta_n\}$ - некоторые заданные убывающие числовые последовательности с определенными свойствами.

- Основное условие - ограничение на свойства помех наблюдения, которое обычно предполагается выполненным, - это условная центрированность помех наблюдения.

- Пусть

$$g(x,\beta) = \frac{Y(\widehat{\theta}^{n-1} + \beta) - Y(\widehat{\theta}^{n-1} - \beta)}{2\beta}$$

математическое ожидание при малом β близко к значению производной функции

$$E\{g(x,\beta)\} \approx f'(x)$$

Поведение последовательности оценок, доставляемых алгоритмом стохастической аппроксимации, зависит от выбора наблюдаемых функций-статистик $g(x,\beta)$.

- Предположим, что функция $f(.)$ дважды непрерывно дифференцируема и задана наблюдаемая реализация некоторой бернуллиевской последовательности независимых случайных величин $\{\Delta_n\}$, равных с одинаковой вероятностью плюс/минус единице, некоррелированных с ошибками наблюдения на шаге n.Тогда процедуру Кифера - Вольфовица можно модифицировать, используя рандомизированную статистику

$$\widehat{g}(x,\beta,\Delta) = g(x,\beta\Delta)$$

Разложив функцию $f(x)$ по формуле Тейлора и воспользовавшись некоррелированностью Δ_n и помех наблюдения, для этой новой статистики имеем

$$E\{\widehat{g}(x,\beta\Delta)\} = f'(x) + E\{\frac{1}{\Delta}V\} + \mathcal{O}(\beta) = f'(x) + \mathcal{O}(\beta)$$

Если значения числовой последовательности $\{\beta_n\}$ в алгоритме стремятся к нулю, то в пределе эта статистика "в среднем"совпадает со значением производной функции $f(.)$. Такими же свойствами обладает и более простая статистика

$$\overline{g}(x,\beta,\Delta) = \frac{\Delta}{\beta}Y(x+\beta\Delta)$$

использующая только одно наблюдение на каждой итерации (шаге).

- Добавление в алгоритм и канал наблюдения нового случайного процесса $\{\Delta_n\}$, называемого пробным одновременным возмущением, приводит к обогащению последовательности наблюдений.

- В многомерном случае, когда $\theta \in \mathbb{R}^r$:

Для построения последовательности оценок обычная процедура Кифера-Вольфовица, основанная на конечно-разностных аппроксиациях вектора-градиента функции, использует $2r$ наблюдений на каждой итерации (по два наблюдения для аппроксимации каждой компоненты r-мерного вектора-градиента).

Рандомизированные статистики $\widehat{g}(x,\beta,\Delta)$ и $\overline{g}(x,\beta,\Delta)$ допускают более простой с вычислительной точки зрения способ обобщения на многомерный случай, использующий всего два или одно измерение функции на каждой итерации. Пусть $\{\Delta_n\}$ — бернуллиевский r-мерный случайный процесс. Тогда

$$\widehat{g}(x,\beta,\Delta) = \begin{pmatrix} \frac{1}{\Delta_1} \\ \frac{1}{\Delta_2} \\ \vdots \\ \frac{1}{\Delta_r} \end{pmatrix} \frac{Y(x+\beta\Delta) - Y(x-\beta\Delta)}{2\beta}$$

5.1.1 Постановка задачи и основные предположения

- Пусть $F(w,x) : \mathbb{R}^p \times \mathbb{R}^r \to \mathbb{R}$ - дифференцируемая по второму аргументу функция, $x^1, x^2, \ldots$ выбираемая экспериментатором последовательность точек измерения (план наблюдения), в которых в каждый момент времени $n = 1, 2, \ldots$ доступно наблюдению с аддитивными помехами v_n значение функции $F(w^n, .)$

$$y_n = F(w^n, x^n) + v_n$$

где $\{w^n\}$ — неконтролируемая последовательность случайных величин, $w^n \in \mathbb{R}^p$, имеющих одинаковое, вообще говоря, неизвестное распределение $P_w(.)$ с конечным носителем.

- Требуется по наблюдениям $y_1, y_2, \ldots$ построить последовательность оценок $\{\widehat{\theta}^n\}$ неизвестного вектора θ, минимизирующего функцию

$$f(x) = \int_{\mathbb{R}^p} F(w,x) P_w(dw)$$

- Перечислим основные предположения:

  (A) Функция $f(.)$ сильновыпуклая, т. е. имеет единственный минимум в $\mathbb{R}^r$ в некоторой точке θ и

$$\langle x - \theta, \nabla f(x) \rangle \geqslant \mu \|x - \theta\|^2, \qquad \forall x \in \mathbb{R}^r$$

  с некоторой постоянной $\mu > 0$.

  (B) Условие Липшица на градиент функции $f(.)$

$$\|\nabla f(x) - \nabla f(\theta)\| \leqslant A \|x - \theta\|, \qquad \forall x, \theta \in \mathbb{R}^r$$

  с некоторой постоянной $A > \mu$.

  (C) Функция $f(.) \in C^l$ (l-раз непрерывно дифференцируема) и все её частные производные до порядка l включительно удовлетворяют на $\mathbb{R}^r$ условию Гёльдера порядка $\rho, 0 < \rho < 1$:

$$|f(x) - \sum_{|\overline{k}| \leqslant l} \frac{1}{\overline{k}!} \mathcal{D}^{\overline{k}} f(\theta)(x-\theta)^{\overline{k}}| \leq M \|x - \theta\|^{\gamma},$$

  где M — некоторая постоянная, $\gamma = l + \rho \geqslant 2, x \in \mathbb{R}^r$

$$x^{\overline{k}} = x_1^{k_1} \cdots x_r^{k_r}, \qquad \mathcal{D}^{\overline{k}} = \frac{\partial^{|\overline{k}|}}{(\partial x_1)^{k_1} \cdots (\partial x_r)^{k_r}}, \qquad \overline{k} = \begin{pmatrix} k_1 \\ k_2 \\ \vdots \\ k_r \end{pmatrix},$$

$$|\overline{k}| = k_1 + \ldots + k_r, \qquad \overline{k}! = k_1! \cdots k_r!, \qquad k_i \geqslant 0$$

5.1.2 Пробное возмущение и рандомизированные алгоритмы

- Пусть пробное одновременное возмущение $\Delta_n, n = 1, 2, \ldots$ - наблюдаемая последовательность независимых случайных векторов из $\mathbb{R}^r$ с функциями распределения $P_n(.)$, $\{\alpha_n\}$ и $\{\beta_n\}$ - последовательности положительных чисел, стремящиеся к нулю, $\widehat{\theta}^0 \in \mathbb{R}^r$ фиксированный начальный вектор. Для построения последовательностей точек измерения $\{x_n\}$ и оценок $\{\widehat{\theta}^n\}$ предлагаются три алгоритма.

  1. Первый из них использует на каждой итерации одно наблюдение:

$$\begin{cases} x^n = \widehat{\theta}^{n-1} + \beta_n \Delta_n, \qquad y_n = F(w^n, x^n) + v_n, \\ \widehat{\theta}^n = \widehat{\theta}^{n-1} - \dfrac{\alpha_n}{\beta_n} \mathcal{K}^n(\Delta_n) y_n \end{cases} \tag{1}$$

2. а второй и третий по два наблюдения:

$$\begin{cases} x^{2n} = \widehat{\theta}^{n-1} + \beta_n \Delta_n, \qquad x^{2n-1} = \widehat{\theta}^{n-1} - \beta_n \Delta_n, \\ \\ \widehat{\theta}^{n} = \widehat{\theta}^{n-1} - \dfrac{\alpha_n}{2\beta_n} \mathcal{K}^n(\Delta_n)(y_{2n} - y_{2n-1}) \end{cases} \tag{2}$$

$$\begin{cases} x^{2n} = \widehat{\theta}^{n-1} + \beta_n \Delta_n, \qquad x^{2n-1} = \widehat{\theta}^{n-1}, \\ \\ \widehat{\theta}^{n} = \widehat{\theta}^{n-1} - \dfrac{\alpha_n}{2\beta_n} \mathcal{K}^n(\Delta_n)(y_{2n} - y_{2n-1}) \end{cases} \tag{3}$$

$\mathcal{K}^n(.) : \mathbb{R}^r \to \mathbb{R}^r$ с компактным носителем, удовлетворяющие вместе с функциями распределения пробного возмущения $P_n(.)$ условиям

$$\int \mathcal{K}^n(x) P_n(dx) = 0, \qquad \int \mathcal{K}^n(x) x^T P_n(dx) = \mathbb{I}, \qquad \sup_n \int \|\mathcal{K}^n(x)\|^2 P_n(dx) < \infty, n = 1, 2, \ldots \tag{4}$$

5.1.3 Сходимость оценок с вероятностью единица и в среднеквадратичном смысле

- $\mathbb{W} = ssup(P_w(.)) \subset \mathbb{R}^p$ - носитель распределения $P_w(.)$.
- $\mathcal{F}_{n-1}$ - σ-алгебру вероятностных событий, порождаемую случайными величинами $\widehat{\theta}^0, \widehat{\theta}^1, \ldots, \widehat{\theta}^{n-1}$.

$$\overline{v}_n = v_{2n} - v_{2n-1}, \qquad \overline{w}^n = \begin{pmatrix} w^{2n} \\ w^{2n-1} \end{pmatrix}, \qquad d_n = 1$$

Теорема Пусть выполнены условия:

1. (A) для функции $f(x) = E_w\{F(w, x)\}$;
2. (B) для функций $F(w, .), \forall w \in \mathbb{W}$;
3. $\forall x \in \mathbb{R}^r$ функции $F(., x)$ и $\nabla_x F(., x)$ равномерно на $\mathbb{W}$ ограничены;
4. (C) для функций $\mathcal{K}^n(.)$ и $P_n(.), n = 1, 2, \ldots$;
5. $\forall n \geqslant 1, E\{\overline{v}_n^2\} \leqslant \sigma_n^2$, случайные величины $\overline{v}_1, \ldots \overline{v}_n$ и векторы $\overline{w}^1, \ldots, \overline{w}^{n-1}$ не зависят от $\overline{w}^n, \Delta_n$ а случайный вектор $\overline{w}^n$ не зависит от Δ_n.

Если $\sum_n \alpha_n = \infty$ и $\alpha_n \to 0, \qquad \beta_n \to 0, \qquad \alpha_n^2 \beta_n^{-2}(1 + d_n^2 + \sigma_n^2) \to 0$ при $n \to \infty$

тогда $E\{\|\widehat{\theta}^n - \theta\|^2\}$ при $n \to \infty$; т. е. последовательность оценок $\{\widehat{\theta}^n\}$, доставляемых алгоритмом (2) (или (3)), сходится к в θ среднеквадратичном смысле

Если, более того, $\sum_n \alpha_n \beta_n^2 < \infty$ и с вероятностью единица

$$\sum_n \alpha_n^2 \beta_n^{-2}(1 + E\{\overline{v}_n^2 | \mathcal{F}_{n-1}\}) < \infty$$

тогда $\widehat{\theta}^n \to \theta$ при $n \to \infty$ с вероятностью единица

5.1.4 Скорость сходимости оценок

- Пусть

$$\mathcal{K}^n(x) = \mathcal{K}(x) = \begin{pmatrix} \mathcal{K}_1(x) \\ \mathcal{K}_2(x) \\ \vdots \\ \mathcal{K}_r(x) \end{pmatrix}, \qquad x \in \mathbb{R}^r, \qquad n = 1, 2, \ldots$$

$$\mathcal{K}_i(x) = K_0(x_i) \prod_{j \neq i} K_1(x_j), \qquad i, j = 1, \ldots, r, \qquad x \in \mathbb{R}^r \tag{5}$$

где $K_0(.)$ и $K_1(.)$ - некоторые скалярные ограниченные функции (ядра), удовлетворяющие условиям

$$\int uK_0(u)P_\Delta(du) = 1, \qquad \int u^k K_0(u)P_\Delta(du) = 0, \qquad k = 0, 2, \ldots, l,$$
$$\int K_1(u)P_\Delta(du) = 1, \qquad \int u^k K_1(u)P_\Delta(du) = 0, \qquad k = 1, , \ldots, l-1, \tag{6}$$

Теорема Пусть в алгоритме (2) (или (3)) вектор-функции $\mathcal{K}^n(.)$ определяются по формулам (5) и

$$\alpha_n = \alpha n^{-1}, \qquad \beta_n = \beta n^{-\frac{1}{2\gamma}}, \qquad \beta > 0, \qquad n = 1, 2, \ldots$$

Если выполнены условия:

1. (A, C) для функции $f(x) = E_w\{F(w, x)\}$ при $\gamma \geqslant 2, \qquad \alpha > \dfrac{\gamma - 1}{2\mu\gamma}$;
2. (B) для функций $F(w, .) \qquad \forall w \in \mathbb{W}$;
3. $\forall x \in \mathbb{R}^r$ функции $F(., x)$ и $\nabla_x F(., x)$ равномерно на $\mathbb{W}$ ограничены;
4. (6) для функций $K_0(.), K_1(.)$ и $P_\Delta(.)$;
5. $d_n n^{-1+\frac{1}{2\gamma}} \to 0$ при $n \to \infty$;
6. $\forall n \geqslant 1$ случайные векторы $\overline{w}^n, \Delta_n$ не зависят от $\overline{v}_1, \ldots, \overline{v}_n, \overline{w}^1, \ldots, \overline{w}^{n-1}$ и случайный вектор Δ_n не зависит от $\overline{w}^n$;
7. $E\{(v_{2n} - v_{2n-1})^2/2\} \leqslant \sigma_2^2, \qquad (E\{v_n^2\} \leqslant \sigma_1^2)$

тогда для среднеквадратичной скорости сходимости последовательности оценок $\{\widehat{\theta}^n\}$, сгенерированных алгоритмом (2) (или (3)), асимптотически при $n \to \infty$ выполняется

$$E\{\|\widehat{\theta}^n - \theta\|^2\} = \mathcal{O}(n^{-\frac{\gamma-1}{\gamma}})$$

5.1.5 Пошаговое выполнение алгоритма

Шаг 1: Инициализация и выбор коэффициентов :

- Выберите $\widehat{\theta}^0$ и значения для неотрицательных коэффициентов $\alpha, \beta, \gamma, \delta$ и ν.
- Выбор последовательностей $\{\alpha_n\}$ и $\{\beta_n\}$: $\alpha_n = \alpha/(\delta + n)^\nu, \qquad \beta_n = \beta/n^\nu$

Шаг 2: Генерация вектора пробного возмущения :

- Сгенерируйте r-мерный случайный вектор возмущения Δ_n, в котором все r компонент независимо смоделированны по вероятностному распределению Бернулли ± 1 с вероятностью $1/2$ для каждого результата.

Шаг 3: Измерение значений функции :

- Получите два измерения функции $f(.)$ в точках $\widehat{\theta}^{n-1}$ и возмущенной относительно текущей оценки $\widehat{\theta}^{n-1} + \beta n^{-\gamma}\Delta_n$: $y(\widehat{\theta}^{n-1})$ и $y(\widehat{\theta}^{n-1} + \beta n^{-\gamma}\Delta_n)$

Шаг 4: Аппроксимация градиента :

- Сгенерируйте компоненты вектора аппроксимации градиента по правилу

$$\widehat{g}_i^n(\widehat{\theta}^{n-1}) = n^\gamma \Delta_{ni} \frac{y(\widehat{\theta}^{n-1} + \beta n^{-\gamma}\Delta_n) - y(\widehat{\theta}^{n-1})}{\beta}, i = 1, 2, \ldots, r$$

  где Δ_{ni} - i-я компонента вектора Δ_n.

Шаг 5: Обновление оценки θ:

$$\widehat{\theta}^n = \widehat{\theta}^{n-1} - \frac{\alpha}{(\beta + n)^\nu}\widehat{g}^n(\widehat{\theta}^{n-1})$$

Шаг 6: Итерация или завершение :

- Возвратитесь к шагу 2 с заменой n на $n + 1$.

6 SPSA для решения задачи о сервере

Выберем некоторые достаточно малые коэффициенты $\alpha > 0$ и $\beta > 0$ и случайную последовательность чисел $\{\Delta_n\}$, равных ± 1 с одинаковой вероятностью. Обозначим $\mathcal{P}_{[a,b]}(.)$ - проектор в интер - $[a, b]$.

6.1 Алгоритм SPSA с одним измерением:

1. Положим $k = 0$ и выберем некоторое начальное значение оценки $\widehat{\theta}_0$.
2. В начале каждого k-го такта вычисляем $\theta_k^{'} = \mathcal{P}_{[a,b]}(\widehat{\theta}_k + \beta\Delta_k)$.
3. Запускаем сервер с значением параметра $x = \theta_k^{'}$.
4. После завершения k-го такта подсчитаем новую оценку по правилу
$$\widehat{\theta}_{k+1} = \mathcal{P}_{[a,b]}(\widehat{\theta}_k - \frac{\alpha}{\beta}\Delta_k y_k).$$
5. Увеличиваем номер такта $k = k + 1$.
6. Переход к п.2 (повтор действий заново).

6.2 Алгоритм SPSA с двумя измерениями

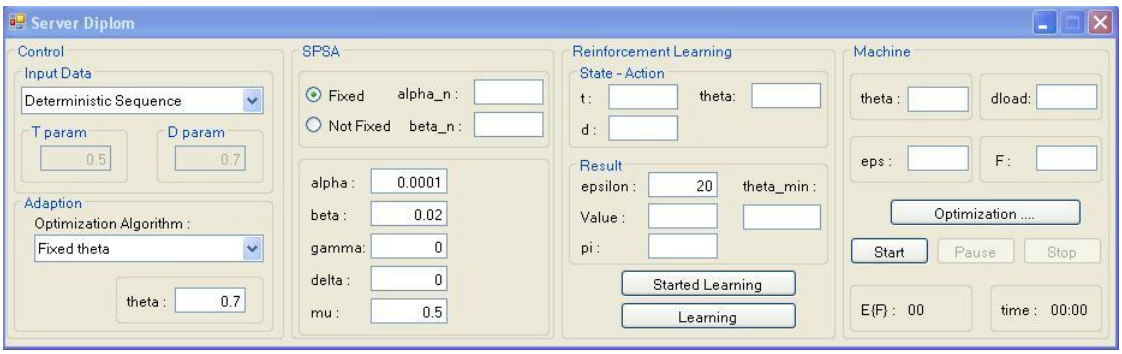
Алгоритм SPSA с двумя измерениями отличается от предыдущего алгоритма только в двух операциях

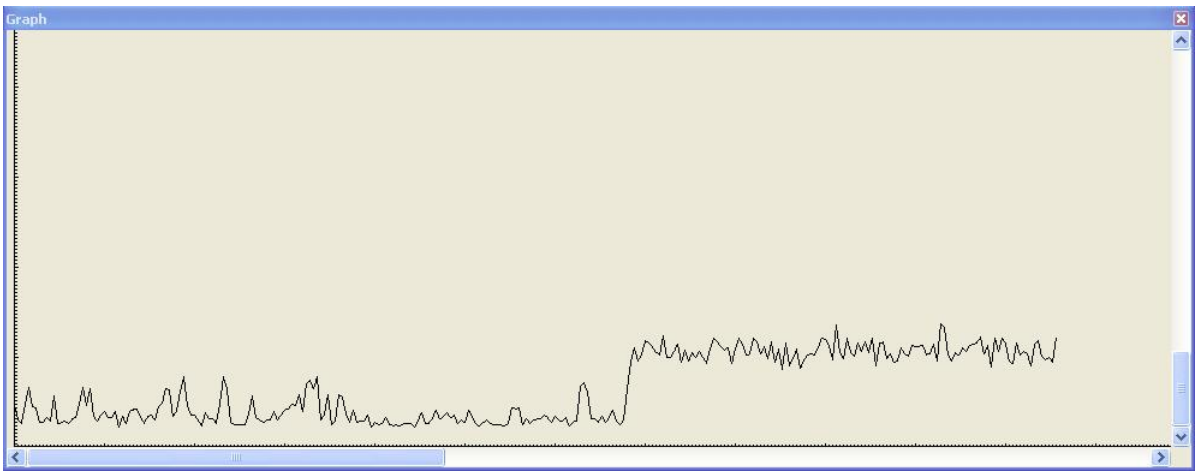
2 В начале каждого такта k вычисляем значение $\theta_k^{'}$, но в этот раз оно зависит от четности счетчика: если k четное, то $\theta_k^{'} = \widehat{\theta}_k$, иначе $\theta_k^{'} = \mathcal{P}_{[a,b]}(\widehat{\theta}_k + \beta\Delta_k)$.

4 После завершения k-го такта подсчитаем новую оценку по правилу: если k четное, то $\widehat{\theta}_{k+1} = \widehat{\theta}_k$, иначе
$$\widehat{\theta}_{k+1} = \widehat{\theta}_k - \frac{\alpha}{\beta}\Delta_k(y_k - y_{k-1}).$$

7 Моделирование

- Интерфейс моей программы состоит из двух основных полей. Одно из них используется для ввода входных данных, а в другом отображается график в процессе работы программы.

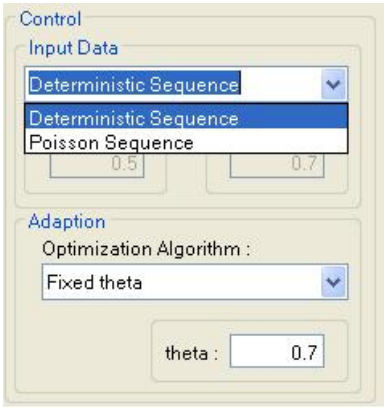




- Поле для входных данных включает в себя четыре логических части:

7.1 Control

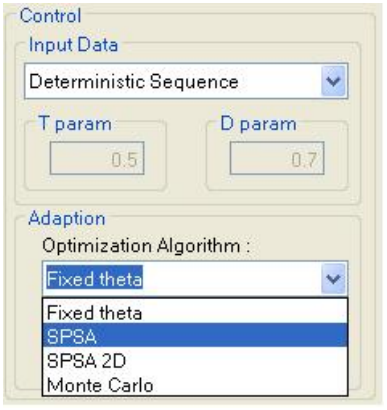
– Здесь в поле Input Data можно выбрать способ ввода данных.



∗ Если мы выбираем Deterministic Sequence то данные будут считываться из файла input.txt.

∗ Если же мы хотим создать новый файл то следует выбрать Poisson Sequence. В этом случае станут доступны два параметра T и D.

– В поле Optimization Algorithm можно выбрать один из трех алгоритмов:

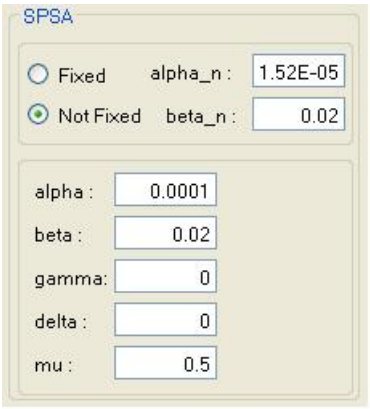


1. Fixed theta.
2. SPSA.
3. Monte Carlo.

– Также можно выбрать значение θ.

7.2 SPSA

– Здесь находятся параметры для настройки алгоритма SPSA.

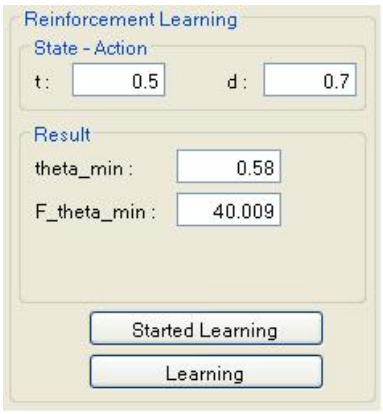


– Можно сказать программе должны ли быть значения α_n и β_n константами или они должны изменяться по формуле, параметры которой также можно задать в оставшихся полях.

– Также здесь можно в любой момент работы программы посмотреть текущие значения α_n и β_n.

7.3 Reinforcement Learning

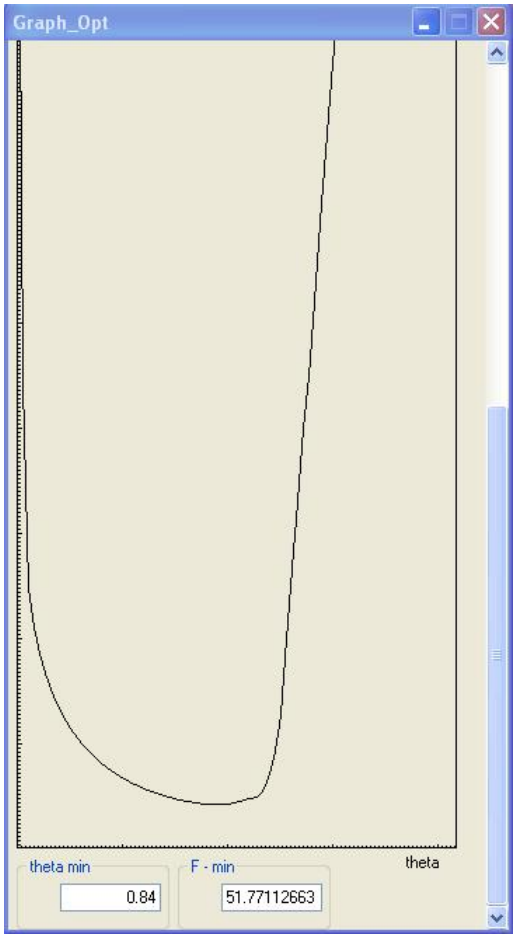
– Когда мы вибираем алгоритм Monte Carlo и запускаем программу в группе Reinforcement Learning можно наблюдать значения t и d параметров а также минимальное значение θ - θ_{min} и значение функции при таком θ.



– Если кликнуть кнопку Start Learning то программа создаст новый файл Learning.txt. При нажатии на кнопку Learning программа будет обрабатывать данные сто раз подряд используя и изменяя данные файла при каждом новом проходе.

7.4 Machine

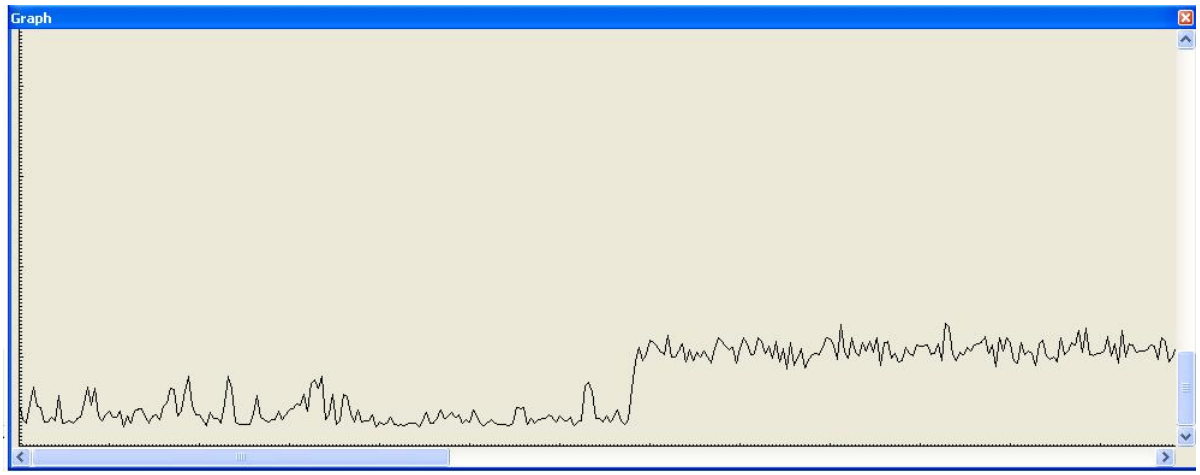
– Здесь можно во время выполнения программы наблюдать значения θ, dload, eps и F (значение функции).
– Также здесь расположены кнопки запуска, паузы и остановки программы.
– После завершения работы программы можно будет увидеть значение $E\{F\}$(среднее значение функции)
– При нажатии на кнопку Optimization будет рассчитано средние значения функции для всех параметров θ от 0.1 до 1.5, построен график зависимости $E\{F\}$ от θ и показано минимальное значение функции и θ при котором оно достигается.



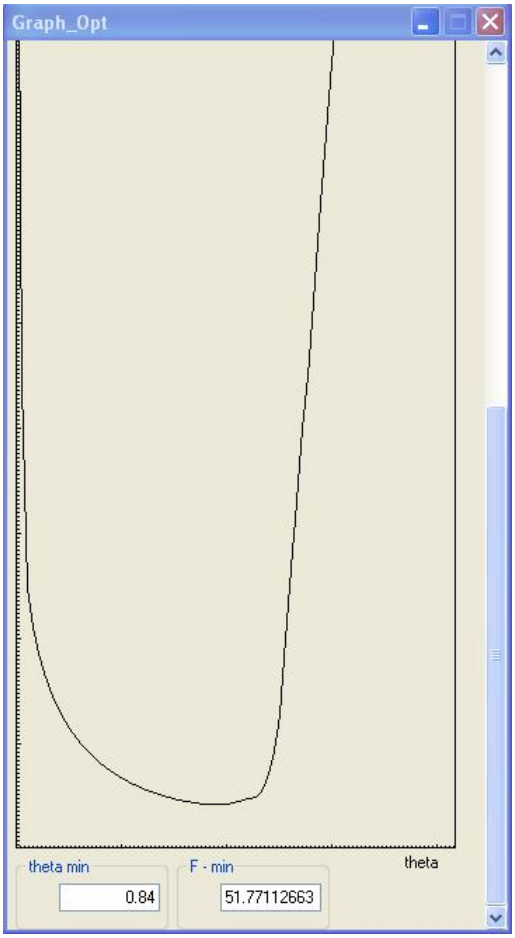
8 Результаты

8.1 Результаты программы с фиксированным значением θ

- Проанализируем результат. Если выбрать режим Deterministic Sequence и фиксированное значение θ равное 0.7 то из полученного графика видно, что первую половину задач программа обрабатывает достаточно хорошо, но потом заметен резкий скачок, и во второй половине сервер работает хуже.

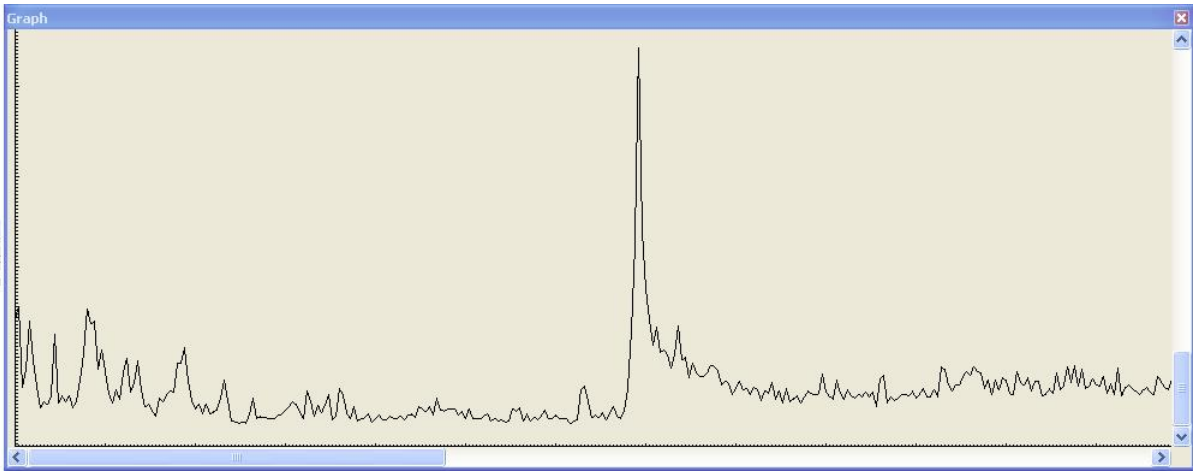


- Теперь запустим optimization при фиксированом файле input.txt. После завершения процесса видно что минимум функции = 51 достигается при значении $\theta = 0.84$.

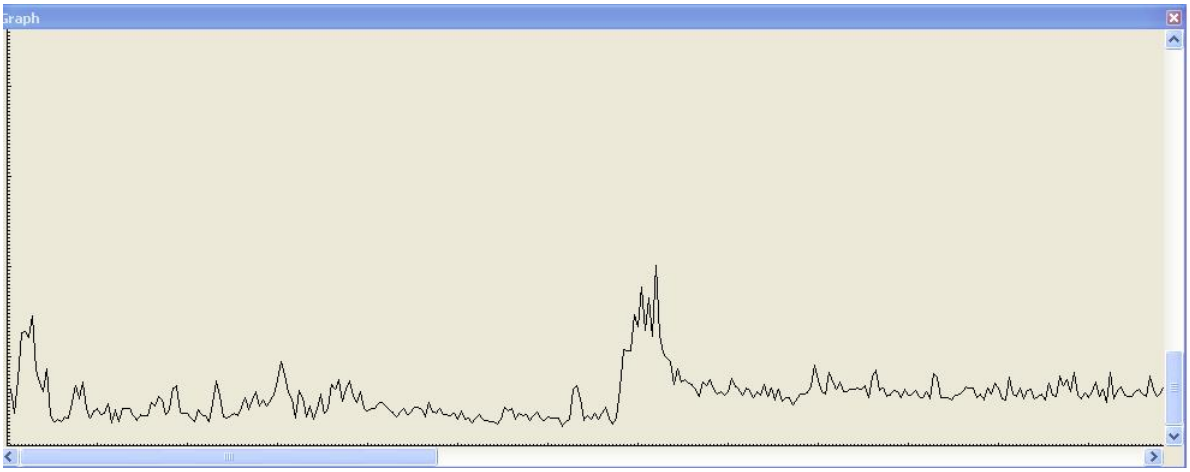


8.2 Результаты программы с алгоритмом SPSA

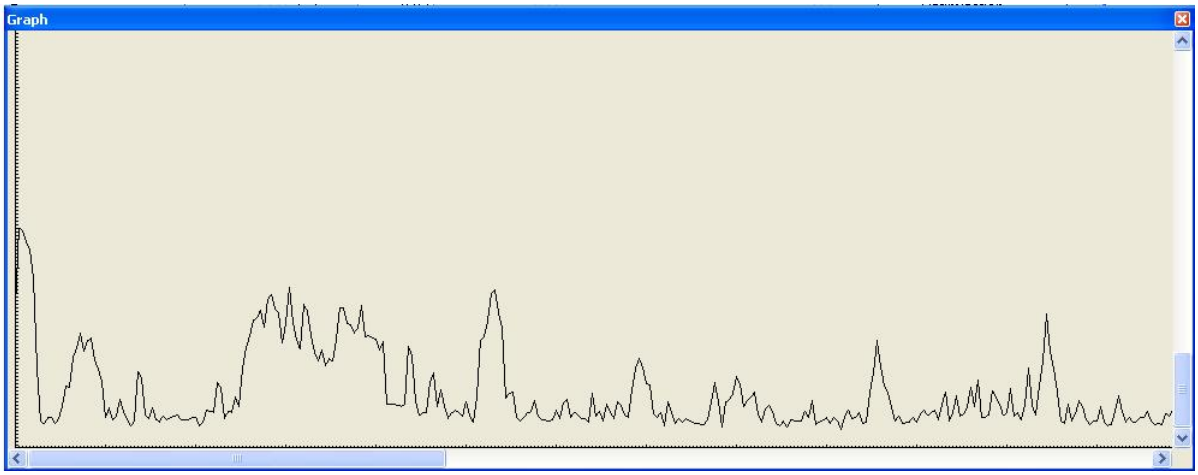
- Попробуем выбрать режим Deterministic Sequence и применим алгоритм SPSA с начальным значением $\theta = 0.12$. Тогда на первых секундах сервер работает плохо, но вскоре оптимизируется и значение θ стремится к 0.6.
  Во второй половине снова наблюдаем резкий скачок который также продолжается недолго. Здесь θ стремится к 1.1.



- Выберем в при неихзменных настройках начальное значение $\theta = 1$. Видим ту же картинку. Только теперь в первой половине θ стремится к 0.7, а во второй к 1.

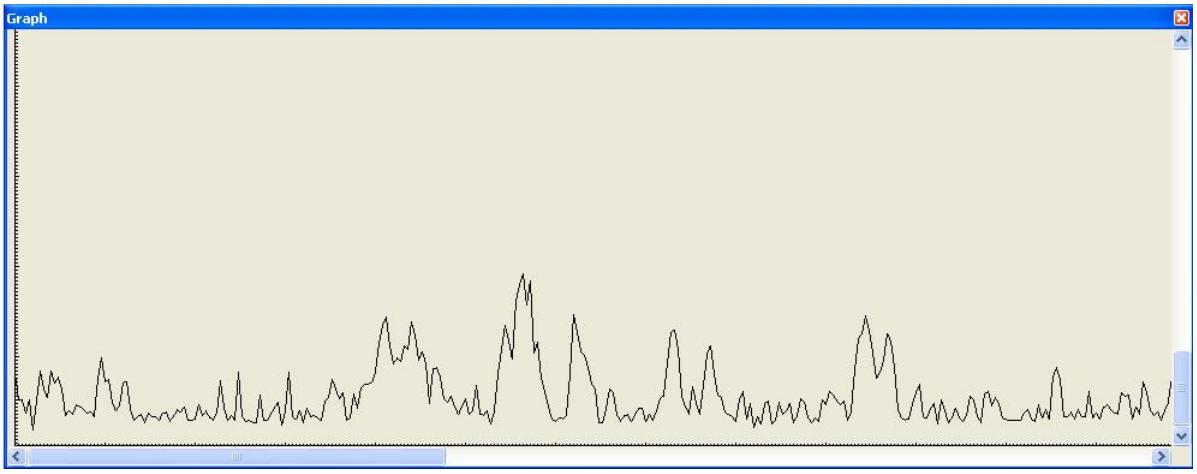


- Посмотрим как работает SPSA при выборе Poisson Sequence, значении $t = 0.5$, $d = 0.7$ и начальным $\theta_0 = 0.12$. Видим вначале знакомую "гору", но потом θ стремится к 0.5 и все выравнивается.



8.3 Результаты программы с алгоритмом Монте-Карло

- Теперь посмотрим на график результата работы алгоритма Монте Карло при выборе Poisson Sequence и значениях t и d параметров 0.5 и 0.7 соответственно.



9 Заключение

- В своей дипломной работе я рассмотрел две теории Рандомизированные алгоритмы стохастической аппроксимации и Обучение с подкреплением для решения задачи эффективного обслуживания сервером очереди заданий. Программа которую я представил в работе написана на языке $C^{\#}$.
- Работая с алгоритмом SPSA я попытался построить алгоритм с изменением значения α_k. Программа стала работать хорошо. Значение функции сходится к минимальному довольно быстро.
- Когда я работал с методом Монте - Карло я не полностью описал идею обучения с подкреплением. В моей программе "состояние"остается неизменным. И хотя я несколько упростил алгоритм он все равно работает замечательно. В итоге я получил среднее значение функции близкое к тому которое дает алгоритм SPSA.
- В дальнейшем я хочу применить все идеи обучения с подкреплением и получить возможность работать с изменяющимся состоянием. Тогда программа должна работать лучше т.к. не будет больших потерь времени при переходе к новому состоянию.

10 Список Литературы

1. *Гранчин О.Н., Поляк Б.Т.* Рандомизированные алгоритмы оценивания и оптимизации при почти произвольных помехах. М.: Наука, 2003. 291 с.

2. *Я. В. Волкович, О. Н. Граничин* Адаптивная оптимизация сервера, обрабатывающего очередь заданий.

3. *Richard S. Sutton and Andrew G. Barto* Reinforcement Learning: An Introduction. MIT Press, Cambridge, MA, 1998 A Bradford Book

4. *Волкович Яна В.* 2004 «Моделирование работы и адаптивная оптимизация сервера, обрабатывающего очередь заданий»