Санкт-Петербургский национальный исследовательский университет
информационных технологий, механики и оптики

Факультет информационных технологий и программирования
Кафедра компьютерных технологий

Маркина Маргарита Анатольевна

Разработка гибридного алгоритма недоминирующей сортировки

Научный руководитель: к.т.н. доцент кафедры КТ М. В. Буздалов

Санкт-Петербург
2016

Содержание

Введение **5**

Глава 1. Обзор работы **7**

1.1 Недоминирующая сортировка 7
1.1.1 Определение 7
1.1.2 Применение и актуальность 8
1.2 Анализ существующих алгоритмов 8
1.2.1 Наивные алгоритмы 9
1.2.2 Алгоритмы ¨Разделяй и властвуй 9
1.2.3 Алгоритм Роя и др 10
1.2.4 Имеющиеся результаты 10
1.3 Постановка задачи 10

Глава 2. Теоретические исследования **11**

2.1 Основные кандидаты для гибридизации 11
2.1.1 Алгоритм на основе метода разделяй и властвуй . . . 11
2.1.2 Алгоритм Best Order Sort 12
2.2 Идеи гибридизации 12
2.3 Анализ работы кандидатов 13
2.4 Гибридный алгоритм 13

Глава 3. Практические исследования **14**

3.1 Виды входных данных 14
3.2 Детали гибридизации 14
3.3 Формат экспериментов 15
3.4 Результаты экспериментов 15
3.4.1 Случайные точки в гиперкубе 15
3.4.1.1 Замеры 16
3.4.1.2 Левая и правая границы 16

3.4.1.3 Выводы 16
3.4.2 Точки одного ранга 17
3.4.2.1 Замеры 17
3.4.2.2 Выводы 17
3.5 Дальнейшие действия 17
3.6 Выводы 18

Заключение **19**

Список литературы **20**

Введение

Множество известных и широко распространенных многокритериальных эволюционных алгоритмов используют процедуру недоминирующей сортировки, или процедуру определения множества недоминирующих решений, которая может быть сведена к недоминирующей сортировке. Примерами таких алгоритмов могут послужить NSGA-II [1], PESA [2], PESA-II [3], SPEA2 [4], PAES [5], PDE [6] и многие другие алгоритмы. Вычислительная сложность одной итерации этих алгоритмов часто определяется сложностью процедуры недоминирующей сортировки, следовательно, снижение сложности последней делает такие многокритериальные эволюционные алгоритмы значительно быстрее.

Существуют разные алгоритмы недоминирующей сортировки, но эффективность их работы очень сильно отличается в зависимости от данных. Этим можно воспользоваться и совместить идеи разных алгоритмов в одном, чтобы получить новый алгоритм, сочетающий в себе преимущества существующих, избавившись при этом от их недостатков.

Цель данной работы – сделать гибридный алгоритм недоминирующей сортировки, который будет использовать наиболее подходящий алгоритм или переключаться между алгоритмами в ходе своей работы.

В Главе 1 данной работы представлен общий обзор работы. В разделе 1.1 подробно рассмотрены определение недоминирующей сортировки и необходимые для этого понятия, а также представлены примеры применения недоминирующей сортировки, подтверждающие актуальность данной работы. В разделе 1.2 произведен обзор имеющихся результатов и подробно описаны лучшие из них. В разделе 1.3 описаны недостатки существующих алгоритмов. В разделе 1.4 сформулирована постановка задачи.

В Главе 2 представлены теоретические исследования по гибридизации алгоритмов недоминирующей сортировки. В разделе 2.1 произведен

анализ существующих алгоритмов и их сравнение. В разделе 2.2 показаны основные проблемы, возникающие при гибридизации алгоритмов, а также предложены пути их решения.

В Главе 3 представлены практические исследования и их результаты. В разделе 3.1 представлена общая архитектура программы и использованные оптимизации. В разделе 3.2 приведены данные экспериментов по сравнению скорости работы гибридного алгоритма относительно старых. В разделе 3.3 показано улучшение производительности практической задачи, использующей разработанный алгоритм для недоминирующей сортировки.

В заключении подведены итоги работы, а также сказано, какие могут быть дальнейшие пути развития гибридных алгоритмов недоминирующей сортировки.

Глава 1. Обзор работы

В этой главе представлен общий обзор работы: уточнены цели и объяснены термины и понятия, присутствующие в решении задачи. Также произведен обзор имеющихся результатов и сформулирована постановка задачи.

1.1. Недоминирующая сортировка

В данном разделе представлено определение недоминирующей сортировки и необходимые для ее понимания понятия. Также рассмотрены случаи применения недоминирующей сортировки, которые обосновывают актуальность нового ускоренного алгоритма недоминирующей сортировки.

1.1.1. Определение

Недоминирующая сортировка – это процедура, которая ранжирует множество точек в многомерном пространсве R^n. Если описывать нефор-мально, то ее задача определить, какие точки ¨учше¨ругих. При этом до-пускается, что две точки могут быть одинаково ¨орошими:

Для того, чтобы сформулировать определение недоменирующей сортировки, сначала надо определить, какие точки мы считаем ¨уч-ше¨ругих. Для этого введем определение доминирования одной точки дру-гой.

Определение. В M-мерном пространстве, точка $A = (a_1, ..., a_M)$ до-минирует точку $B = (b_1, ..., b_M)$ тогда и только тогда, когда для всех $1 \leq i \leq M$ выполняется неравенство $a_i \leq b_i$, и существует такое j, что $a_j < b_j$.

¨учшими¨ данном контексте будут считаться точки, которые не до-минируются ни одной другой точкой или, другми словами, лежащие на Парето-фронте. Однако часто бывают не только точки с парето-фронта,

но и другие ¨орошие¨очки. Таким образом мы приходим к определению процедуры недоминирующей сортировки.

Определение. Недоминирующая сортировка множества точек S в M-мерном пространстве — это процедура, которая назначает всем точкам из S ранг. Все точки, которые не доминируются ни одной точкой из S имеют ранг ноль. Точка имеет ранг $i+1$, если максимальный ранг среди доминирующих её точек равен i.

1.1.2. Применение и актуальность

Многокритериальные эволюционные алгоритмы работают с кандидатами решений, или особями. Каждый индивид оценивается функцией приспособленности $q = f(p)$, которая считается для каждой особи p и является вектором из M значений.

Рассмотрим пример задачи, которая используют недомининирующую сортировку: $(\mu + \lambda)$ - эволюционная стратегия. Пусть есть начальное множество особей S, причем $|S| = \mu$. Далее по этому множеству генерируется новое множество особей P, где $|P| = \lambda$. Следующим шагом необходимо выбрать результирующее множество R из $S \cup P$, где $|S| + |P| \leq (\mu + \lambda)$, чтобы $|R| = \mu$. Требуется отобрать лучшие особи. Для этого необходимо произвести недоминирующую сортировку, тогда лучшими будут те, которые имеют наименьший ранг.

Таким образом, ускорив недомиминирующую сортировку, мы получит ускорение многокритериальных эвалючионных алгоритмов, которые ее используют.

1.2. Анализ существующих алгоритмов

В данном разделе будут рассмотрена история развития алгоритмов недоминирующей сортировки. И особое внивание будет уделено алгоритмам, которые применяются в гибридизации в данной работе.

1.2.1. Наивные алгоритмы

В работе Кунга и др. [7] предлагается алгоритм определения множества недоминирующих точек, при этом его вычислительная сложность составляет $O(Nlog^{M-1}N)$, где N — это число точек, а M — размерность пространства. Этот алгоритм возможно использовать для выполнения недоминирующей сортировки. Сначала в множестве S находятся множество точек с рангом 0. Затем алгоритм Кунга запускается на оставшемся множестве точек, и получившемуся множеству точек присваивается ранг 1. Процесс выполняется до тех пор, пока имеются точки, которым не присвоен ранг. Описанная процедура в худшем случае выполняется за $O(N^2log^{M-1}N)$, если максимальный ранг точки равен $O(N)$.

1.2.2. Алгоритмы ¨азделяй и властвуй

Йенсен [8] впервые предложил алгоритм недоминирующей сортировки с вычислительной сложностью $O(Nlog^{M-1}N)$. Однако, как корректность, так и оценка сложности алгоритма доказывалась в предположении, что никакие две точки не имеют совпадающие значения ни в какой размерности. Устранить указанный недостаток оказалось достаточно трудной задачей — первой успешной попыткой сделать это, насколько известно исполнителю данной НИР, является работа Фортена и др. [9]. Исправленный (или, согласно работе, «обобщенный») алгоритм корректно работает во всех случаях, и во многих случаях его время работы составляет $O(Nlog^{M-1}N)$, но единственная оценка времени работы для худшего случая, доказанная в работе [8], равна $O(N^2M)$. Наконец, в работе Буздалова и др. [10] предложены модификации алгоритма из работы [8], которые позволили доказать в худшем случае также и оценку $O(Nlog^{M-1}N)$, не нарушая корректности работы алгоритма.

1.2.3. Алгоритм Роя и др

Большой интерес представляет алгоритм Роя *Best Order Sort* (*BOS*) [**Roy**], который в отличии вышеупомянутых не использует метод разделяй и властвуй. Его вычислительная сложность $O(MNlogM+MN^2)$. В лучшем случае алгоритм работает за $O(MNlogM)$, что лучше алгоритма предложенного Буздаловым и др. Однако в худшем случае его асимптотика совсем другая - $O(MN^2)$. Авторами алгоритма не было проведено более точных исследований по его времени работы.

1.2.4. Имеющиеся результаты

1.3. Постановка задачи

Задача содержит несколько пунктов:

- Выбрать наиболее подходящие для гибридизации алгоритмы.
- Основываясь на практических экспериментах на разных видах входных данных, оценить время обработки каждым выбранным алгоритмом.
- Выдвинуть предположение о том, как и в какой момент менять стратегию сортировки.
- Проверить предположение
- Сформулировать и реализовать гибридный алгоритм.

Глава 2. Теоретические исследования

В данной главе будут представлены основные результаты работы. Сначала будет рассмотрен основные кандидаты для гибридизации. Затем будет представлен анализ работы кандидатов на разных входных данных. Потом будет сделано некоторое предположение на основании экспериментов и на его основе сформулирован алгоритм гибридизации.

2.1. Основные кандидаты для гибридизации

2.1.1. Алгоритм на основе метода разделяй и властвуй

Далее в работе этот алгоритм, для краткости, будет называться Fast.

Основная идея данного алгоритма аналогична идее алгоритма сортировки массива QuickSort. Множество точек разделяется медианой, и все точки разделяются на три множества: меньшие, равные и большие медианы по какому-либо критерию, причем можно сразу сказать, что точки первого множества доминируют над точками второго, которые в свою очередь доминируют над точками третьего. Далее алгоритм запускается рекурсивно на этих множествах.

Для понимания сути алгоритма Буздалова и др. [10] опишем основные моменты. Самый большой для нас интерес представляет процедура $NDHelperA$.

$NDHelperA$ принимает множество точек S и номер рассматриваемого критерия k. Далее, как написано выше, разделяем на 3 множества L, M, H, основываясь только на k критерии. L и H запускам рекурсивно, при этом продолжаем рассматривать k критерий. А M запускаем с номером критерием k-1.

Когда количество точек равно двум или количество критериев, по которым мы выявляем ранг, равно 2, рекурсивный запуск не происходит. Вместо этого запускается алгоритм сканирующей прямой, который подроб-

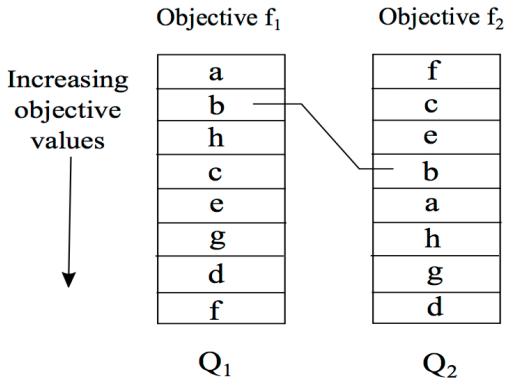


Рис. 2.1: На рисунке представлены отсортированные списки по критерию f1 и f2. Точка b будет сравниваться только с точкой a и в последствии ее ранг не будет меняться.

но описан еще в статье Фортина [9].

Таким образом по завершению процесса все точки проранжированы.

2.1.2. Алгоритм Best Order Sort

Далее в работе этот алгоритм, для краткости, будет называться BOS.

Данный алгоритм создает отсортированный списки точек соответствующие каждому критерию (см. 2.1). Если у точек критерии совпадают, используется порядок, основанный на предыдущих критериях. Далее просматриваем точки начиная с первых в этих списках, переходя от списка к списку, потом переходим ко вторым точкам в этих списках. Ранг назначаются точкам при первой встречи, если точка уже имеет ранг, она пропускается.

Более детальное описание алгоритма не так существенно, так как алгоритм BOS встраивается в алгоритм Fast, и его внутреннее устройство не очень интересно.

2.2. Идеи гибридизации

Простейший способ гибридизации двух вышеупомянутых алгоритмов – создание алгоритма, который будет по входным данным выбирать один из двух алгоритмов и запускать его. Он должен принимать решение

выбора алгоритма не медленнее, чем время работы выбранного им алгоритма.

Однако мы можем воспользоваться тем, что алгоритм л

Следовательно, наша основная задача заключается в том, чтобы быстро на каждом этапе выбирать наилучший алгоритм.

2.3. Анализ работы кандидатов

Эти два алгоритма были выбраны в качестве основных кандидатов на гибридизацию именно из-за их асимптотики. Как указано в [10], алгоритм Fast работает за $O(N \log^{M-1} N)$. Алгоритм BOS работает в лучшем случае за $\Theta(MN \log M)$, а в худшем случае – за $\Theta(MN^2)$ [11]. Можно видеть, что лучший и худший случаи у алгоритма BOS очень разнятся: при фиксированном числе критериев M BOS работает в лучшем для себя случае асимптотически лучше, чем Fast, однако он проигрывает, если попадает в худший для себя случай.

2.4. Гибридный алгоритм

В данной секции я опишу свой гибридный алгоритм, когда соберу больше данных по работе кандидатов

Глава 3. Практические исследования

В данной главе рассмотрены экспериментальные результаты, полученные в течение работы.

Наша основная задача, как было описано в главе посвященной теоретическим исследованиям, заключается в том, чтобы быстро выбирать наилучший алгоритм, основываясь на входных данных. Для этого необходимо сначала собрать довольно много данных по времени их работы на разных входных данных.

3.1. Виды входных данных

Рассмотрим какие именно входные данные будем рассматривать.

- Случайный точки в гиперкубе.
- Точки на одном ранге.
- Точки часто имеею одинаковый критерии.
- Каждая точка имеет отличный от других ранг.
- Крайние случаи с шумами.

3.2. Детали гибридизации

Примитивный избиратель алгоритмов, который принимает решение в самом начале, основываясь на всех данных в целом.

Следующая идея сбора информации: сделаем дампы в момент каждого рекурсивного вызовов алгоритма Fast и сравним время сортировки этих подмножеств алгоритмами.

3.3. Формат экспериментов

Будем засекать время на каждом интересующем нас множестве данных. Если время незначительно, запускаем алгоритм на одних и тех же данных 2^i раз, где i - количество пренебрежимо маленьких замеров времени.

Обозначим T_{Fast} - как время за которое алгоритм Fast отсортировал экспериментальное множество точек S, T_{BOS} - время за которое справился алгоритм BOS. $T_{\max} = max(T_{\text{BOS}}, T_{\text{Fast}})$

Оценивать будем с помощью графика, где по абсцисе будет мощность множества S для которого проводился эксперимент. По ординате будет $\frac{T_{\text{BOS}} - T_{\text{Fast}}}{T_{\max}}$.

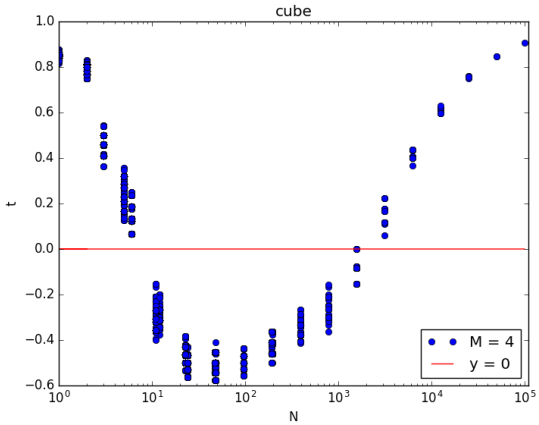
3.4. Результаты экспериментов

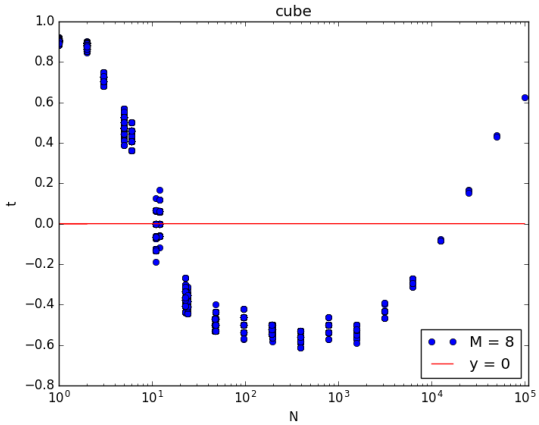
На данном этапе работы были исследованы случайные точки в гиперкуб и точки на одном ранге.

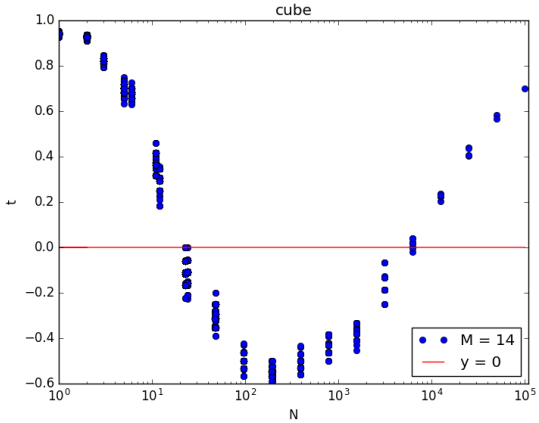
3.4.1. Случайные точки в гиперкубе

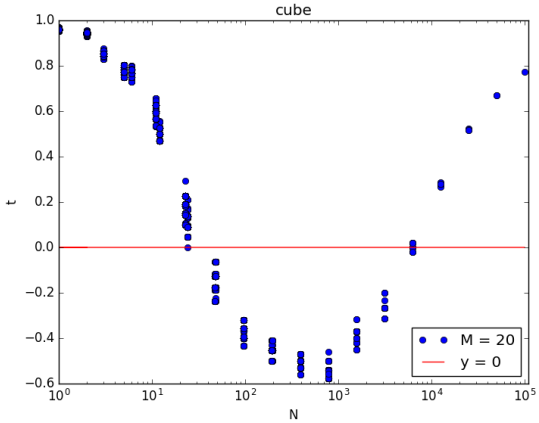
Эксперименты были запущены для N = 100000 и M = 4, 6, 8, 10, 12, 14, 16, 18, 20

3.4.1.1. Замеры

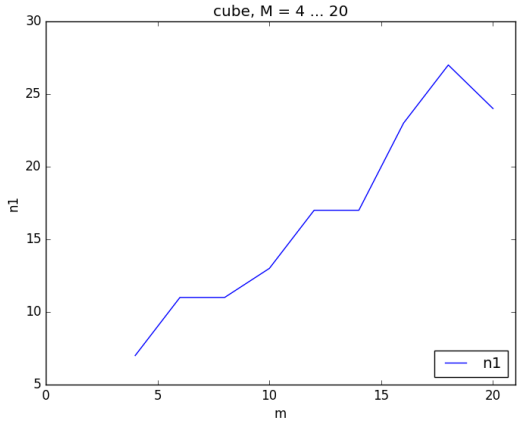


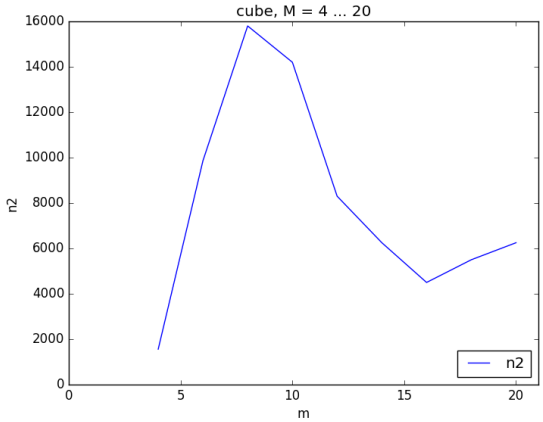






3.4.1.2. Левая и правая границы





3.4.1.3. Выводы

Левая граница похожа на линейную с коэффициентом 4/3

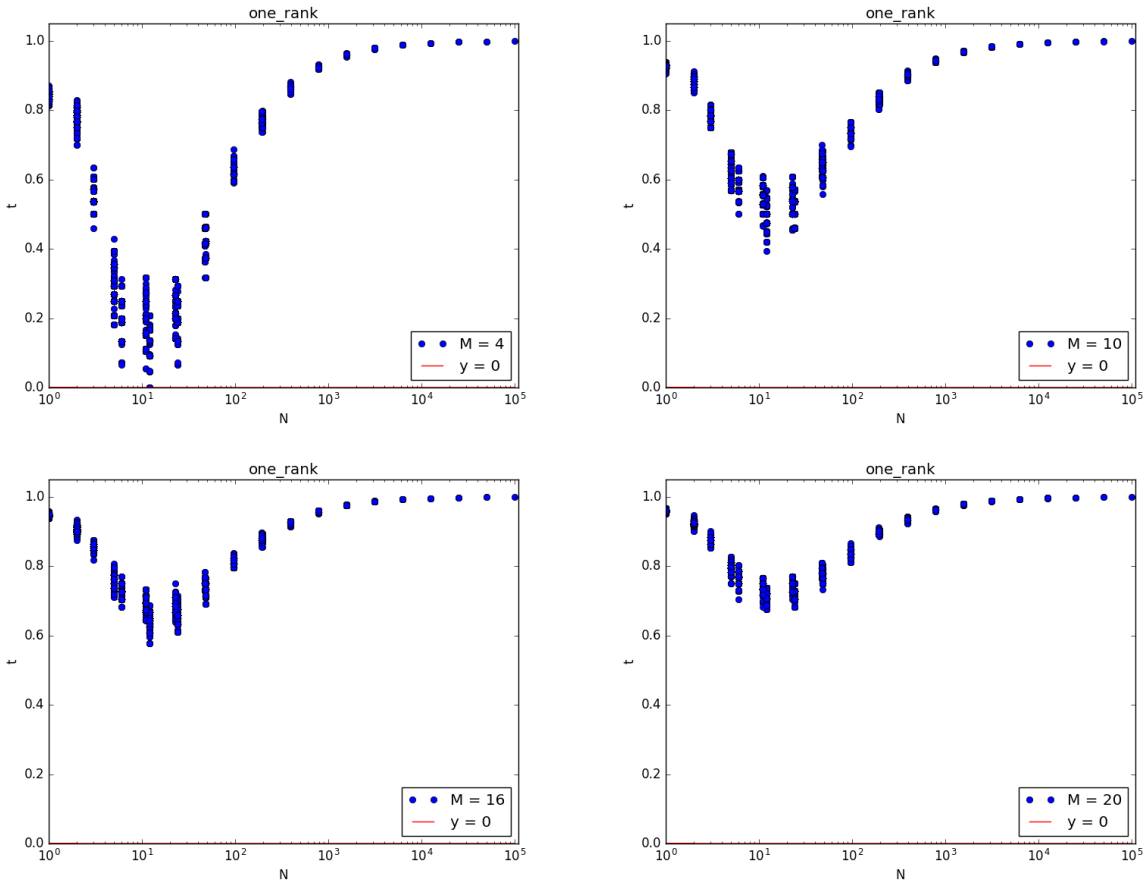
Правая граница ведет себя непонятным образом с максимумом око-

ло 8.

3.4.2. Точки одного ранга

Эксперименты были запущены для N = 100000 и M = 4, 8, 10, 14, 16, 18, 20

3.4.2.1. Замеры



3.4.2.2. Выводы

BOS работает всегда хуже.

3.5. Дальнейшие действия

Запустить на других N и M, чтобы проверить гипотезы.

Исследовать поведение в крайних случаях с шумами. Например, взять однофронтовый набор точке и добавить шумы, при этом сколько

конкретно шума - параметр, который надо будет подвигать.

3.6. Выводы

- Соотношение времени - функция выпуклая вниз.
- Чем больше разбросаны точки, тем лучше работает BOS
- Чем меньше фронтов, тем быстрее работают оба алгоритма

  у BOS это в рамках константы

  У Fast логарифмическое ускорение

Важная идея заключается в том, что с уменьшением числа фронтов функция соотношений времени работы поднимается и смещается влево. И если научится предсказывать примерное число фронтов, то можно понять, в какой момент переключать алгоритм.

Заключение

Результатом данной работы будет гибридный алгоритм.

Список литературы

1. *Deb K.* A Fast Elitist Multi-Objective Genetic Algorithm: NSGA-II // Transactions on Evolutionary Computation. 2000. С. 182—197.
2. *Corne D.* The Pareto Envelope-based Selection Algorithm for Multiobjective Optimization // Parallel Problem Solving from Nature Parallel Problem Solving from Nature VI. 2000. С. 839—848.
3. *Corne D.* PESA-II: Region-based Selection in Evolutionary Multiobjective Optimization // Proceedings of Genetic and Evolutionary Com- putation Conference. 2001. С. 283—290.
4. *Zitzler E.* SPEA2: Improving the Strength Pareto Evolutionary Algorithm for Multiobjective Optimization // Proceedings of the EUROGEN'2001 Conference. 2001. С. 91—100.
5. *Knowles J.* Approximating the Nondominated Front Using the Pareto Archived Evolution Strategy // Evolutionary Computation. 2000. С. 149—172.
6. *Abbass H.* PDE: A Pareto Frontier Differential Evolution Approach for Multiobjective Optimization Problems // Proceed- ings of the Congress on Evolutionary Computation. 2001. С. 971—978.
7. *Kung H.* On Finding the Maxima of a Set of Vectors // Journal of ACM. 1975. С. 469—476.
8. *Jensen M.* Reducing the Run-time Complexity of Multiobjective EAs: The NSGA-II and Other Algorithms // Transactions on Evolutionary Computation. 2003. С. 503—515.
9. *Fortin F.* Generalizing the Improved Run-time Complexity Algorithm for Non-dominated Sorting // Proceeding of Genetic and Evolutionary Computation Conference. 2013. С. 615—622.
10. *Buzdalov M.* A Provably Asymptotically Fast Version of the Gen- eralized Jensen Algorithm for Non-Dominated Sorting // International Con- ference on Parallel Problem Solving from Nature. 2014. С. 528—537.
11. Best Order Sort: A New Algorithm to Non-dominated Sorting for Evolutionary Multi-objective Optimization //. 2016.