Наука и Образование
МГТУ им. Н.Э. Баумана
Сетевое научное издание
ISSN 1994-0408

Наука и Образование. МГТУ им. Н.Э. Баумана. Электрон. журн. 2016. № 12. С. 154–181.

DOI: **10.7463/1216.0852423**

Представлена в редакцию: 09.11.2016
Исправлена: 23.11.2016

УДК 519.6

Исследование эффективности и настройка модифицированного алгоритма «хищник-жертва» в задаче многоцелевой оптимизации

Карпенко А. П.[1,*], **Пугачев А. В.**[1]

[*]apkarpenko@mail.ru

[1]МГТУ им. Н.Э. Баумана, Москва, Россия

Рассматриваем класс алгоритмов многоцелевой оптимизации, которые предполагают предварительное построение некоторой конечномерной аппроксимации множества Парето, а тем самым и фронта Парето этой задачи. Представляем канонический алгоритм «хищник-жертва». Показываем его недостатки. Предлагаем ряд модификаций канонического алгоритма, имеющих целью преодоление этих недостатков. Первой целью работы является широкое исследование эффективности указанных модификаций алгоритма. Особенность исследования заключается в использовании индикаторов качества Парето-аппроксимации, не используемых в предшествующих публикациях. Вторая цель работы состоит в определении оптимальных значений некоторых свободных параметров модифицированного алгоритма «хищник-жертва».

Ключевые слова: многоцелевая оптимизация; Парето-аппроксимация; алгоритм «хищник-жертва»; метаоптимизация

Введение

Рассматриваем относительно новый и быстро развивающийся класс алгоритмов многоцелевой оптимизации (МЦО) - алгоритмы Парето-аппроксимации, которые предполагают предварительное построение некоторой конечномерной аппроксимации множества Парето, а тем самым и фронта Парето. Алгоритм «хищник-жертва» относится к *популяционным* алгоритмам этого класса, характерной чертой которых является отыскание на каждой итерации некоторого набора точек, близких к множеству или к фронту Парето данной МЦО-задачи [1].

Среди МЦО популяционных алгоритмов выделяют следующие классы: алгоритмы на основе лексикографической селекции; алгоритмы чередующихся целевых функций; алгоритмы на основе ранжирования агентов популяции; алгоритмы, не использующие ранжирование агентов [2].

Алгоритм на основе *лексикографической турнирной селекции* (*lexicographic tournament selection*) исходит из того, что частные целевые функции упорядочены по важности,

так что самой важной является первая функция, следующей по важности – вторая функция и так далее [2].

К классу алгоритмов *чередующихся целевых функций* относят алгоритмы переключающихся функций и динамического соседства [2]. В плане логики использования целевых функций именно к алгоритмам этого класса близок алгоритм «хищник-жертва».

Алгоритмы на основе *ранжирования агентов*. Основным понятием алгоритмов данного класса является понятие *ранга агента популяции*. Обзор правил вычисления рангов представлен, например, в работе [2].

Алгоритмы, не использующие ранжирование агентов. Данный класс алгоритмов представляет сигма-алгоритм, алгоритмы композитных точек и гиперкубов, а также некоторые другие алгоритмы [2].

Алгоритм «хищник-жертва» (*predator-prey*) предложили Лоуманс (*M.Laumans*) с соавторами в 1998 [3]. Идея алгоритма состоит в следующем. Имеется небольшое число хищников и большое число жертв, случайно расположенных в узлах тороидальной сетки. Каждый из хищников ассоциирован с одной из целевых функций, жертвы ассоциированы с потенциальными решениями (точками, принадлежащими множеству Парето рассматриваемой МЦО-задачи). Хищник в своей окрестности убивает самую слабую жертву (жертву, которой соответствует худшее решение). В процессе эволюции популяции жертв появляется более сильная жертва, которая заменяют слабую. Жертвы остаются неподвижными, в то время как хищники перемещаются по узлам указанной сетки.

Известно значительное число модификаций алгоритма «хищник-жертва», имеющие целью повышение его эффективности (см., например, публикации [4 - 9]). Однако, в целом, данный алгоритм в настоящее время не достаточно исследован и обладает высоким потенциалом повышения эффективности. Первой целью работы является широкое исследование эффективности модификации алгоритма, предложенной в работе [6]. Особенность исследования заключается в использовании индикаторов качества Парето-аппроксимации, не используемых в предшествующих публикациях.

Алгоритм «хищник-жертва», как и большинство других популяционных алгоритмов оптимизации, обладает некоторым числом свободных параметров, от значений которых может в значительной степени зависеть его эффективность. Задача отыскания в некотором смысле наилучших значений этих параметров называется задачей настройки параметров алгоритма оптимизации или, иными словами, задачей мета-оптимизации этого алгоритма [10]. Вторая цель работы состоит в определении оптимальных значений некоторых свободных параметров рассматриваемого модифицированного алгоритма «хищник-жертва».

Научная новизна работы заключается в результатах исследования эффективности алгоритма и результатах его настройки.

1. Постановка задачи многоцелевой оптимизации

Полагаем, что множеством допустимых значений вектора варьируемых параметров X является параллелепипед $D_X \in R^{|X|}$. Целевая вектор-функция

$F(X)=\left(f_1(X),f_2(X),...,f_{|F|}(X)\right)$ со значениями в критериальном пространстве $R^{|F|}$ определена в этом параллелепипеде. Здесь и далее запись вида $|A|$, где A - вектор или некоторое счетное множество, означает размерность этого вектора или мощность множества соответственно.

Лицо, принимающие решение (ЛПР), стремится минимизировать в этой области каждую из частных целевых функций. Целевая функция $F(X)$ выполняет отображение множества Π во *множество достижимости* D_F. Решением поставленной МЦО-задачи называем ее множество и фронт Парето $D_X^* \subset D_X$, $D_F^* \subset D_F$ соответственно. Конечномерные аппроксимации этих множеств обозначаем Θ^X, Θ^F и называем архивными.

2. Алгоритм «хищник-жертва» и его модификации

Базовый алгоритм

Каждая из жертв представляет собой один вектор решений МЦО-задачи, а каждый хищник - одну целевую функцию. Популяция эволюционирует на тороидальной сетке (рисунок 1), в узлах которой случайным образом инициализированы жертвы и хищники.

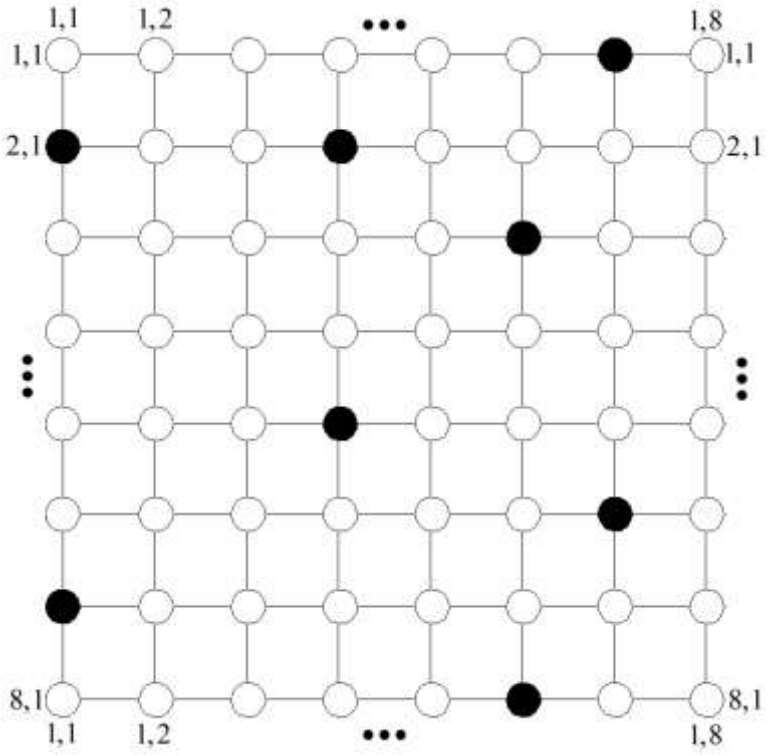


Рисунок 1 - Пример размещения хищников (закрашенные кружки) и жертв (не закрашенные кружки) на тороидальной сетке

Для каждого хищника рассматриваем все жертвы в его окрестности и удаляем жертву с худшим (наибольшим) значением целевой функции. Затем в той же окрестности вы-

бираем и модифицируем случайную жертву. Полученное решение помещаем на место удаленной жертвы. Хищника случайным образом перемещаем в один из соседних с ним узлов сетки. Процесс итерационно повторяем до выполнения условий окончания.

Схема базового алгоритма «хищник-жертва» имеет следующий вид [3].

1) В параллелепипеде D_X случайным образом инициализируем популяцию S жертв, то есть решений X_i, $i \in [1:|S|]$.
2) Ставим в соответствие жертвам случайные вершины ненаправленного связного графа в виде тороидальной решетки (рисунок 1).
3) Случайным образом помещаем хищников в вершины указанного графа.
4) Ставим в соответствие каждому хищнику одну из целевых функций $f_k(X)$, $k \in [1:|F|]$ таким образом, чтобы каждая из этих функций была представлена, по меньше мере, одним хищником.
5) Вычисляем значения соответствующих целевых функций для всех жертв в окрестности каждого их хищников и выбираем худшие жертвы.
6) Полагаем, что выбранные жертвы поглощены хищниками, и удаляем их из популяции.
7) Взамен каждого из удаленных агентов создаем новую жертву путем изменения случайной выбранной жертвы в окрестности удаленного агента.
8) Каждого из хищников случайным образом перемещаем с некоторой вероятностью в один из соседних с ним узлов решетки.
9) Если условие окончания итерации не выполнено, возвращаемся к шагу 5.

Экспериментально показано, что с ростом числа поколений популяция жертв стремится к фронту Парето, то есть представляет собой искомую Парето-аппроксимацию. Недостатком рассмотренного базового алгоритма «хищник жертва» является возможная потеря лучших (ближайших к множеству Парето) решений, неравномерность архивного множества, вычислительная сложность, необходимость использования большого числа жертв [6].

2. Модификации базового алгоритма

Модифицированную схему алгоритма «хищник-жертва» задает следующая последовательность шагов [6].

1) В параллелепипеде D_X случайным образом инициализируем популяцию жертв, то есть решений X_i, $i \in [1:|S|]$, где $|S| = I \times J$; I, J – размерности тороидальной решетки (рисунок 1).
2) Помещаем жертв в вершины указанной решетки.
3) Случайным образом помещаем хищников в центры клеток решетки. Число хищников полагаем равным

$$M_P = |F| \rfloor |S| / 20 \lfloor,$$

где $\rfloor\lfloor$ - символ целой части числа. Ставим в соответствие каждому хищнику одну из целевых функций $f_k(X)$, $k \in [1:|F|]$ и ее вес (см. ниже).

4) Вычисляем значения соответствующих целевых функций для всех жертв, и в окрестности каждого хищника выбираем худшие жертвы. Каждую из худших жертв заменяем ее потомком, полученным путем скрещивания двух сильнейших жертв и последующей мутацией полученного решения.
5) Указанную в п. 4 замену производим при выполнении трёх следующих условий:
   - с точки зрения значения целевой функции хищника, потомок должен быть лучше, чем заменяемая жертва;
   - потомок не доминирует остальные три жертвы (по взвешенному значению целевых функций);
   - потомок не находится в окрестностях остальных жертв.

   Если заданное число попыток найти потомка, удовлетворяющего указанным условиям, оказываются неудачными, то худшую жертву не заменяем.
6) Каждого из хищников перемещаем случайным образом с некоторой вероятностью в один из соседних с ним узлов решетки при выполнении условия

$$cellvisit\,(i,j) \le cellvisit_{avg} + 1,$$

   где *cellvisit* (*i,j*) - число посещений ячейки (*i,j*); $cellvisit_{avg}$ – среднее число посещений ячеек всей решетки.
7) Если условие окончания итерации не выполнено, то возвращаемся к шагу 4.

В качестве оператора скрещивания используем оператор *BLX*-α [11]

$$x_v^{(1,t+1)} = (1 - y_v)x_v^{(1,t)} + y_v x_v^{(2,t)},$$

$$x_v^{(2,t+1)} = y_v x_v^{(1,t)} + (1 - y_v)x_v^{(2,t)},$$

$$y_v = (1 + 2\alpha)\nu_v - \alpha,$$

где $x_v^{(1,t)}, x_v^{(2,t)}$ - компоненты векторов двух лучших жертв; $x_v^{(1,t+1)}$, $x_v^{(2,t+1)}$ - компоненты потомка; ν_v - случайное число из интервала $[0;1]$; константа $\alpha = 0,5$. Подчеркнем, что базовый алгоритм «хищник-жертва» не использует оператор скрещивания.

В качестве оператора мутации применяем оператор неравномерной мутацией Михалевича [12]

$$\beta = (1 - \frac{t}{t_{\max}})^b,$$

$$y_v^{(1,t+1)} = x_v^{(1,t+1)} + \beta\tau(x_v^{(U)} - x_v^{(L)})(1 - r_v^{(1-t/t_{\max})^b}),$$

где $y_v^{(1,t+1)}$ - компонент вектора потомка; $x_v^{(U)}$, $x_v^{(L)}$ - максимальное и минимальное значения компонента вектора потомка; величина τ принимает значение 1 и -1 с вероятно-

стью 0,5; t, $t_{\max}$ - номера текущего поколения и максимального поколений соответственно; b - свободный параметр; r_v - - случайное число из интервала $[0;1]$. Вероятность мутации принадлежит интервалу $[0{,}05;\ 0{,}10]$. Заметим, что базовый алгоритм «хищник-жертва» в качестве оператора мутации использует оператор гауссовой (*Gauss*) мутации.

Взвешивание целевых функций осуществляем по формуле

$$f(X)=\sum_{k=1}^{|F|} w_k\ f_k(X),$$

где w_k - вес целевой функции f_k. Для двухкритериальной задачи веса распределяем по правилу $w_1 \in [0;1]$, $w_2 = 1 - w_1$. Для задач с большим числом целевых функций может быть использована квазислучайная ЛП$_\tau$ последовательность [13].

3. Исследование эффективности базового алгоритма «хищник-жертва» и его модификаций

Реализация базового и модифицированного алгоритмов выполнена на языке программирования *Java* в силу его кросс-платформенности и высокой эффективности. Для графического вывода результатов работы программы использована утилита *Gnuplot*. Общий объем программного кода составляет около 1900 строк.

Качество аппроксимации множества Парето оцениваем с помощью следующих индикаторов [14]:

- мощность множества решений (*overall non-dominated vector generation)* $I_{ONVG}(\Theta)=|\Theta| \to \max$, где Θ - архив решений;
- близость найденных решений к точному множеству Парето рассматриваемой МЦО-задачи (*generational distance*)

$$GD=\frac{\sqrt{\sum_{i=1}^{|\Theta|}\delta_i^2}}{I_{ONVG}(\Theta)},\quad i\in[1:|\Theta|],$$

  где δ_i - расстояние от i-ой точки архивных решений до ближайшего точного решения;
- равномерность распределения решений в полученной Парето-аппроксимации (*spacing*)

$$SP=\sqrt{\frac{1}{I_{ONVG}-2}\sum_{i=1}^{|\Theta|-1}(\bar{s}-s_i)^2},\quad i\in[1:|\Theta|],$$

где s_i - расстояние от i-ой архивной точки до ближайшего соседа; $\bar{s}$ - среднее значение s_i;

– число испытаний, то есть вычислений значений целевых функций n_E.

В качестве тестовых рассматриваем следующие задачи из известного набора тестовых МЦО-задач [15].

Задача Аудета (*C. Audet*). Задача является двумерной двухкритериальной; фронт Парето - непрерывный, являющийся при значении параметра $\alpha = 0{,}25$ выпуклым, а при $\alpha = 4$ - невыпуклым.

*Задача ZDT*1 (двумерная, двухкритериальная, имеет выпуклый фронт Парето).

*Задача ZDT*2 (двумерная, двухкритериальная, имеет невыпуклый фронт Парето).

*Задача ZDT*3 (двумерная, двухкритериальная). Задача имеет несвязный, хотя и выпуклый, фронт Парето.

*Задача ZDT*4 (двумерная, двухкритериальная). Особенностью задачи является наличие большого числа локальных субоптимальных фронтов.

*Задача ZDT*6 (двумерная, двухкритериальная). Задача имеет слабо невыпуклый фронт Парето.

Базовый алгоритм «хищник-жертва» обозначаем *PP*, а модифицированный – *MPP*. Значения параметров алгоритмов, использованные в вычислительном эксперименте, приведены в таблице 1 (для алгоритма *PP* использованы значения, предложенные его авторами).

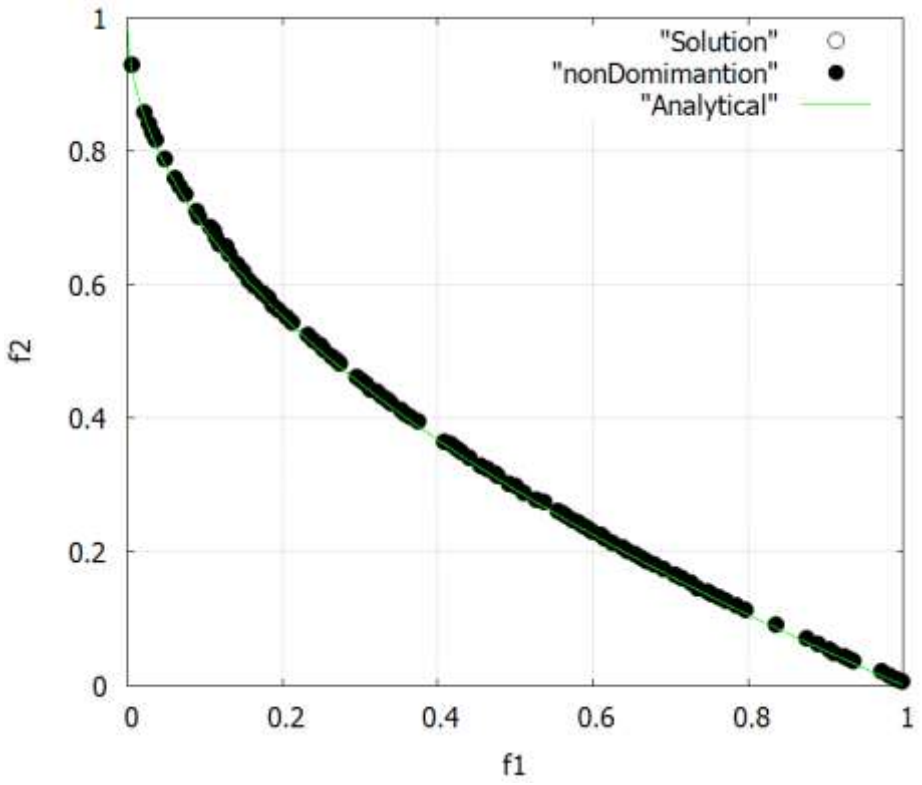
Таблица 1 - Параметры алгоритмов *MPP* и *PP*

Параметры	***MPP***	***PP***
Жертв	100	900
Хищников	10	200
Итераций	1000	1000

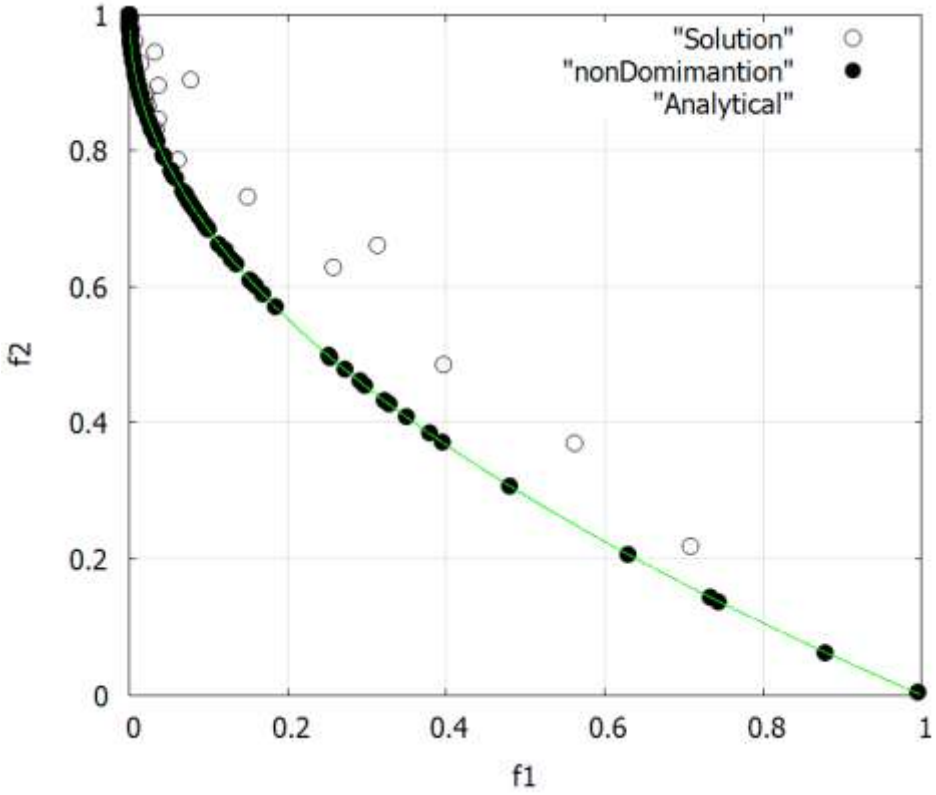
Результаты экспериментов представлены в таблицах 2 - 7 и на рисунках 2 - 7. Здесь и далее приняты следующие обозначения: совокупность светлых и темных кружков – все найденные алгоритмом решения; темные кружки – не доминируемые решения; сплошная линия – точный фронт Парето задачи.

Таблица 2 - Результаты экспериментов: тестовая задача ZDT1

Алгоритм	$n_E(f)$	$I_{ONVG}(\Theta)$	***GD***	***SP***
MPP	$1{,}6\cdot10^5$	100	$4{,}5\cdot10^{-4}$	$4{,}4\cdot10^{-4}$
PP	$4{,}0\cdot10^5$	459	$5{,}3\cdot10^{-4}$	$2{,}2\cdot10^{-3}$



а) алгоритм *MPP*

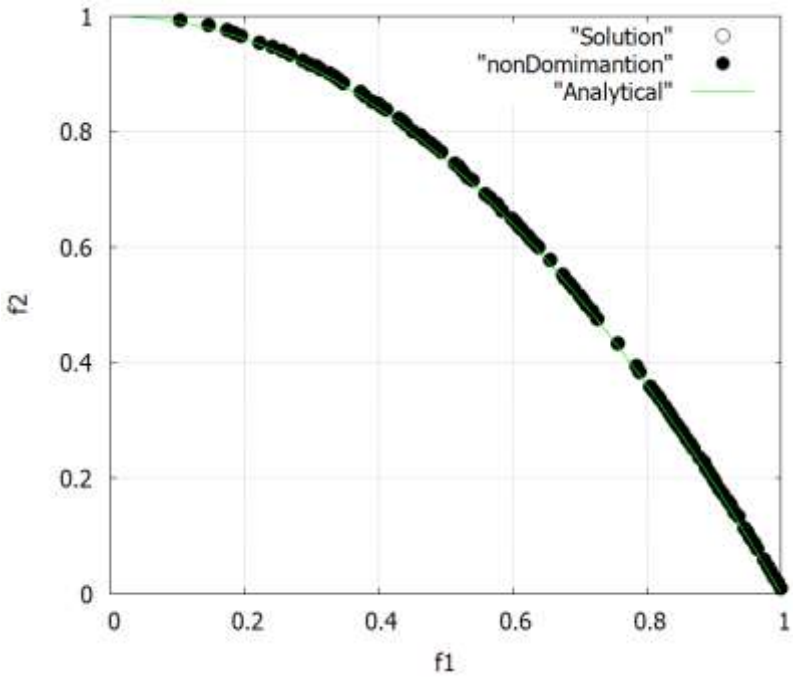


б) алгоритм *PP*

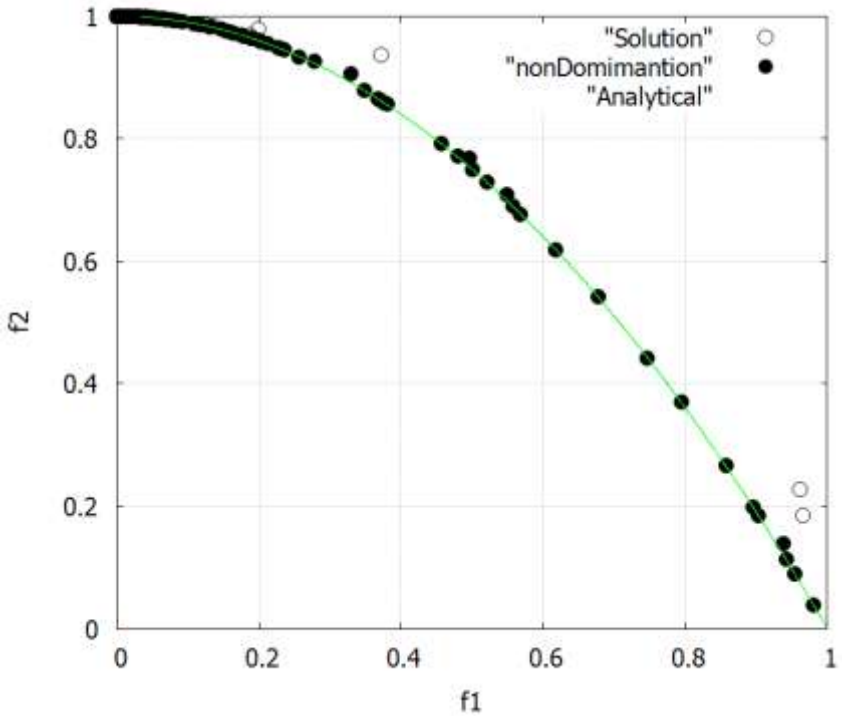
Рисунок 2 - Результаты экспериментов: тестовая задача *ZDT*1

Таблица 3 - Результаты экспериментов: тестовая задача *ZDT*2

Алгоритм	$n_E(f)$	$I_{ONVG}(\Theta)$	***GD***	***SP***
MPP	$1{,}6\cdot10^5$	100	$3{,}9\cdot10^{-4}$	$5{,}3\cdot10^{-4}$
PP	$4{,}0\cdot10^5$	164	$1{,}2\cdot10^{-4}$	$2{,}1\cdot10^{-3}$



а) алгоритм *MPP*

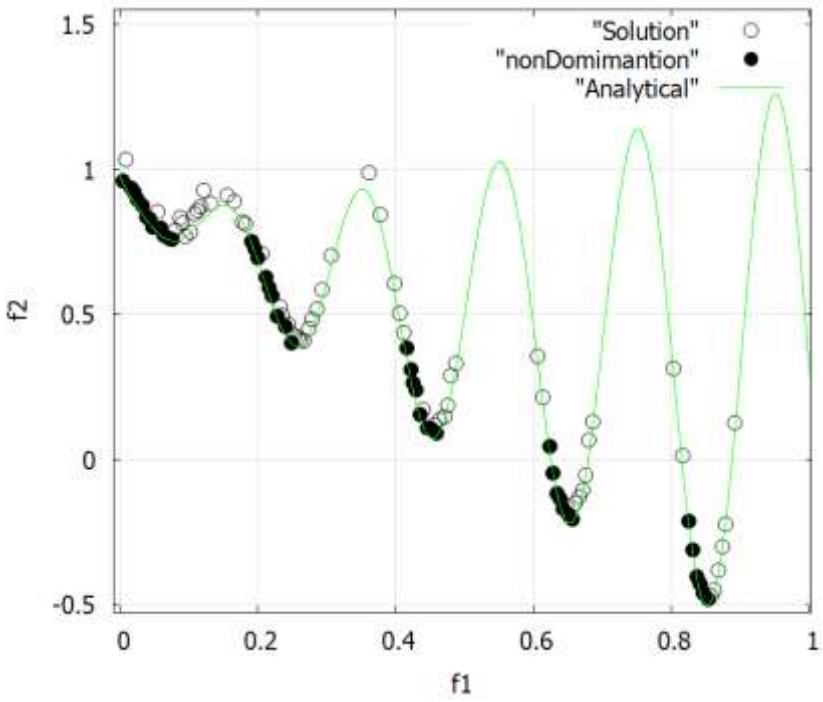


б) алгоритм *PP*

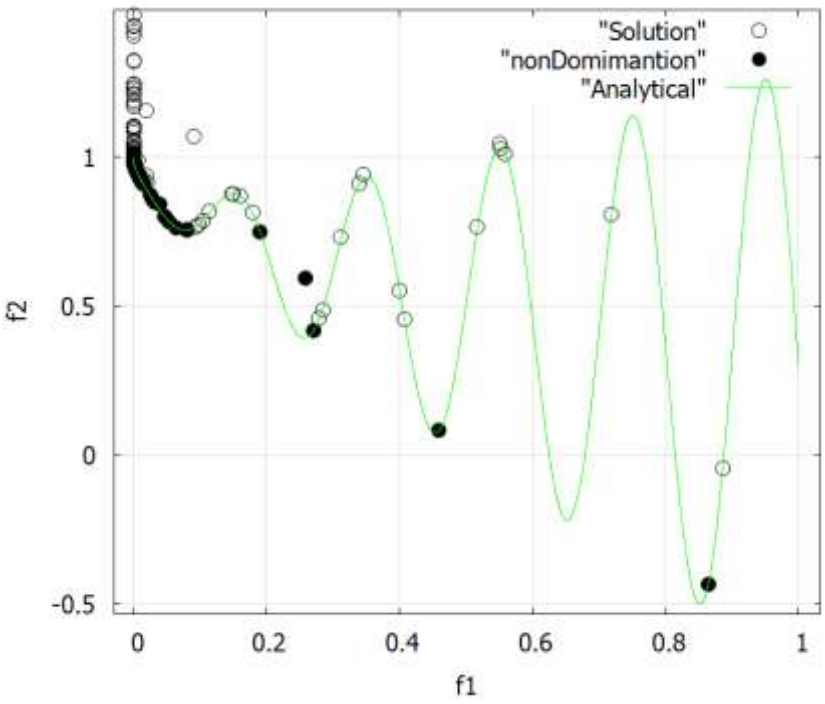
Рисунок 3 - Результаты экспериментов: тестовая задача *ZDT*2

Таблица 3 – Результаты экспериментов: тестовая задача *ZDT*3

Алгоритм	$n_E(f)$	$I_{ONVG}(\Theta)$	***GD***	***SP***
MPP	$1{,}7\cdot10^{5}$	44	$4{,}9\cdot10^{-4}$	$8{,}2\cdot10^{-3}$
PP	$4{,}0\cdot10^{5}$	412	$3{,}6\cdot10^{-4}$	$6{,}2\cdot10^{-4}$



а) алгоритм *MPP*

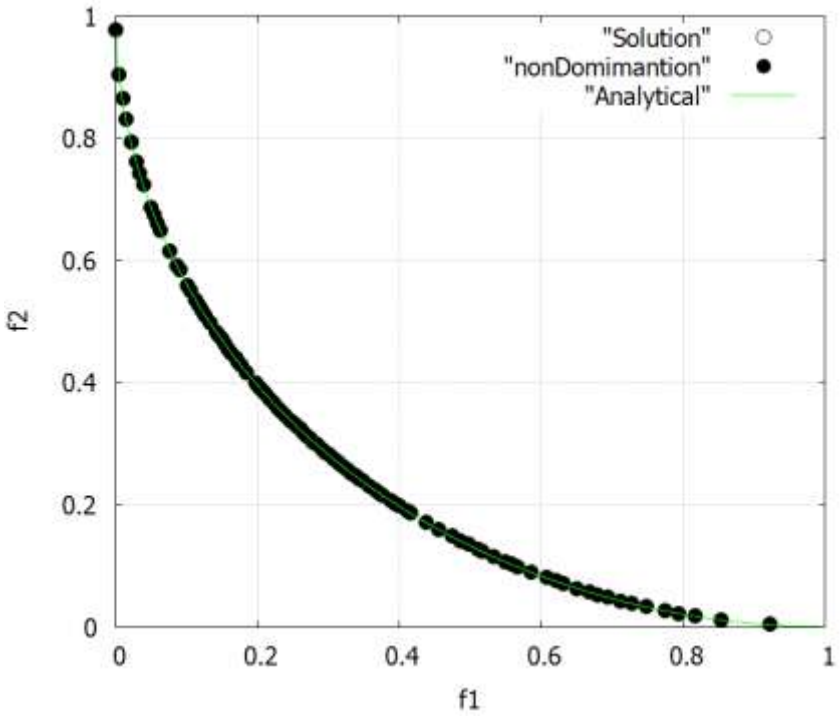


б) алгоритм *PP*

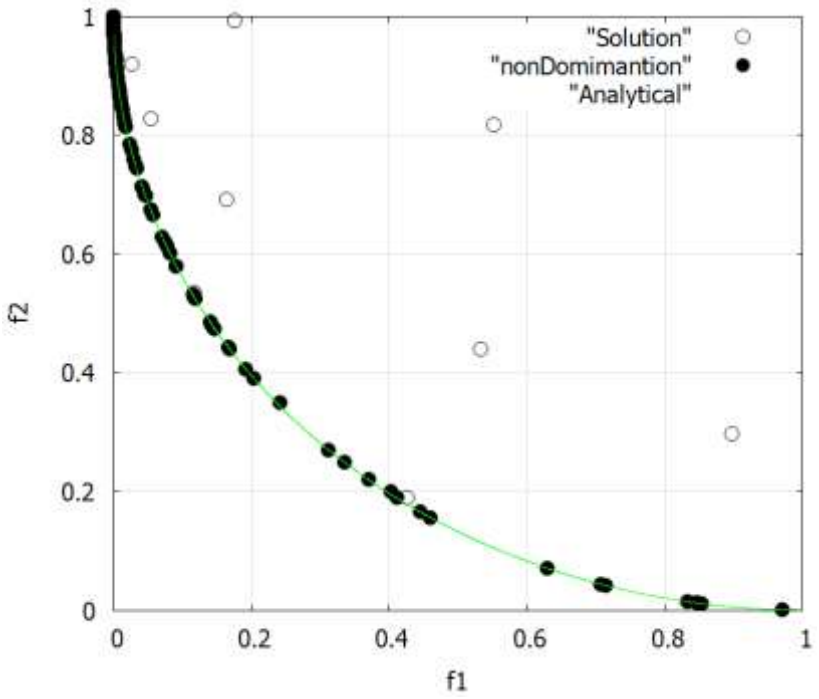
Рисунок 4 - Результаты экспериментов: тестовая задача *ZDT*3

Таблица 5 - Результаты экспериментов: тестовая задача *ZDT*4

Алгоритм	$n_E(f)$	$I_{ONVG}(\Theta)$	***GD***	***SP***
MPP	$1{,}8\cdot10^{5}$	99	$1{,}4\cdot10^{-5}$	$9{,}3\cdot10^{-5}$
PP	$4{,}0\cdot10^{5}$	488	$6{,}4\cdot10^{-5}$	$1{,}9\cdot10^{-3}$



а) алгоритм *MPP*

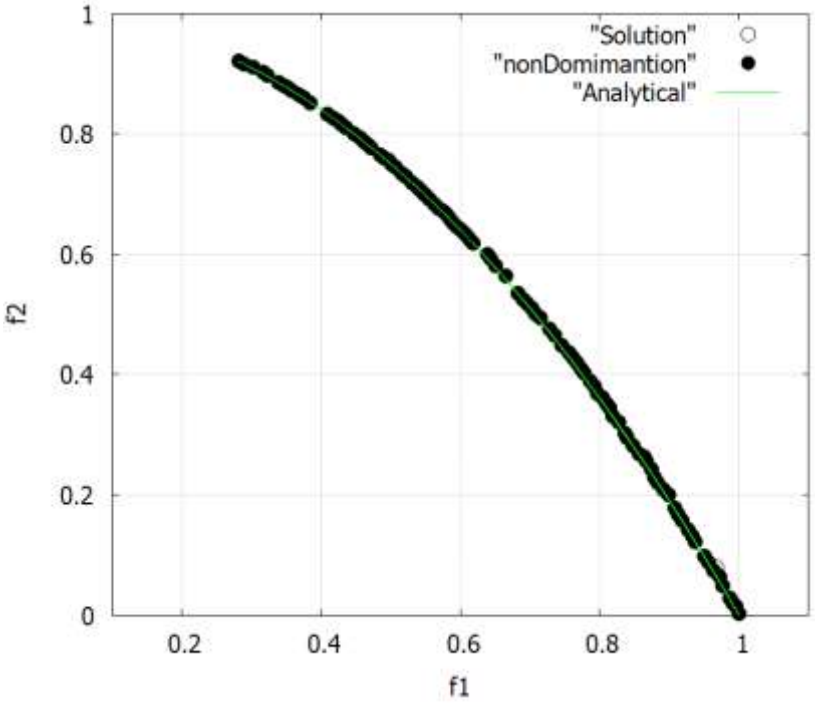


б) алгоритм PP

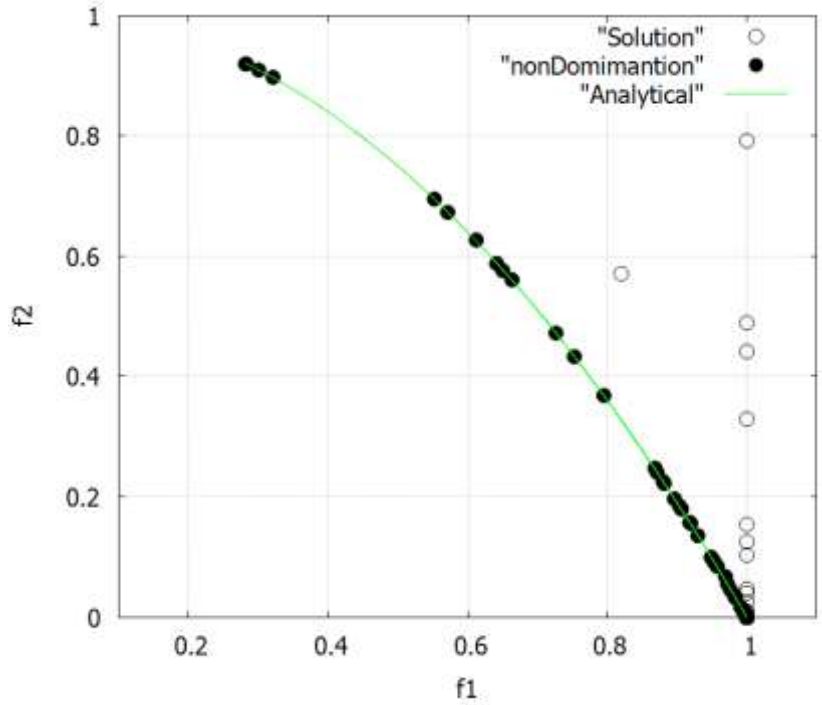
Рисунок 5 - Результаты экспериментов: тестовая задача *ZDT*4

Таблица 6 - Результаты экспериментов: тестовая задача *ZDT*6

Алгоритм	$n_E(f)$	$I_{ONVG}(\Theta)$	***GD***	***SP***
MPP	$1{,}8\cdot10^{5}$	98	$7{,}7\cdot10^{-5}$	$1{,}2\cdot10^{-4}$
PP	$4{,}0\cdot10^{5}$	162	$2{,}7\cdot10^{-5}$	$7{,}6\cdot10^{-3}$



а) алгоритм *MPP*

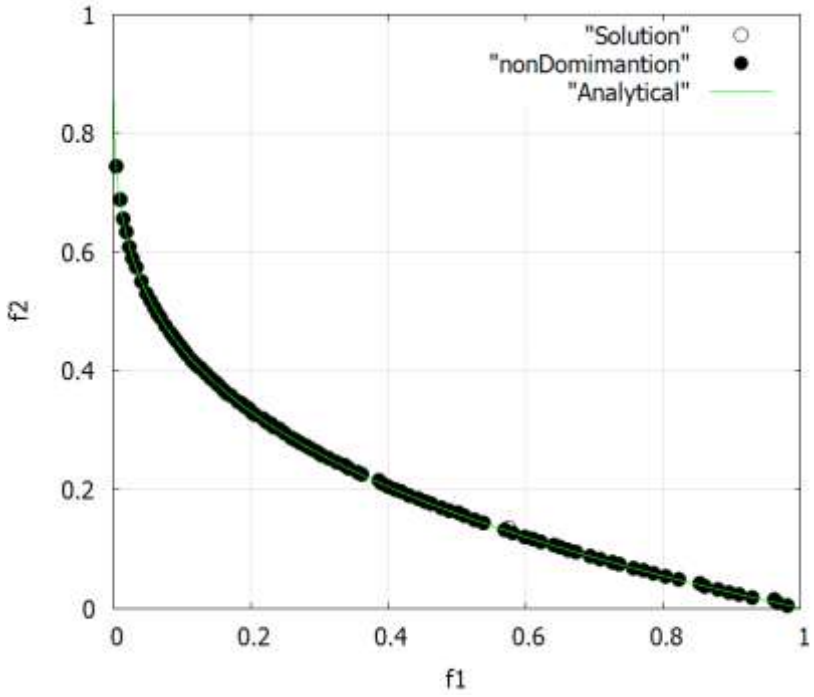


б) алгоритм *PP*

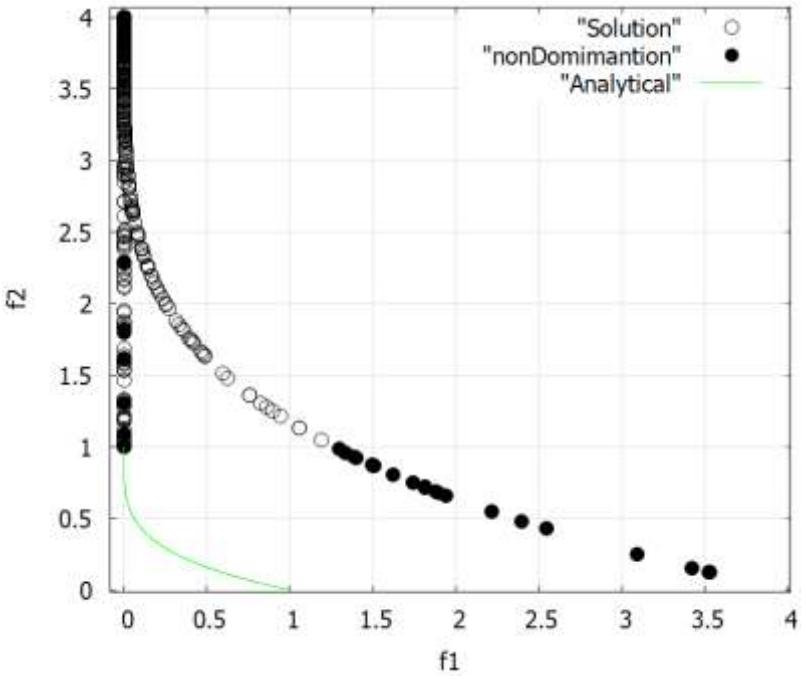
Рисунок 6 - Результаты экспериментов: тестовая задача *ZDT*6

Таблица 7 – Результаты экспериментов: тестовая задача Аудета ($\alpha = 0{,}25$)

Алгоритм	$n_E(f)$	$I_{ONVG}(\Theta)$	*GD*	*SP*
MPP	$1{,}8 \cdot 10^5$	98	$3{,}6 \cdot 10^{-2}$	$7{,}1 \cdot 10^{-4}$
PP	$4{,}0 \cdot 10^5$	28	0,4	0,3



а) алгоритм *MPP*



б) алгоритм *PP*

Рисунок 7 - Результаты экспериментов: тестовая задача Аудета ($\alpha = 0{,}25$); алгоритм *PP*

Из представленных результатов следует, что для всех рассмотренных тестовых задач модифицированный алгоритм показал лучшие результаты по сравнению с базовым алгоритмом:

- увеличена мощность множества архивных решений (индикатор $I_{ONVG}(\Theta)$);
- обеспечена большая близость найденных решений к точному множеству Парето (индикатор *GD*);
- улучшена равномерность распределения архивных решений (индикатор SP);
- уменьшено число испытаний (индикатор n_E).

Подчеркнем, что тестовая задача Аудета была решена базовым алгоритмом неверно (рисунок 7,б), в то время как модифицированный алгоритм позволил получить Парето-аппроксимацию с «хорошими» значениями рассматриваемых индикаторов качества аппроксимации.

4. Настройка параметров модифицированного алгоритма

В качестве настраиваемых рассматриваем следующие свободные параметры модифицированного алгоритма:

- число попыток модифицировать худшую жертву таким образом, чтобы потомок удовлетворил всем трем указанным в п. 3. условиям;
- число итераций.

Для оценки качества полученной Парето-аппроксимации используем индикаторы, представленные в п.3:

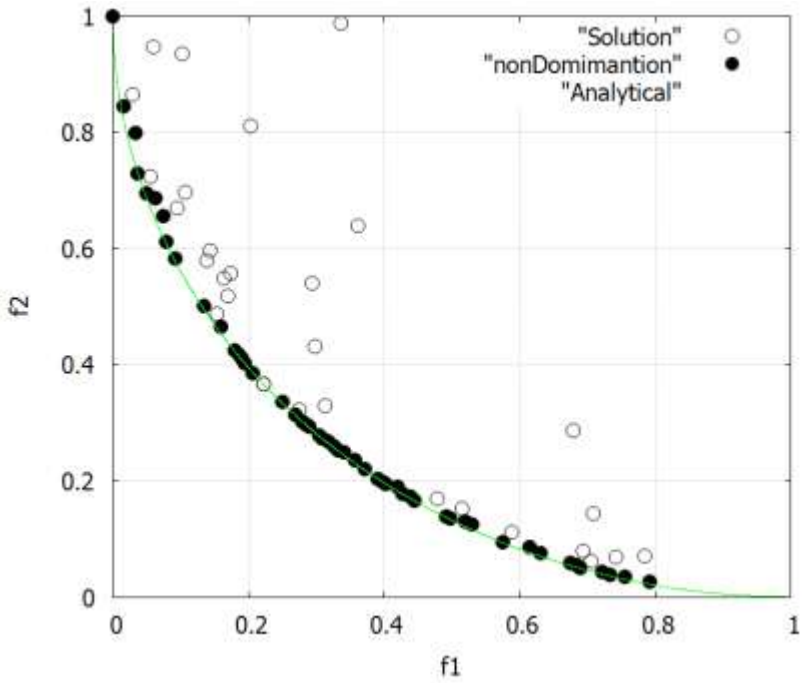
- мощность множества решений $I_{ONVG}(\Theta)$;
- близость найденных решений к точному множеству Парето GD;
- равномерность распределения решений SP.

Настройку производим для тестовой задачи *ZTD*4 (п. 3).

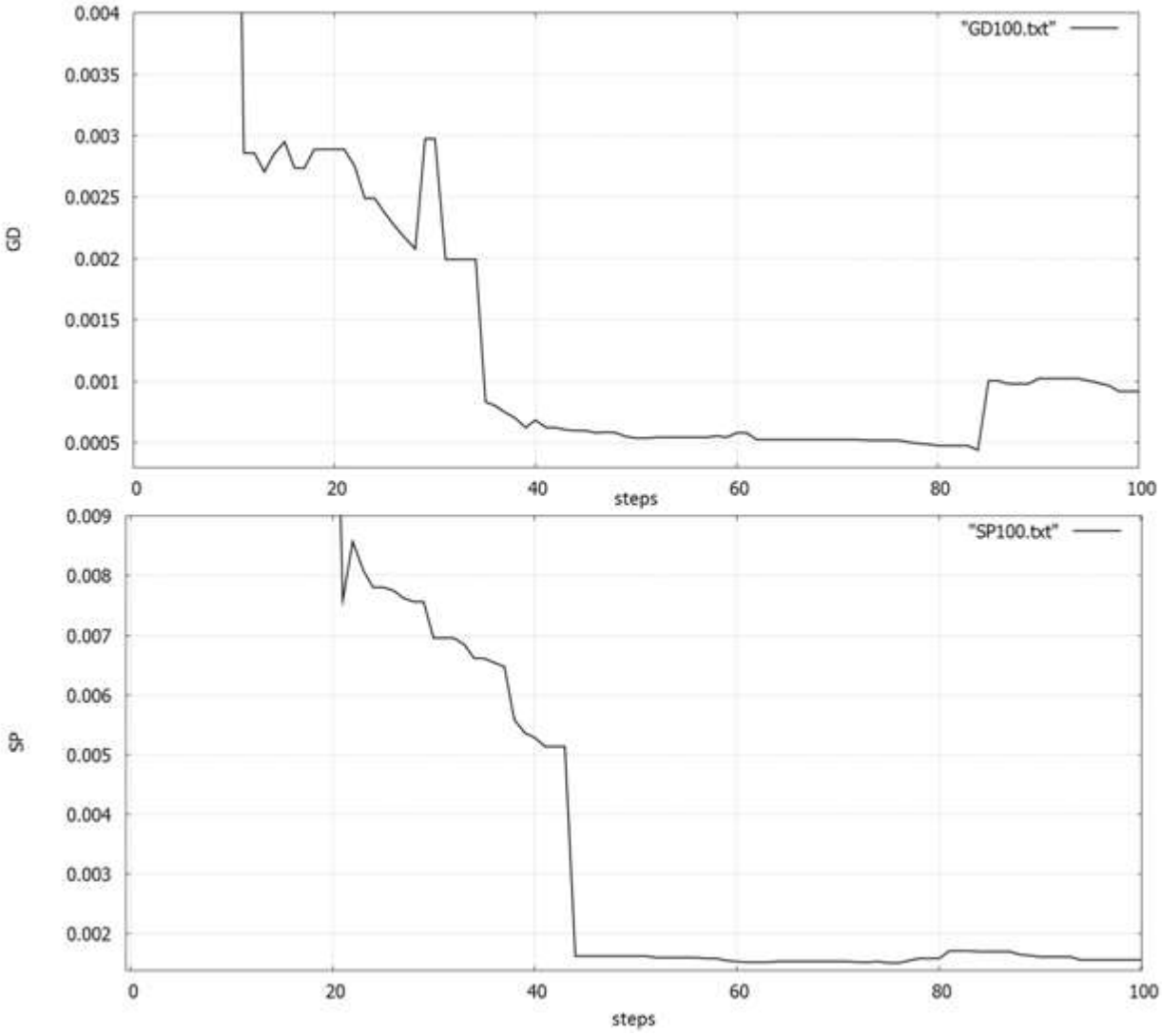
Результаты вычислительных экспериментов представлены в таблице 8 и на рисунках 8 - 13.

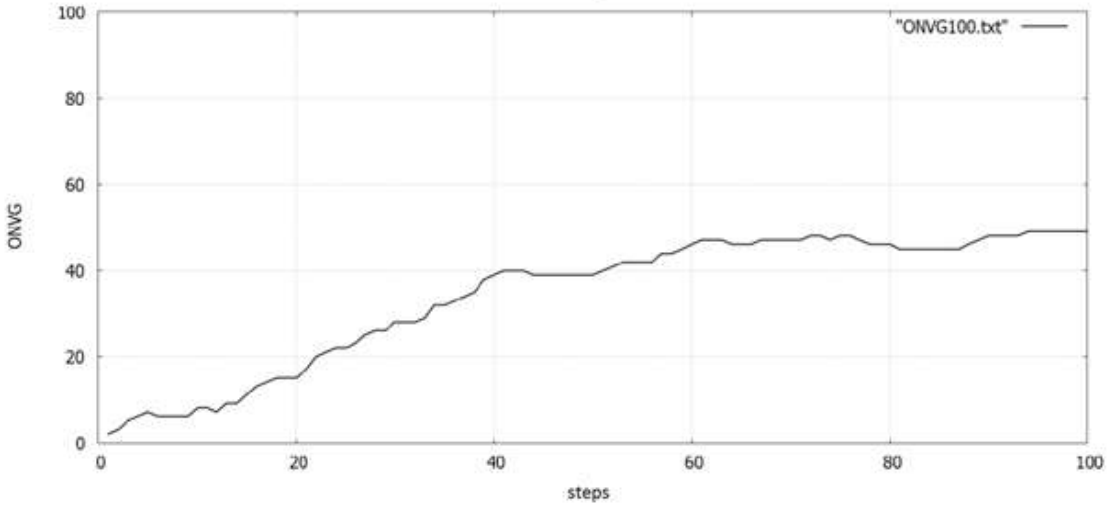
Таблица 8 - Результаты настройки для тестовой задачи *ZTD*4

№ теста	**Число попыток**	**Число итераций**	$n_E(f)$	$I_{ONVG}(\Theta)$	***GD***	***SP***
1	10	100	$1{,}6\cdot10^{4}$	51	$8{,}7\cdot10^{-4}$	$6{,}7\cdot10^{-3}$
2	10	1000	$1{,}7\cdot10^{5}$	92	$1{,}6\cdot10^{-4}$	$1{,}4\cdot10^{-3}$
3	10	10000	$1{,}8\cdot10^{6}$	99	$4{,}2\cdot10^{-5}$	$7{,}8\cdot10^{-4}$
4	30	1000	$4{,}9\cdot10^{5}$	95	$9{,}6\cdot10^{-5}$	$2{,}2\cdot10^{-3}$
5	60	1000	$9{,}6\cdot10^{5}$	98	$2{,}8\cdot10^{-5}$	$7{,}1\cdot10^{-4}$
6	100	1000	$1{,}5\cdot10^{6}$	100	$2{,}6\cdot10^{-5}$	$1{,}6\cdot10^{-3}$



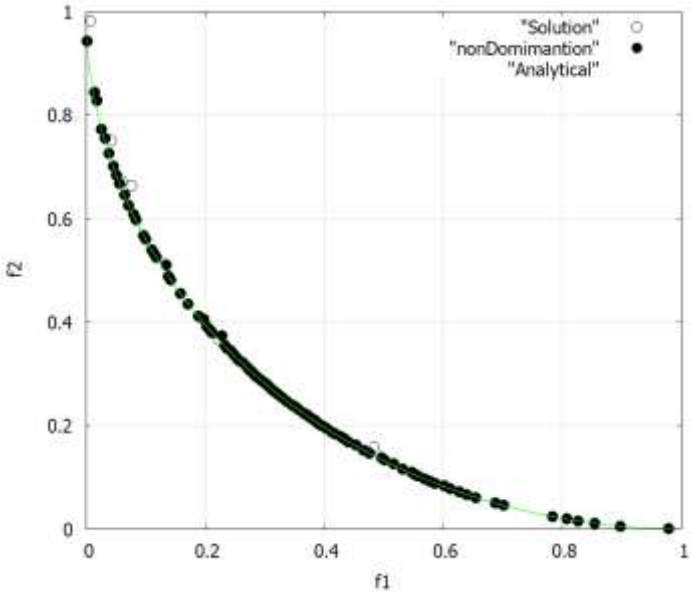
а) аппроксимация фронта Парето



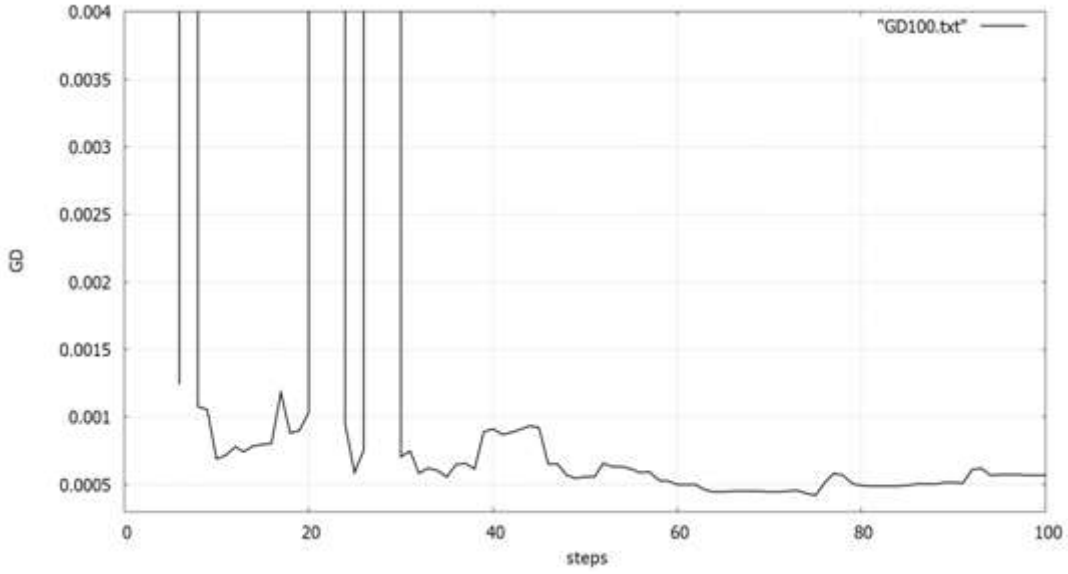


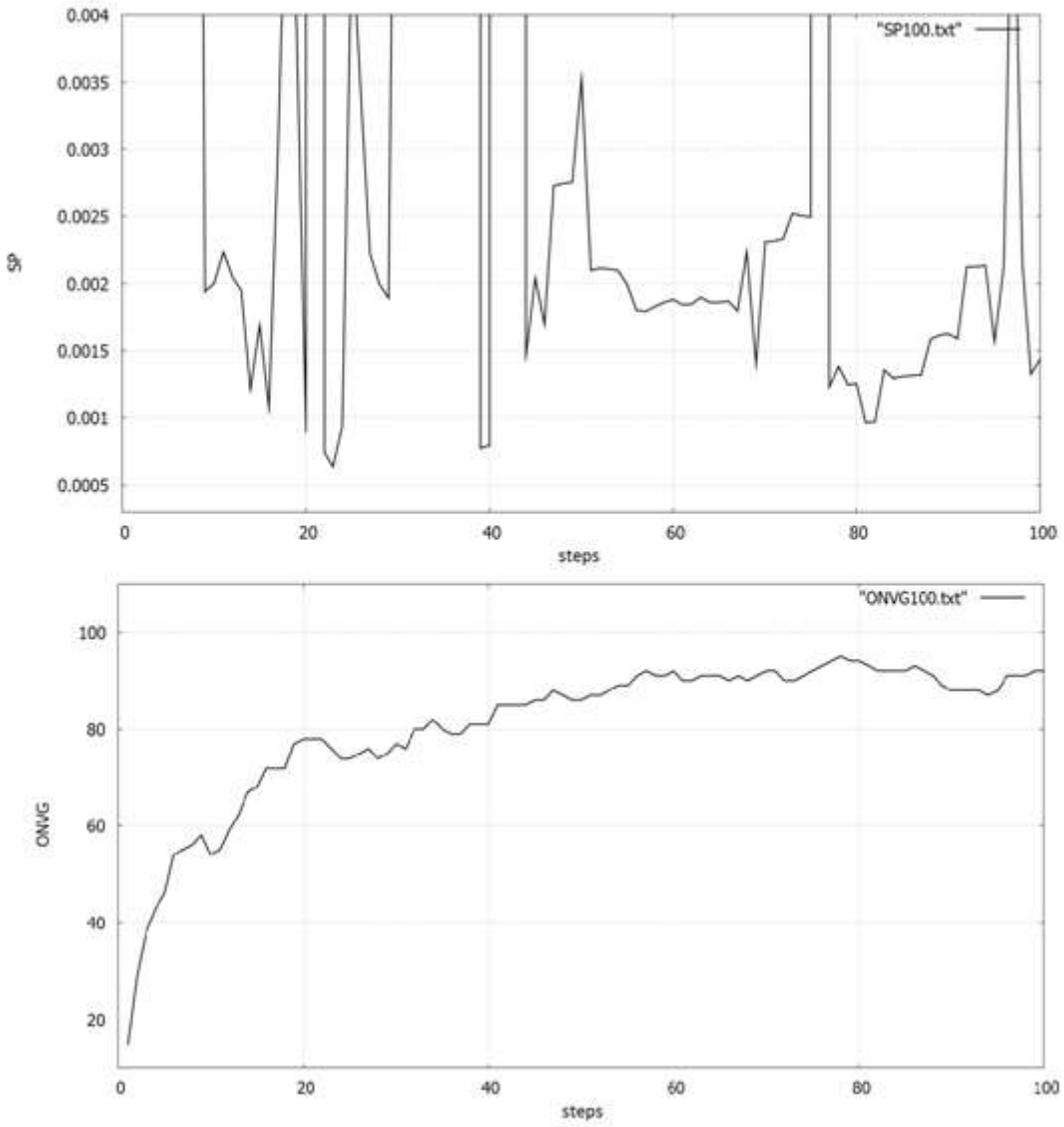
б) Индикаторы качества

Рисунок 8 – Результаты настройки: задача *ZTD*4; тест 1



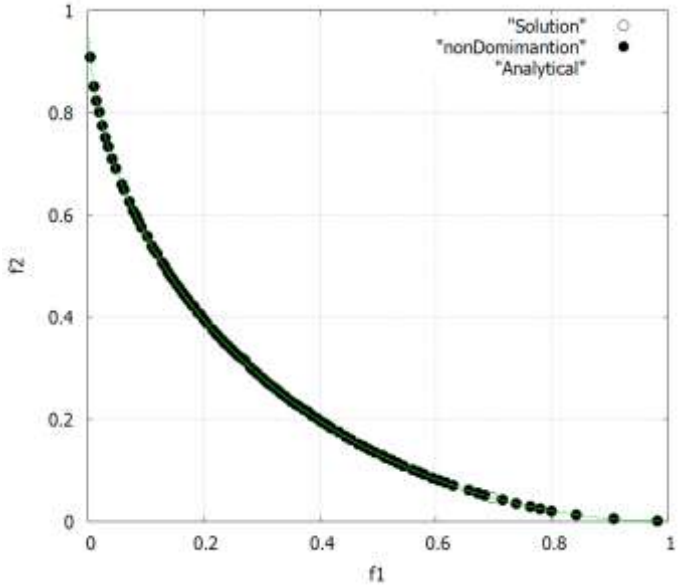
а) аппроксимация фронта Парето



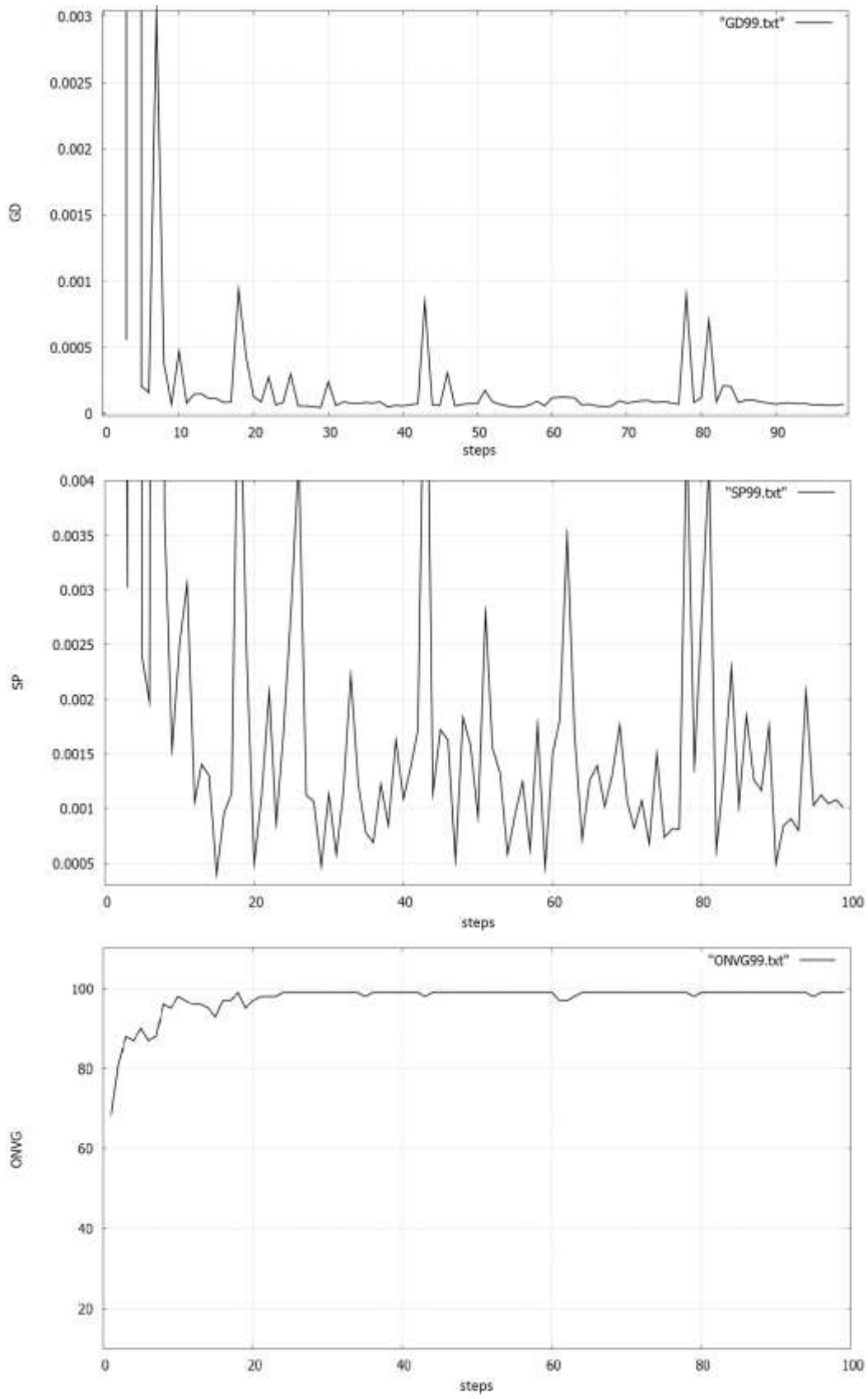


б) Индикаторы качества

Рисунок 9 – Результаты настройки: задача *ZTD*4; тест 2

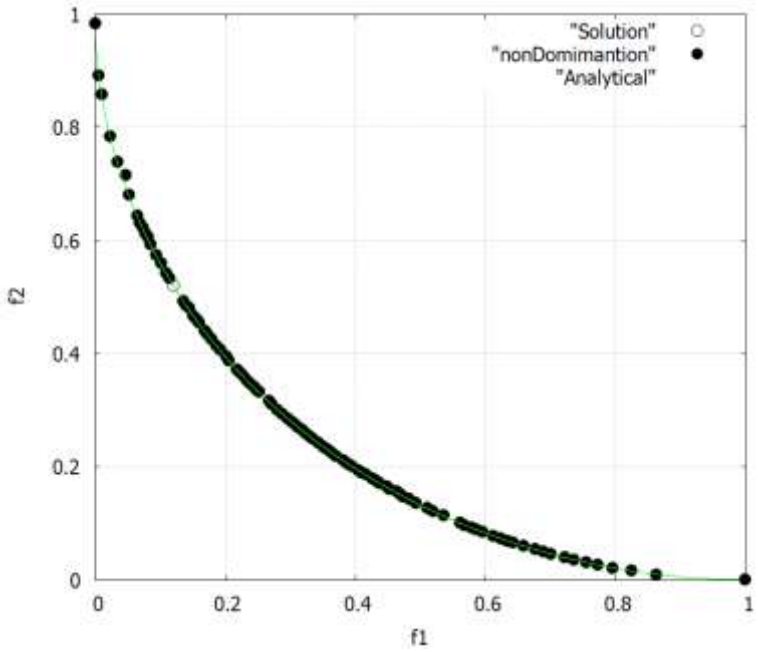


а) Аппроксимация фронта Парето

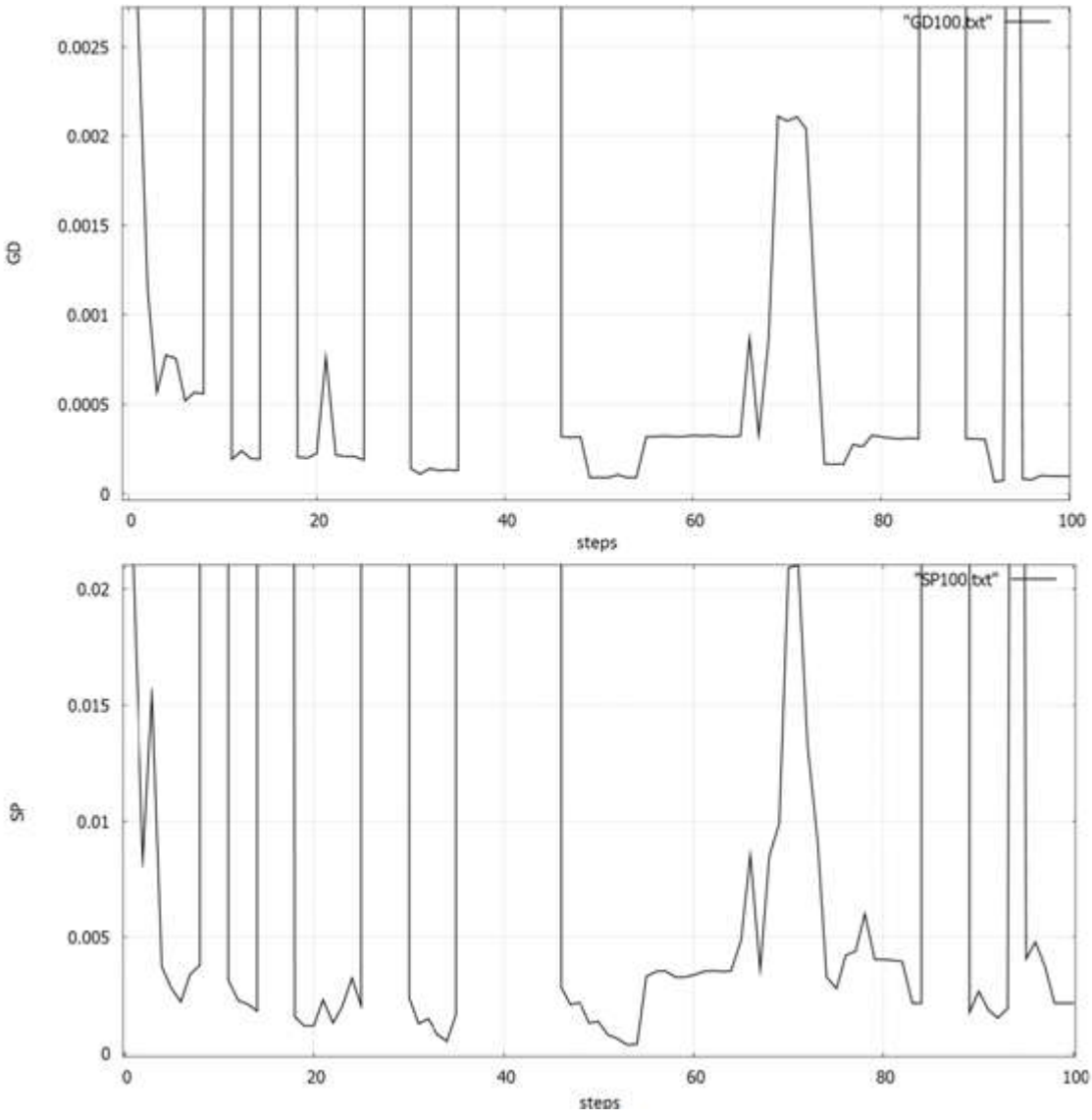


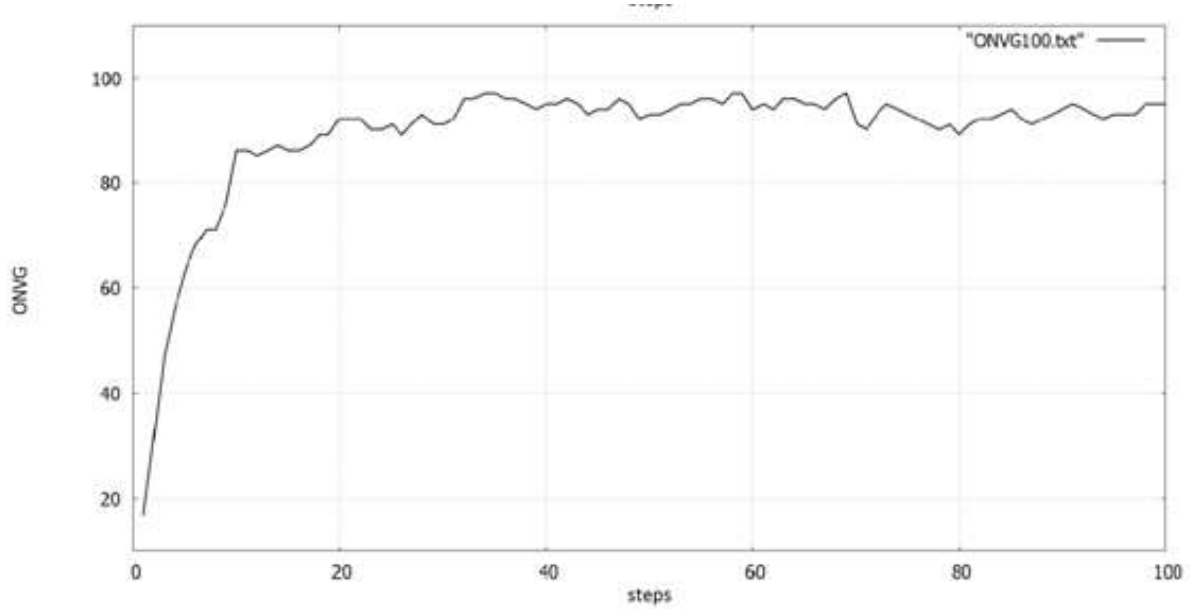
б) Индикаторы качества

Рисунок 10 – Результаты настройки: задача *ZTD*4; тест 3



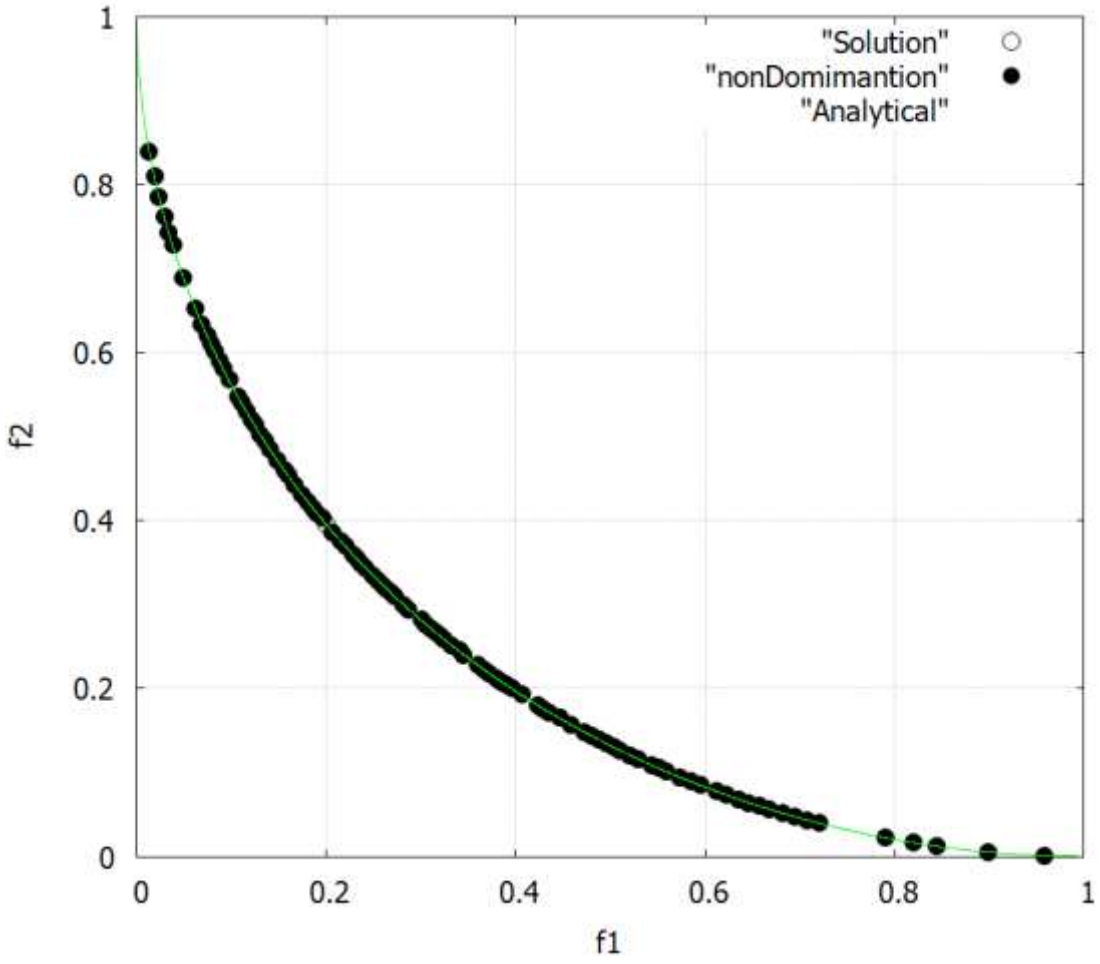
а) Аппроксимация фронта Парето



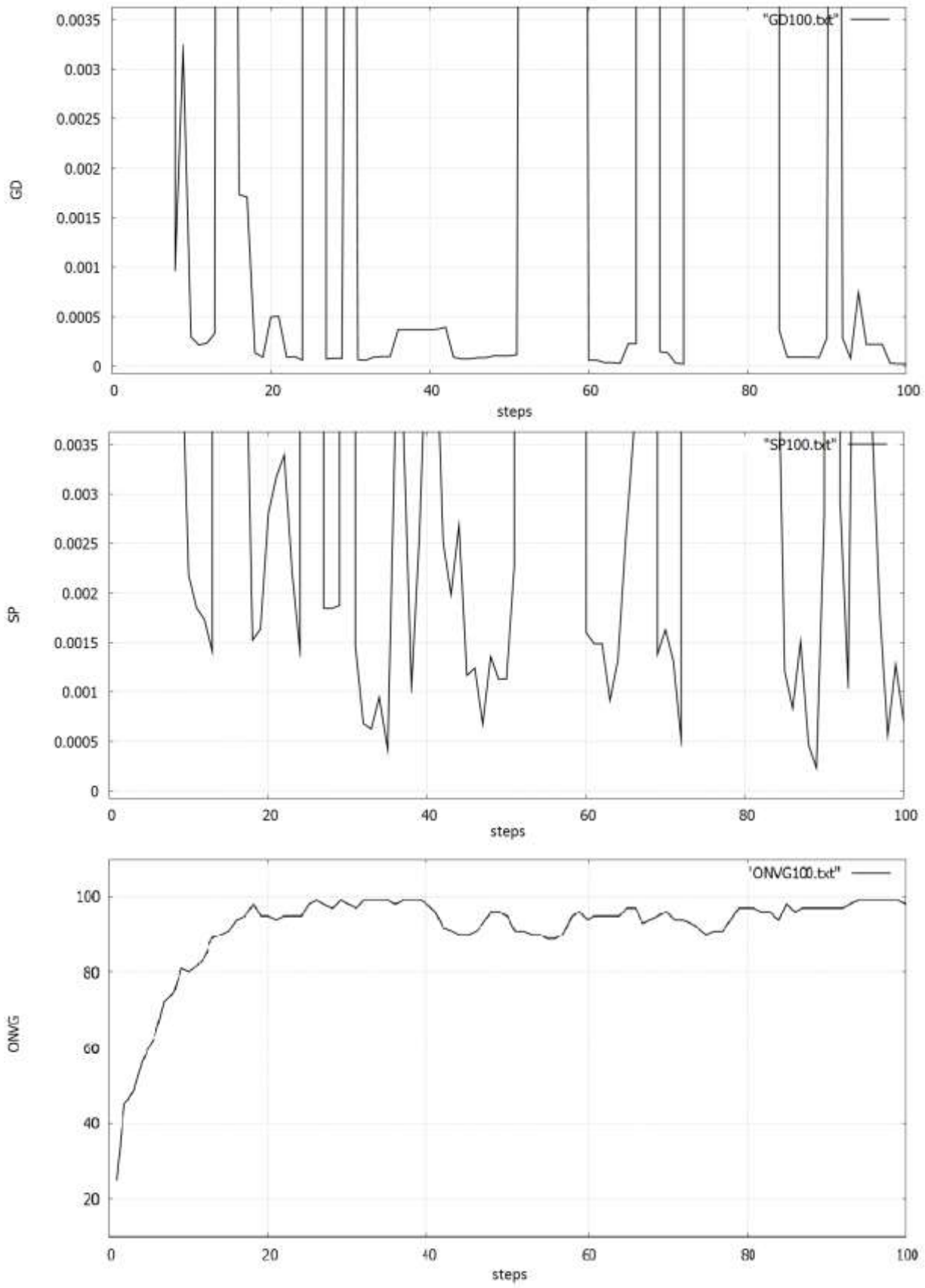


б) Индикаторы качества

Рисунок 11 – Результаты настройки: задача *ZTD*4; тест 4

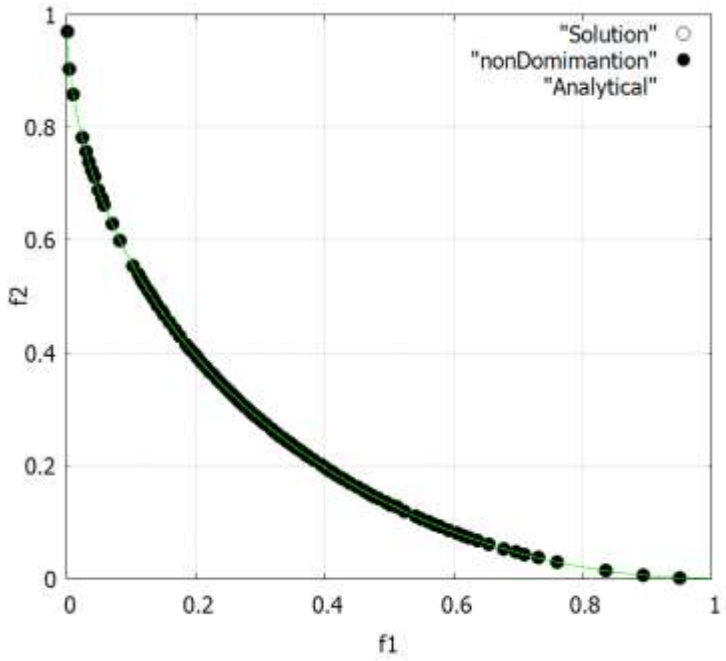


а) Аппроксимация фронта Парето

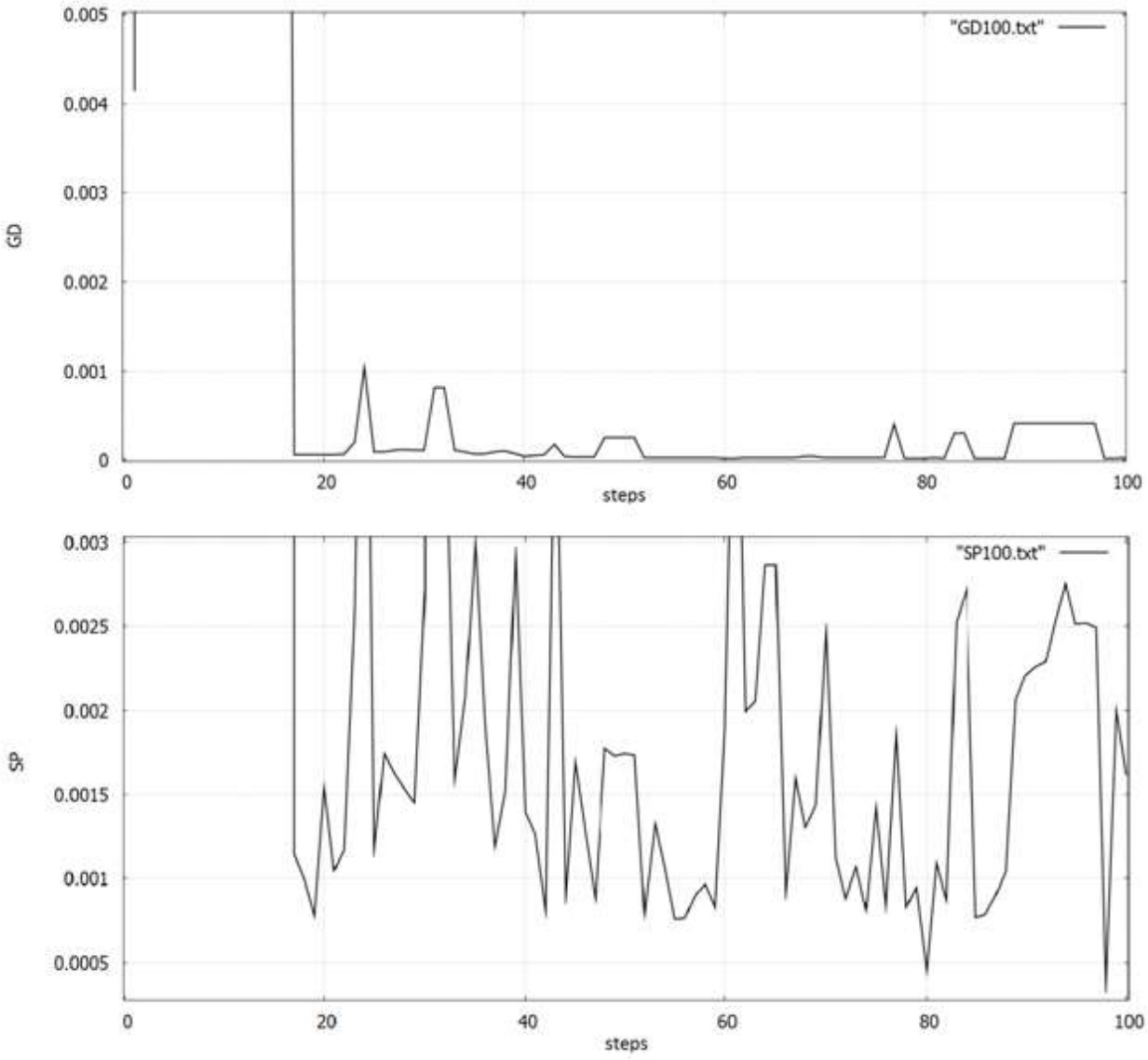


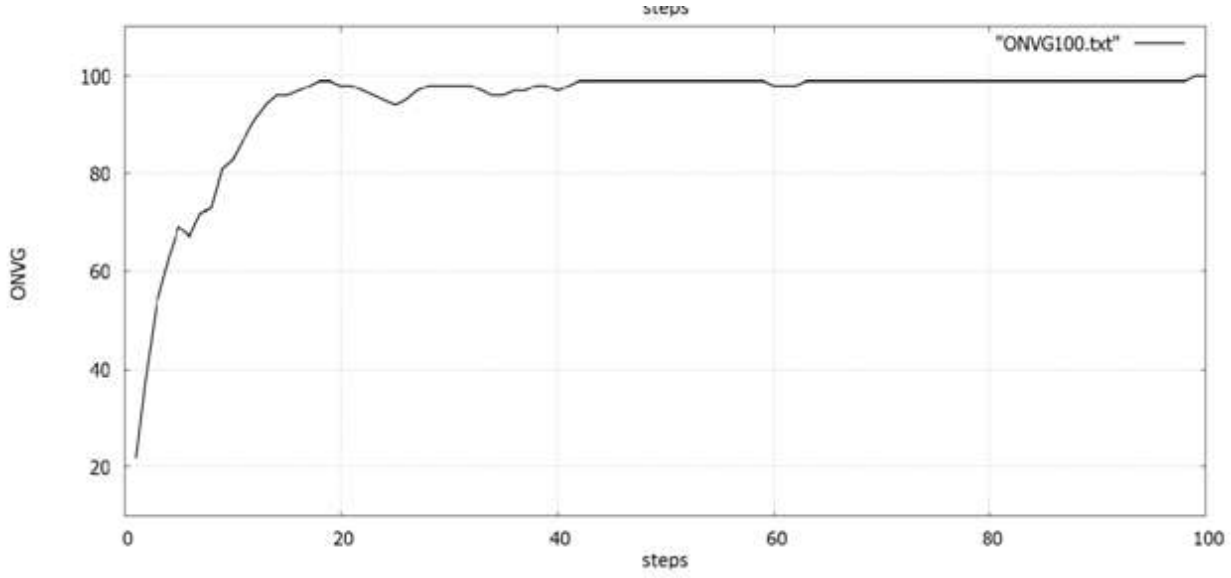
б) Индикаторы качества

Рисунок 12 – Результаты настройки: задача *ZTD*4; тест 5



а) Аппроксимация фронта Парето





б) Индикаторы качества

Рисунок 13 – Результаты настройки: задача *ZTD*4; тест 6

Заключение

В работе выполнен обзор популяционных и непопуляционных алгоритмов Парето-аппроксимации, выявлены их достоинства и недостатки. Представлен канонический алгоритм «хищник-жертва», показаны его недостатки. Предложены модификации канонического алгоритма, имеющие целью преодоление этих недостатков:

- добавлен оператор скрещивания и изменен оператор мутации, что должно привести к большей вероятности нахождения лучших решений;
- для повышения равномерности распределения архивных решений, для каждой из жертв введена окрестность, в которую не могут попасть другие жертвы;
- уменьшена окрестность хищника, что должно уменьшить вычислительную сложность алгоритма.

Исследование эффективности модифицированного алгоритма «хищник-жертва» показали, что указанные модификации позволили улучшить следующие показатели базового алгоритма:

- мощность множества архивных решений;
- равномерность распределения архивных решений;
- время вычислений.

Выполнен анализ эффективности модифицированного алгоритма в зависимости от значений его следующих свободных параметров:

- число итераций;
- число повторных проверок потомка жертвы на адаптацию.

Показано, что увеличение на порядок числа итераций равносильно такому же увеличению числа испытаний. Небольшие отклонения от этой закономерности являются результатом стохастичности алгоритма.

Был выявлен недостаток использования в качестве условия окончания итераций достижение их заданного числа.

В целом, результаты исследований показали, что модифицированный алгоритм позволяет достичь той же точности аппроксимации, что и базовый алгоритм, но с числом жертв меньшим на порядок. Пропорционально уменьшаются вычислительные затраты.

В развитие работы авторы предполагают

- разработать адаптивный метод расчета окрестности жертвы,
- увеличить число целевых функций,
- найти более адекватные условия окончания итераций.

В практически значимых задачах целевые функции часто имеют высокую вычислительную сложность. Поэтому актуальной является задача разработки и исследования параллельных вариантов алгоритма «хищник-жертва», ориентированных на различные классы параллельных вычислительных систем (системы с общей памятью, кластерные системы, графические процессорные устройства и т.д.) [16]. Предполагается, что решение этой задачи также станет предметом дальнейших исследований авторов.

Работа поддержана РФФИ (проект № 16-07-00287).

Список литературы

1. Kalyanmoy Deb. Multi-objective optimization using evolutionary algorithms. Chichester; N.Y.: Wiley, 2001. 497 p.
2. Карпенко А. П., Митина Е. В., Семенихин А. С. Популяционные методы аппроксимации множества Парето в задаче многокритериальной оптимизации. Обзор // Наука и образование. МГТУ им. Н.Э. Баумана. Электрон. журн. 2012. № 4. С.25. DOI: 10.7463/0412.0363023
3. Laumanns M., Rudolph G., Schwefel H.P. A spatial predator-prey approach to multi-objective optimization: a preliminary study // Parallel problem solving from nature: 5th Intern. conf. on parallel problem solving from nature (Amsterdam, Netherlands, Sept. 27– 30, 1998): proceedings. B.: Springer, 1998. Pp. 241–249. DOI: 10.1007/BFb0056867
4. Xiaodong Li. A real-coded predator-prey genetic algorithm for multiobjective optimization // Evolutionary multi-criterion optimization: 2nd Intern. conf. on evolutionary multi-criterion optimization. EMO-2003 (Faro, Portugal, April 8-11, 2003): proceedings. B.: Springer, 2003. Pp. 207-221. DOI: 10.1007/3-540-36970-815
5. Kalyanmoy Deb, Udaya Bhaskara Rao N. Investigating predator-prey algorithms for multi-objective optimization // KanGAL Report. 2005. No. 2005010. 12 p.
6. Chowdhury S., Dulikravich G. S., Moral R.J. Modified predator-prey algorithm for constrained and unconstrained multi-objective optimisation // Int. J. of Mathematical Modelling and Numerical Optimisation. 2009. Vol. 1. No. 1/2. Pp. 1–38. DOI: 10.1504/IJMMNO.2009.030085

7. Surafel Luleseged Tilahun, Hong Choon Ong. Prey-predator algorithm: A new metaheuristic algorithm for optimization problem // Intern. J. of Information Technology & Decision Making. 2015. Vol. 14. № 6. Pp. 1331 -- 1352. DOI: 10.1142/S021962201450031X
8. Silva A., Neves A., Costa E. An empirical comparison of particle swarm and predator prey optimisation // Artificial intelligence and cognitive science: 13th Irish Intern. conf. AICS 2002 (Limerick, Ireland, Sept. 12-13, 2002): proceedings. B.: Springer, 2002. Pp. 103-110. DOI: 10.1007/3-540-45750-X_13
9. Grimme C., Schmitt K. Inside a predator-prey model for multi-objective optimization: A second study // Genetic and evolutionary computation conf. GECCO'06 (Seattle, Washington, USA, July 8–12, 2006): proceedings. Vol. 1. Pp. 707 -- 714. DOI: 10.1145/1143997.1144121
10. Eiben A. E., Michalewicz Z., Schoenauer M., Smith J. E. Parameter control in evolutionary algorithms // Parameter setting in evolutionary algorithms. B.: N.Y.: Springer, 2007. Pp. 19 -- 46. DOI: 10.1007/978-3-540-69432-82
11. Eshelman L.J., Schaffer J.D. Real-coded genetic algorithms and interval- schemata // Foundations of genetic algorithms - 2: Workshop on foundations of genetic algorithms: FOGA-92 (Vail, Colo., 1992). San Mateo: Morgan Kaufmann, 1993. Pp. 187 -- 202. DOI: 10.1016/B978-0-08-094832-4.50018-0
12. Michalewicz Z. Genetic algorithms + data structures = evolutionary programs. B.; N.Y.: Springer, 1992. 250 p.
13. Соболь И. М., Статников Р. Б. Выбор оптимальных параметров в задачах со многими критериями. 2-е изд. М.: Дрофа, 2006. 175 с.
14. Kalyanmoy Deb, Thiele L., Laumanns M., Zitzler E. Scalable test problems for evolutionary multi-objective optimization // Evolutionary multiobjective optimization. L.: Springer, 2005. Pp. 105-145. DOI: 10.1007/1-84628-137-7_6
15. Zitzler E., Kalyanmoy Deb, Thiele L. Comparison of multiobjective evolutionary algorithms: Empirical results // Evolutionary Computation. 2000. Vol. 8. Iss. 2. Pp. 173-195. DOI: 10.1162/106365600568202
16. Карпенко А.П. Современные алгоритмы поисковой оптимизации. Алгоритмы, вдохновленные природой. М.: Изд-во МГТУ им. Н.Э. Баумана, 2014. 446 с.

Science & Education of the Bauman MSTU
Electronic journal
ISSN 1994-0408

Science and Education of the Bauman MSTU, 2016, no. 10, pp. 154–181.

DOI: **10.7463/1216.0852423**

Received: 09#.11.2016
Revised: 23.11.2016

Research and Setting the Modified Algorithm "Predator-Prey" in the Problem of the Multi-Objective Optimization

.P. Karpenko[1,*], A.V. Pugachev[1]

*apkarpenko@mail.ru

[1]Bauman Moscow State Technical University, Moscow, Russia

Keywords: multi-objective optimization; Pareto-approximation; algorithm "predator-prey"; meta-optimization

We consider a class of algorithms for multi-objective optimization - Pareto-approximation algorithms, which suppose a preliminary building of finite-dimensional approximation of a Pareto set, thereby also a Pareto front of the problem. The article gives an overview of population and non-population algorithms of the Pareto-approximation, identifies their strengths and weaknesses, and presents a canonical algorithm "predator-prey", showing its shortcomings.

We offer a number of modifications of the canonical algorithm "predator-prey" with the aim to overcome the drawbacks of this algorithm, present the results of a broad study of the efficiency of these modifications of the algorithm. The peculiarity of the study is the use of the quality indicators of the Pareto-approximation, which previous publications have not used. In addition, we present the results of the meta-optimization of the modified algorithm, i.e. determining the optimal values of some free parameters of the algorithm.

The study of efficiency of the modified algorithm "predator-prey" has shown that the proposed modifications allow us to improve the following indicators of the basic algorithm: cardinality of a set of the archive solutions, uniformity of archive solutions, and computation time. By and large, the research results have shown that the modified and meta-optimized algorithm enables achieving exactly the same approximation as the basic algorithm, but with the number of preys being one order less. Computational costs are proportionally reduced.

References

1. Deb K. Multi-objective optimization using evolutionary algorithms. Chichester; N.Y.: Wiley, 2001. 497 p.
2. Karpenko A.P., Semenikhin A.S., Mitina E.V. 77-30569/363023. Population methods of Pareto set approximation in multi-objective optimization problem: Review. *Nauka i obrazovanie MGTU im. N.E. Baumana = Science and Education of the Bauman MSTU,* 2012, no. 4. DOI: 10.7463/0412.0363023

3. Laumanns M., Rudolph G., Schwefel H.P. A spatial predator-prey approach to multi-objective optimization: a preliminary study. *Parallel problem solving from nature: 5th Intern. conf. on parallel problem solving from nature* (Amsterdam, Netherlands, Sept. 27 – 30, 1998): proceedings. B.: Springer, 1998. Pp. 241 -- 249. DOI: 10.1007/BFb0056867
4. Xiaodong Li. A real-coded predator-prey genetic algorithm for multiobjective optimization. *Evolutionary multi-criterion optimization: 2nd Intern. conf. on evolutionary multi-criterion optimization. EMO-2003* (Faro, Portugal, April 8-11, 2003): proceedings. B.: Springer, 2003. Pp. 207-221. DOI: 10.1007/3-540-36970-815
5. Kalyanmoy Deb, Udaya Bhaskara Rao N. Investigating predator-prey algorithms for multi-objective optimization. *KanGAL Report.* 2005. No. 2005010. 12 p.
6. Chowdhury S., Dulikravich G. S., Moral R.J. Modified predator-prey algorithm for constrained and unconstrained multi-objective optimization. *Int. J. of Mathematical Modelling and Numerical Optimisation*, 2009, vol. 1, no. 1/2, pp. 1 -- 38. DOI: 10.1504/IJMMNO.2009.030085
7. Surafel Luleseged Tilahun, Hong Choon Ong. Prey-predator algorithm: A new metaheuristic algorithm for optimization problem. *Intern. J. of Information Technology & Decision Making,* 2015, vol. 14, no. 6, pp. 1331 -- 1352. DOI: 10.1142/S021962201450031X
8. Silva A., Neves A., Costa E. An empirical comparison of particle swarm and predator prey optimization. *Artificial intelligence and cognitive science: 13th Irish Intern. conf. AICS 2002* (Limerick, Ireland, Sept. 12-13, 2002): proceedings. B.: Springer, 2002. Pp. 103-110. DOI: 10.1007/3-540-45750-X_13
9. Grimme C., Schmitt K. Inside a predator-prey model for multi-objective optimization: A second study. *Genetic and evolutionary computation conf. GECCO'06* (Seattle, Washington, USA, July 8–12, 2006): proceedings. Vol. 1. Pp. 707 -- 714. DOI: 10.1145/1143997.1144121
10. Eiben A. E., Michalewicz Z., Schoenauer M., Smith J. E. Parameter control in evolutionary algorithms // Parameter setting in evolutionary algorithms. B.: Springer, 2007. Pp. 19 -- 46. DOI: 10.1007/978-3-540-69432-8 2
11. Eshelman L.J., Schaffer J.D. Real-coded genetic algorithms and interval- schemata. *Foundations of genetic algorithms - 2: Workshop on foundations of genetic algorithms: FOGA-92* (Vail, Colo., 1992). San Mateo: Morgan Kaufmann, 1993. Pp. 187 -- 202. DOI: 10.1016/B978-0-08-094832-4.50018-0
12. Michalewicz Z. Genetic algorithms + data structures = evolutionary programs. B.; N.Y.: Springer, 1992. 250 p.
13. Sobol' I.M., Statnikov R.B. *Vybor optimal'nykh parametrov v zadachakh so mnogimi kriteriiami* [The choice of optimal parameters in tasks with many criteria]. 2nd ed. Moscow: Drofa Publ., 2006. 175 p.

14. Kalyanmoy Deb, Thiele L., Laumanns M., Zitzler E. Scalable test problems for evolutionary multi-objective optimization. *Evolutionary multiobjective optimization.* L.: Springer, 2005. Pp. 105-145. DOI: 10.1007/1-84628-137-7_6
15. Zitzler E., Kalyanmoy Deb, Thiele L. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary Computation,* 2000, vol. 8, iss. 2, pp. 173-195. DOI: 10.1162/106365600568202
16. Karpenko A.P. *Sovremennye algoritmy poiskoskovoj optimizatsii* [Modern algorithms of search engine optimization]. Moscow: MSTU im. N.E. Baumana, 2014. 446 p.