УДК 519.6

Распределенная программная система для построения множества Парето в задаче многокритериальной оптимизации динамических систем с использованием параллельного генетического алгоритма

А.П.Карпенко, *МГТУ им. Н.Э. Баумана*
В.А.Овчинников, *ООО «Ладуга»*
А.С.Семенихин, *IBM*

Рассматривается распределенная программная система PRADIS//FRONT для приближенного построения множества Парето в задаче многокритериальной оптимизации динамических систем с использованием параллельных генетических алгоритмов. В качестве среды моделирования динамических систем используется программный комплекс PRADIS.

Ключевые слова: задача многокритериальной оптимизации, множество Парето, фронт Парето, динамическая система, генетический алгоритм, параллельный генетический алгоритм, программный комплекс PRADIS

Distributed program system for construction of Pareto-set in the multiple criteria optimization problem of dynamic systems based on a parallel genetic algorithm

A.P.Karpenko, V.A.Ovchnnikov, A.S.Semenikhin

The paper discusses distributed program system PRADIS//FRONT for approximate construction of Pareto-set in the multiple criteria optimization problem of dynamic systems based on a parallel genetic algorithm. We use program package PRADIS to model of investigated dynamic system.

Key words: multiple criteria optimization problem, Pareto-set, Pareto-front, dynamic system, genetic algorithm, parallel genetic algorithm, program package PRADIS/

Введение

Проектирование сложной технической системы обычно включает в себя этап параметрической оптимизации []. В настоящее время при этом чаще всего используется многокритериальная оптимизация, при которой качество функционирования объекта проектирования определяется некоторым наборов критериев оптимальности.

Методы решения задачи многокритериальной оптимизации чрезвычайно разнообразны. Существует несколько способов классификации этих методов, например, классификация, основанная на содержании и форме использования дополнительной информации о предпочтениях лица, принимающего решения (ЛПР) []. В соответствии с этой классификацией выделяются следующие классы методов решения задачи многокритериальной оптимизации:

- методы, основанные на предварительном построении аппроксимации множества Парето;
- априорные методы;

- апостериорные методы;
- адаптивные методы.

Для пользователя наиболее удобны методы, основанные на непосредственном использовании множества Парето (а, тем самым, и фронта Парето). В этом случае ЛПР выбирает компромиссное решение на фронте Парето не формальными методами, исходя только из своих предпочтений. Основным недостатков методов этого класса, сдерживающим их широкое применение, является их высокая вычислительная сложность.

Известно значительное количество методов и алгоритмов приближенного построения множества Парето (см., например, [,]). Относительно новым и высокоэффективным классом таких методов являются методы на основе генетических алгоритмов.

Генетические алгоритмы принадлежат к классу эволюционных алгоритмов и обладают рядом характеристик, делающих их более предпочтительными, чем классические методы оптимизации []. Так генетические алгоритмы применимы к задачам большой размерности и способны захватить Парето-оптимальные точки даже при однократном запуске алгоритма. Краткий обзор последовательных генетических алгоритмов приближенного построения множества Парето дан в п. работы, а параллельных генетических алгоритмов - в п. 4. В рассматриваемой программной системе PRADIS//FRONT реализован параллельный генетический алгоритм GPGA (Global Parallel Genetic Algorithm) типа master-slave [], построенный на основе модифицированного последовательного метода NPGA (Niched Pareto Genetic Algorithm).

Построение множества Парето в задаче многокритериальной оптимизации динамических систем требует многократного моделирования исследуемой динамической системы при различных значениях варьируемых параметров. Моделирование динамических систем является самостоятельной проблемой, для решения которой разработано значительное количество программных комплексов (см., например, [,,]). В программной системе PRADIS//FRONT моделирование выполняется средствами программного комплекса PRADIS [].

Программный комплекс PRADIS функционирует под управлением операционной системы Windows NT и предназначен для анализа динамических процессов в объектах, описываемых системами обыкновенных дифференциальных уравнений. Можно выделить следующие основные составляющие комплекса PRADIS.

1) Препроцессор Qucs для создания схемы моделируемого объекта.
2) Входные языки PSL, PPL (Python) и трансляторы с этих языков. Входные языки PRADIS предназначены для описания модели исследуемого объекта и задания на ее анализ.
3) Вычислительное ядро, предназначенное для расчета переходных процессов в модели исследуемого объекта в соответствии с заданием на анализ. В вычислительное ядро входит модуль многовариантного анализа, активно используемый программной системой PRADIS//FRONT.
4) Постпроцессор для отображения результатов расчета на экране дисплея. Постпроцессор позволяет отобразить текущие состояния объекта моделирования, графики изменения во времени и текущие числовые значения нескольких выбранных переменных состояния объекта, трехмерную анимацию механической системы и пр.
5) Библиотека моделей элементов для объектов моделирования различной физической природы.
6) Библиотека программ для расчета выходных переменных. Программы этой библиотеки позволяют сформировать (для отображения, как в ходе расчета, так и после него) те переменные, которые характеризуют исследуемый объект, но не входят в число его переменных состояния.

7) Библиотека программ для формирования графических образов. Программы библиотеки обеспечивают визуальное представление элементов исследуемого объекта в постпроцессоре.
8) Библиотека программ для отображения результатов расчетов. Данные программы предназначены для постпроцессорной обработки результатов моделирования.
9) Процедуры управления заданиями, а также сервисные программы.

Многократное моделирование сложной динамической системы требует больших вычислительных ресурсов и в приемлемое для исследователя время может быть выполнено только на параллельных вычислительных машинах. Система PRADIS//FRONT ориентирована на использование в качестве таких машин распределенных ЭВМ, например, вычислительных кластеров [].

Существует большое количество программных средств, обеспечивающих эффективное использования ресурсов распределенной ЭВМ. Программная система PRADIS//FRONT построена на основе коммуникационной библиотеки MPI, точнее – ее версии MPICH []. Данное решение определено следующими факторами:

- большое количество реализаций MPICH, его широкое распространение;
- наличие свободных, регулярно обновляемых имплементаций;
- поддержка операционными системами семейства Windows (программный комплекс PRADIS, как отмечалось выше, может функционировать только в среде Windows).

Распределенная программная система PRADIS//FRONT осуществляет взаимодействие с программным комплексом PRADIS (решателем) в следующих аспектах: генерирование файла задания на языке PSL; запуск решателя на узлах распределенной вычислительной системы; получение результатов математического моделирования. Описание реализационных аспектов системы PRADIS//FRONT приведено в п. . В качестве базовых средств разработки системы были использованы Visual Studio 9.0 Express Edition, MPICH 1.2.5, VMware и программа gnuplot для отображения результатов.

Работоспособность системы проверена на стандартных тестовых задачах многокритериальной оптимизации [] – см. п. . Практическое использование системы рассмотрено на примере приближенного построения множества Парето для задачи многокритериальной оптимизации автомобильной коробки переключения передач – см. п. .

1. Постановка задачи

Совокупность частных критериев оптимальности $\Phi(X) = (\phi_1(X),\phi_2(X),...,\phi_m(X))$ назовем векторным критерием оптимальности. Положим, что ставится задача максимизации каждого из указанных критериев в одной и той же области допустимых значений $D_X \in \Pi \cap D$, где $\Pi = \left\{X \middle| x_i^- \le x_i \le x_i^+, i \in [1,n]\right\}$ - «технологический» параллелепипед, $D = \left\{X \middle| g_1(X) \ge 0, g_2(X) \ge 0,...\right\}$. Здесь $X \subset R^n$, где R^n - n-мерное арифметическое пространство, $g_1(X), g_2(X),...$ - ограничивающие функции. Задача многокритериальной оптимизации записывается в виде

$$\max_{X \in D_X} \Phi(X) = \Phi(X^*). \qquad (1)$$

Запись (1) понимается лишь в том смысле, что для ЛПР желательна максимизация каждого из частных критериев.

Не формально, множество Парето можно определить как множество, в котором значение любого из частных критериев оптимальности можно улучшить (увеличить) только за счет ухудшения (уменьшения) хотя бы одного из остальных критериев. Т.е. любое из решений, принадлежащих множеству Парето, не может быть улучшено одновременно по всем частным критериям оптимальности.

Приведем формальное определение множества Парето. Векторный критерий оптимальности $\Phi(X)$ выполняет отображение множества D_X в некоторое множество D_ϕ пространства критериев, которое называется множеством достижимости. Введем на множестве D_Φ отношение предпочтения. Будем говорить, что вектор $\Phi^1 \in D_\Phi$ предпочтительнее вектора $\Phi^2 \in D_\Phi$ или вектор Φ^1 доминирует вектор Φ^2, и писать $\Phi^1 \succ \Phi^2$, если среди равенств и неравенств $\phi_k(X^1) \geq \phi_k(X^2), k \in [1:\mathrm{m}]$ имеется хотя бы одно строгое неравенство. Выделим из множества D_Φ подмножество точек D_Φ^* (фронт Парето), для которых нет более предпочтительных точек. Множество $D_X^* \in D_X$, соответствующее множеству D_Φ^*, называется множеством Парето. Таким образом, если $X \in D_X^*$, то $\Phi(X) \in D_\Phi^*$.

2. Приближенное построение множества Парето на основе генетических алгоритмов. Основные понятия

Введем в рассмотрение бинарный вектор индивида (генотипа) I и множество всех возможных его значений D_I. В этих обозначениях популяция P определяется как некоторое мультимножество векторов $I \in D_I$. Качество индивида определяется скалярной функцией пригодности $f(I)$, вычисляемой на основе значений частных критериев оптимальности $\phi_1(X), \phi_2(X), ..., \phi_m(X)$. Введем в рассмотрение также функцию $M(I)$, осуществляющую отображение индивида $I \in D_I$ в его фенотип $X = M(I)$. Таким образом, для вычисления функции пригодности некоторого индивида $I \in D_I$ необходимо произвести цепочку вычислений

$$X = M(I) \rightarrow \Phi = \Phi(\mathrm{X}) \rightarrow f = f(\Phi). \qquad (2)$$

Основными операциями генетического алгоритма при приближенном построении множества Парето являются операции скрещивания (кроссовера, рекомбинации), мутации, вычисления пригодности индивида и отбора (селекции), из которых специфическими для задачи построения множества Парето являются только две последние операции [].

Выделяют следующие основные способы селекции в генетических алгоритмах приближенного построения множества Парето:

- селекция по переключающимся частным критериям оптимальности;
- агрегирующая селекция;
- селекция, основанная на понятии Парето-доминирования.

Для того чтобы найти репрезентативную аппроксимацию множества Парето, необходимо обеспечить разнообразие популяций. Эта задача является одной из важнейших в проблематике приближенного построения множества Парето с помощью генетических алгоритмов. Для решения данной задачи могут быть использованы следующие подходы]:

- формирование популяционных ниш (подпопуляций);
- ограниченное скрещивание;
- переопределение;
- перезапуск процесса эволюции;
- уплотнение (объединение).

Наряду с поддержанием разнообразия популяций важнейшую роль в обеспечении эффективности генетических алгоритмах приближенного построения множества Парето играет элитизм – механизм включения лучших индивидов данной популяции в следующую популяцию, т.е. клонирование лучших индивидов.

3. Последовательные методы приближенного построения множества Парето на основе генетических алгоритмов

В настоящее время в вычислительной практике наиболее часто используются четыре метода приближенного построения множества Парето на основе генетических алгоритмов []:

- метод VEGA (Vector Evaluated Genetic Algorithm);
- метод FFGA (Fonseca and Fleming's Multiobjective Genetic Algorithm);
- метод NPGA (Niched Pareto Genetic Algorithm);
- метод SPEA (Strength Pareto Evolutionary Algorithm).

В методе VEGA селекция производится по переключающимся частным критериям оптимальности. Для каждого из m частных критериев создается подпуляция, содержащая N_P / m индивидов, где N_P - размер всей популяции. Индивиды в j-ю подпуляцию отбираются с помощью пропорциональной селекции по критерию $\phi_j(X)$. Далее подпопуляции смешиваются для получения популяции размера N_P, а затем по общей схеме осуществляются скрещивание и мутация.

Метод FFGA использует процедуру ранжирования индивидов, основанную на Парето-доминировании. При этом ранг каждого из индивидов определяется количеством доминирующих его других индивидов данной популяции (так что чем ниже ранг, тем индивид ближе к множеству Парето). Пригодность индивида вычисляется на основе величины, обратной его рангу. Для отбора индивидов в следующее поколение используется процедура турнирной селекции.

В методе NPGA, в отличие от методов VEGA, FFGA, существует механизм поддержания разнообразия популяции. Метод основан на формировании популяционных ниш.

Метод SPEA является самым сложным из числа рассматриваемых методов и, так же, как метод FFGA, использует селекцию, основанную на Парето-доминировании. Для предотвращения преждевременной сходимости, метод использует популяционные ниши. Очень важным свойством метода SPEA является возможность априорного задания количества итоговых точек в искомой аппроксимации множества Парето.

В работе [] на двух-, трех- и четырехкритериальных тестовых задачах исследована эффективность рассмотренных методов приближенного построения множества Парето. Эффективность оценивалась по трем критериям, формализующим равномерность распределения получаемых решений, степень покрытия множества Парето, а также количество доминируемых решений в итоговой популяции. По совокупности указанных критериев лучшим назван метод SPEA.

4. Параллельные методы приближенного построения множества Парето на основе генетических алгоритмов

Существует два подхода к проектированию параллельных генетических алгоритмов - однопопуляционный и многопопуляционный подходы.

В однопопуляционном подходе параллельно вычисляются только функции пригодности, а генетические операции выполняются централизованно на host-процессоре. Такой подход, очевидно, ориентирован на master-slave парадигму параллельного программирования [].

С точки зрения параллелизации более естественным является, конечно, многопопуляционный подход. Суть этого подхода состоит в разделении главной популяции на части и реализации каждым из процессоров собственного генетического алгоритма над той частью популяции, которая хранится в его памяти. Для повышения эффективности поиска процессоры могут обмениваться между собой генетическим материалом.

Существует большое разнообразие параллельных генетических алгоритмов, ориентированных на различные классы параллельных вычислительных систем и на различные критерии качества алгоритмов. Один из известных вариантов классификации параллельных генетических алгоритмов (PGA) приведен на Рис. 1 [].

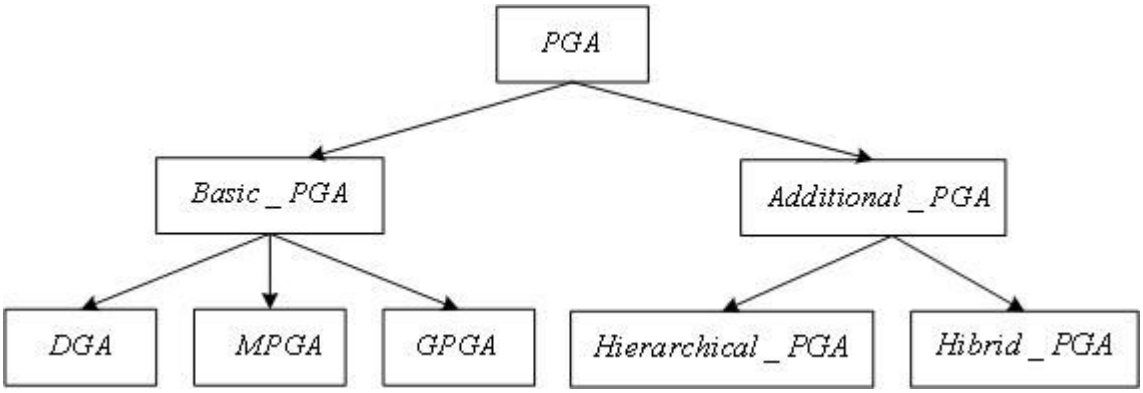

Рис. 1. Классификация параллельных генетических алгоритмов

GPGA (Global Parallel Genetic Algorithm) представляет собой однопопуляционный алгоритм. Host-процессор содержит всё поколение в своей памяти и выполняет над ним операции селекции, кроссовера и мутации. Функции пригодности индивидов вычисляются на slave-процессорах. При высокой вычислительной сложности функций пригодности основной проблемой при реализации данного алгоритма является проблема балансировки загрузки slave-процессоров. Очевидно, что рассматриваемый алгоритм ориентирован на MIMD-вычислительные системы [].

DGA (Distributed Genetic Algorithm) – это многопопуляционный алгоритм, также ориентированный на MIMD-вычислительные системы. В этом случае каждый процессор запускает свой собственный генетический алгоритм на выделенной ему части всей популяции (подпопуляции). Различают два класса распределенных генетических алгоритмов - с наличием механизма обмена подпопуляций индивидами и без этого механизма. В общем случае, при использовании DGA каждый процессор начинает вычисления с разными параметрами генетического алгоритма и с разными наборами генетических операторов. Как следствие, в процессе вычислений подпопуляции будут развиваться в различных направлениях, что гарантирует большое генетическое разнообразие. Отметим, что миграционные процессы между подпопуляции требуют очень аккуратной настройки: при большом количестве индивидов, подлежащих обмену, и высокой частоте обменов генетическое разнообразие подпопуляций нивелируется; наоборот, при малом объеме обменов и низкой их частоте может иметь место преждевременная сходимость внутри подпопуляций.

MPGA (Massively Parallel Genetic Algorithm) называют также Cellular Algorithm (клеточный алгоритм). Данный алгоритм ориентирован на вычислительные машины класса SIMD []. В этом случае каждым процессорным элементом в один момент времени обрабатывается один индивид. Индивиды выбирают пару и рекомбинируют со своими непосредственными соседями (северный, южный, восточный, западный).

Hierarchical PGA представляют собой иерархические алгоритмы, в которых на разных уровнях иерархии реализованы разные генетические алгоритмы (рассмотренные выше и другие). Обычно используются 2-х уровневые иерархии: DGA-модель - на верхнем уровне; GPGA- или MPGA-модель – на нижнем уровне.

Hybrid PGA – класс алгоритмов, использующих сочетание параллельных генетических алгоритмов и классических оптимизационных методов.

5. Используемые в работе методы и алгоритмы приближенного построения множества Парето

В работе в качестве базового метода выбран последовательный метод NPGA. Это объясняется теми обстоятельствами, что метод NPGA существенно более прост в реализации, чем метод SPEA, и, в то же время, обеспечивает достаточно высокую эффективность []. Использована модификация метода, предложенная в работе [] и заключающаяся в следующем:

- использование аутбридинга (дальнеродственного скрещивания) вместо турнирной селекции [];
- использование метода ранжирования индивидов из метода FFGA;
- использование операции клонирования.

Пусть P_t - текущая популяция, P' - промежуточная популяция, $P_{dom} \subseteq P_t$ - сравнительное множество, σ_{share} - заданный радиус популяционной ниши, N_{dom} - заданное количество индивидов в сравнительном множестве (давление доминирования), $d(I_1, I_2)$ - расстояние между индивидами I_1, I_2 популяции P_t в некоторой метрике.

Схему вычисления пригодности индивида и селекции в методе NPGA можно представить в следующем виде [].

Шаг 1. Положим $s = 1$, $P' = \varnothing$, $P_{dom} = \varnothing$.

Шаг 2. Из t_{dom} случайно выбранных индивидов популяции P_t формируем сравнительное множество P_{dom}.

Шаг 3. Из оставшейся части популяции P_t случайно выбираем два индивида I_1, I_2.

Шаг 4. Если $X_1 = M(I_1)$ недоминируем ни одним из векторов $X_k = M(I_k)$, $I_k \in P_{dom}$, а $X_2 = M(I_2)$ доминируется хотя бы одним вектором сравнительного множества P_{dom}, то победителем турнира считаем индивида I_1: $P' = P' + I_1$.

Шаг 5. Если $X_2 = M(I_2)$ недоминируем ни одним из векторов $X_k = M(I_k)$, $I_k \in P_{dom}$, а $X_1 = M(I_1)$ доминируется хотя бы одним вектором сравнительного множества P_{dom}, то победителем турнира считаем индивида I_2: $P' = P' + I_2$.

Шаг 6. Если победитель турнира не определен, то выполняем следующие действия.

а) Вычисляем количество индивидов в промежуточной популяции P', которые находятся от индивида I_1 на расстоянии, не превышающем радиус ниши σ_{share}:

$$n(I_1) = \{I_k \mid I_k \in P' \wedge d(I_1, I_k) < \sigma_{share}\}. \qquad (3)$$

б) Аналогично вычисляем количество индивидов $n(I_2)$ в промежуточной популяции P', которые находятся от индивида I_2 на расстоянии, не превышающем радиус ниши σ_{share}.

в) Если $n(I_1) < n(I_2)$, то полагаем $P' = P' + I_1$. Иначе - $P' = P' + I_2$.

Шаг 7. Полагаем $s = s + 1$. Если $s \le N_P$, то переходим к шагу 3, иначе – заканчиваем вычисления.

Далее по общей схеме генетического алгоритма выполняется скрещивание индивидов промежуточной популяции P' и формируется популяция P''. Затем реализуется мутация индивидов популяции P'', на основе которой получается популяция $P''' = P_{t+1}$.

Как отмечалось выше, вместо рассмотренной турнирной селекции мы используем ранжирование индивидов на основе Парето-доминирования []. По определению ранга, индивиду, для которого ни один из индивидов текущей популяции не лучше него по всем

частным критериям оптимальности, присваивается ранг 1. Ранг остальных индивидов определяется по формуле

$$rank(I_k) = 1 + a_k,$$

где a_k - количество индивидов текущей популяции, лучших по всем частным критериям оптимальности.

Используемые механизмы формирования индивидов обеспечивают выполнение условия $X = M(I) \in \Pi$, но не обеспечивают, в общем случае, выполнение условия

$$X = M(I) \in D. \quad (4)$$

Ранг индивидов, которые нарушают ограничение (4) назначается в зависимости от того, в какой мере эти ограничения нарушены. Ранг любого из индивидов из числа тех, для которых ограничение (4) нарушено, выше ранга любого из индивидов, для которых это ограничение выполнено.

Функция пригодности строится на основе функции

$$\varphi(I) = 1 + \sum_{k=1}^{rank(I)-1} \eta(k), \quad (5)$$

где $\eta(k)$ - число индивидов ранга *k*. Точнее говоря, в качестве функции пригодности используется функция

$$f(I_1) = \mu(I_1)\varphi(I_1), \quad (6)$$

где $\mu(I_1)$ - нишевое число индивида I_1, вычисляемое по формуле

$$\mu(I_1) = \sum_{I_k \in P_t} Sh(d(I_1, I_k)). \quad (7)$$

Здесь $Sh(d)$ - функция разделения:

$$Sh(d) = \begin{cases} 1 - d/\sigma_{share}, d < \sigma_{share}, \\ 0, d \geq \sigma_{share}. \end{cases} \quad (8)$$

В системе PRADIS//FRONT индивиды для скрещивания (индивиды-родители) выбираются по следующему правилу. Первый индивид выбирается случайно (из промежуточной популяции P') вне зависимости от значения его функции пригодности. Второй индивид выбирается из той же популяции на некотором расстоянии от данного индивида (исходя из принципов поддержания разнообразия популяции). В случае если нельзя найти второго индивида на указанном расстоянии, в качестве этого индивида из множества P' выбирается случайный индивид.

Важнейшим оператором в любом генетическом алгоритме является оператор скрещивания. Существует множество разновидностей этого оператора: одноточечный, двухточечный, равномерный и пр. []. В системе PRADIS//FRONT скрещивание выполняется по следующему правилу: выбирается ведущий родитель; к значениям его генов прибавляется разность между соответствующими генами родителей, умноженная на коэффициент скрещивания (CrossoverRate); индивид, полученный в результате скрещивания, записывается во множество P''.

Известно множество операторов мутации индивидов, отличающихся количеством индивидов, подверженных мутации, алгоритмами мутации, количеством генов, участвующих в мутации, и пр. []. В системе PARET//FRONT используется оператор мутации, основанный на стандартном операторе, но со следующими отличиями.

- Чтобы усилить эффект от выбранного правила скрещивания, все произведенное потомство подвергается мутации в одном гене хромосомы.
- Ген модифицируется с помощью стохастического параметра и параметра мутации – MutationRate.

Нами была сделана попытка использовать также классическую схему оператора мутации – инвертирование заданного количества битов в хромосоме. Однако

тестирование показало, что такая схема обеспечивает существенно худшие результаты, чем схема, рассмотренная выше.

Распараллеливание вычислений в работе реализовано по схеме алгоритма GPGA. Основания для такого выбора следующие.

1) Ориентация работы на распределенные вычислительные системы. В результате из рассмотрения исключаются алгоритмы, ориентированные на SIMD-системы, а также алгоритмы, требующие большого количества коммуникаций между процессорами.

2) Высокая вычислительная сложность функции пригодности индивидов, обусловленная тем, что каждое вычисление такой функции требует интегрирования соответствующей системы обыкновенных дифференциальных уравнений и вычисления значений всех частных критериев оптимальности.

3) Простота реализации модели параллельных вычислений master-slave.

Для балансировки загрузки процессоров используется равномерная статическая балансировка [], при которой каждый slave-процессоров обрабатывает одинаковое количество индивидов, назначаемых ему host-процессором.

6. **Организация программной системы PRADIS//FRONT**

Программная система PRADIS//FRONT состоит из серверной и клиентской частей. Серверная часть системы реализует следующие основные функции:

- инициализация приложения;
- поддержка файлов, с помощью которых осуществляется обмен данными с клиентской частью;
- поддержка очереди заданий для клиентов;
- синхронизация работы;
- выдача клиентам заданий для расчета;
- прием от клиентов результатов расчета;
- реализация генетического алгоритма;
- формирование файла результата.

Клиентская часть системы реализует следующие функции:

- поддержка работы программного комплекса PRADIS;
- запуск PRADIS для расчета;
- получение от PRADIS результатов расчета;
- прием от серверной части заданий на расчет;
- передача серверной части результатов расчета.

В системе PRADIS//FRONT определено два типа сообщений. Сообщения, посылаемые сервером и принимаемые клиентами, имеют формат

<Номер задачи для расчета>,

а сообщения, посылаемые клиентами и принимаемые сервером, - формат

<Номер рассчитанной задачи> + <ID процессора> + <Результаты расчета>.

Для синхронизации серверной и клиентских частей перед началом обмена сообщениями используется барьерная синхронизация (с помощью вызова MPI_Barrier (MPI_COMM_WORLD)).

6.1. Серверная часть системы. Сервер назначает задания клиентам по одному. Благодаря этому, при сбое в работе одного из клиентов пропадает лишь то задание, которое данный клиент обрабатывал в момент сбоя.

Для формирования заданий, сервер использует шаблон задания, в котором варьируемые параметры заменены специальными символами. Формирование заданий в серверной части системы выполняет функция replaceFiles(). Функция формирует необходимое количество заданий, сохраняя каждое из них в отдельном файле. Имена файлов формируются следующим образом: к имени файла, в котором находится шаблон задания, добавляется постфикс «_multi_» и <номер задания>. Если, например, имя файла

шаблона есть test_multi.psl, то идентификатор файла задания номер 97 выглядит как test_multi_multi_97.psl.

После того как все файлы с заданиями сформированы, сервер начинает выполнять только диспетчерские функции: раздает номера заданий клиентам; получает от них результаты расчетов и сохраняет их в буфере FunctionValues.

После завершения обработки клиентами всех заданий и приема сервером всех результатов обработки, сервер с помощью функции SetMainObject осуществляет обработку данных из буфера FunctionValues: отбрасывает номера заданий и ID процессоров, упорядочивает результаты расчета в соответствии с номерами заданий и записывает результаты в массив funcSets, представляющий собой поле класса FunctionStudy. Затем сервер реализует одну итерацию генетического алгоритма.

6.2. Клиентская часть системы. Основными функциями клиентской части системы являются обмен данными с серверной часть и взаимодействие с программным комплексом PRADIS. Как указывалось выше, задания на расчет решателю, записанные на одном из его внутренних языков, готовит серверная часть системы, а клиентской части сообщается только номер задания.

В памяти каждого клиента содержится объект многовариантного анализа FunctionStudyMPI, у которого есть поля параметров и поля результатов расчета.

Запуск PRADIS осуществляет функция системы RunSingle(int n), где *n* – номер задания для расчета. В первую очередь функция RunSingle(…) формирует имя соответствующего файла задания. Сформированное имя файла служит аргументом для вызова функции решателя solver->Run(filename).

Программный комплекс PRADIS дает возможность использовать задания, написанные на языках PSL, PPL (Python), а также задания в формате схем предпроцессора Qucs. Для каждого из способов требуется запускать свою часть решателя. Функция solver->Run(filename) автоматически запускает требуемый решатель (в зависимости от расширения имени файла, указанного в ее аргументе). В настоящей версии программной системы PRADIS//FRONT реализован запуск заданий только на языке PSL.

Результаты своей работы PRADIS записывает в выходной файл. Для примера, рассмотренного в п. 6.1, этот файл имеет имя test_multi_multi_97.psl.dat.

7. Тестирование алгоритма приближенного построения множества Парето

Работоспособность алгоритма, рассмотренного в п. 5, проверялась на тестовых задачах ZDT3, ZDT6, DTLZ4 из стандартного набора тестов для непрерывных многокритериальных задач [] – см. Табл. 1. При решении всех тестовых задач использовались следующие значения параметров алгоритма:

- параметр разделения σ_{share}=0.01;
- доля отбираемых для скрещивания индивидов T_r=0.3;
- параметр рекомбинации CrossoverRate=0.7;
- мутационный параметр MutationRate=1.0;
- число индивидов $N_P = 1000$;
- число поколений $N_{gen} = 1000$.

Табл. 1. Тестовые задачи.

Задача	**Частные критерии оптимальности**	**Размерность и множество допустимых значений *D***
ZDT_3	$\phi_1(X) = x_1$ $\phi_2(X) = g\left[1 - \sqrt{\frac{\phi_1(X)}{g}} - \frac{\phi_1(X)}{g}\sin(10\pi\phi_1(X))\right]$	n=30 $0 \le x_i \le 1$

	$g(x_2,...,x_n) = 1 + 9\sum_{i=2}^{n} \frac{x_i}{n-1}$	
ZDT_6	$\phi_1(X) = 1 - \exp(-4x_1)\sin^6(6\pi x_1)$ $\phi_2(X) = g\left[1 - \left(\frac{\phi_1(X)}{g}\right)^2\right]$ $g(x_2,...,x_n) = 1 + 9\left(\sum_{i=2}^{n} \frac{x_i}{n-1}\right)^{0.25}$	n=10 $0 \le x_i \le 1$
$DTLZ_4$	$\phi_1(X) = (1+g)\cos\left(\frac{\pi x_1^\alpha}{2}\right)\cos\left(\frac{\pi x_2^\alpha}{2}\right)$ $\phi_2(X) = (1+g)\cos\left(\frac{\pi x_1^\alpha}{2}\right)\sin\left(\frac{\pi x_2^\alpha}{2}\right)$ $\phi_3(X) = (1+g)\sin\left(\frac{\pi x_1^\alpha}{2}\right)$ $g(x_1,...,x_n) = \sum_{i=3}^{n} (x_i - 0.5)^2$, $\alpha = 1$	n=12 $0 \le x_i \le 1$

Рис. 2 - 4, иллюстрирующие результаты тестирования, получены с помощью свободно распространяемой утилиты gnuplot. На рисунках представлены точные фронты Парето (сплошные линии) и их аппроксимации, полученные с помощью программной системы PRADIS//FRONT (обозначены крестиками). Количество поколений, при которых получены приведенные на рисунках результаты, изменяется от примерно 300 до 1000.

Результаты тестирования показывают, что используемый в системе алгоритм приближенного построения множества Парето позволяет получить хорошие решения для тестовых задач ZDT3, ZDT6 и удовлетворительные результаты – для задачи $DTLZ_4$.

8. Приближенное построение множества Парето для задачи многокритериальной оптимизации автомобильной коробки переключения передач

Программная система PRADIS//FRONT использована для построения фронта Парето в задаче многокритериальной оптимизации автомобильной коробки переключения передач. Рассматривается участок разгона заднеприводного автомобиля массой 1500 кг. с пятиступенчатой коробкой передач при заданной характеристике двигателя. Разгон осуществляется из состояния покоя в течение 40 секунд. В начальный момент времени, полагается, включена первая передача. При движении автомобиля учитывается сопротивление воздуха, трение качения, инерционные и упругие характеристики трансмиссии. Общая структура модели автомобиля представлена на Рис. 5.

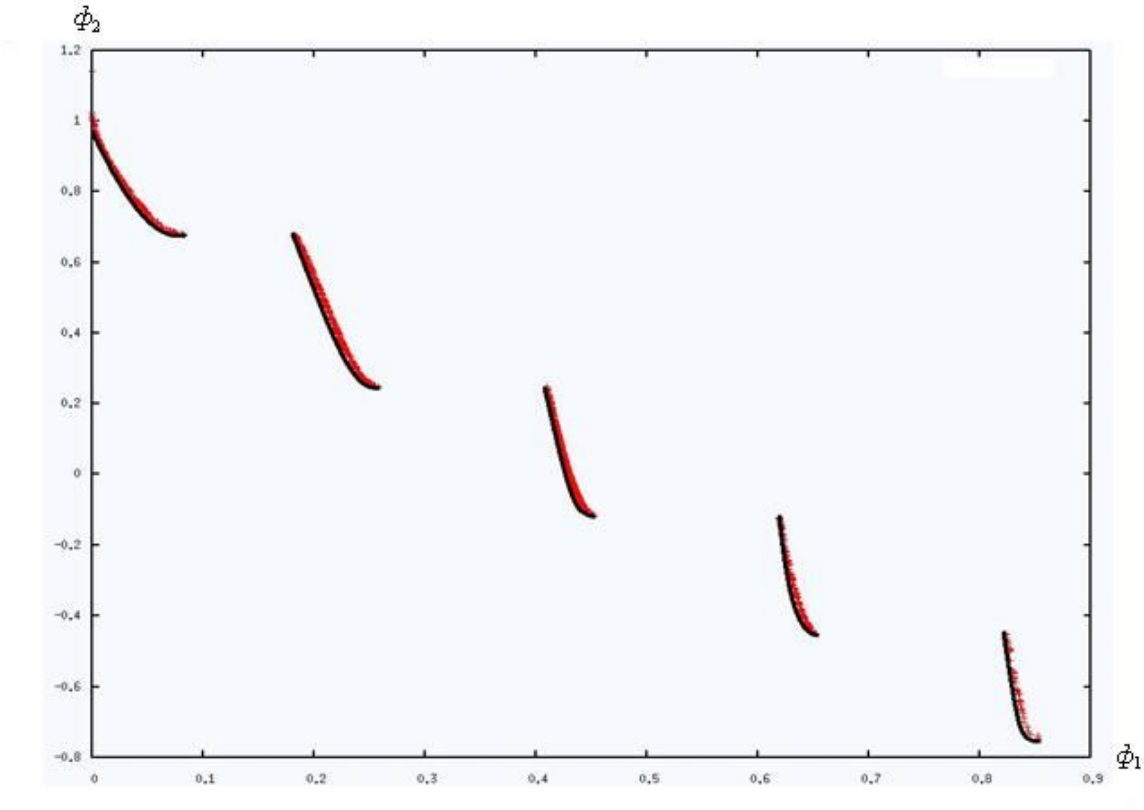

Рис. 2. Результаты тестирования для задачи ZDT3

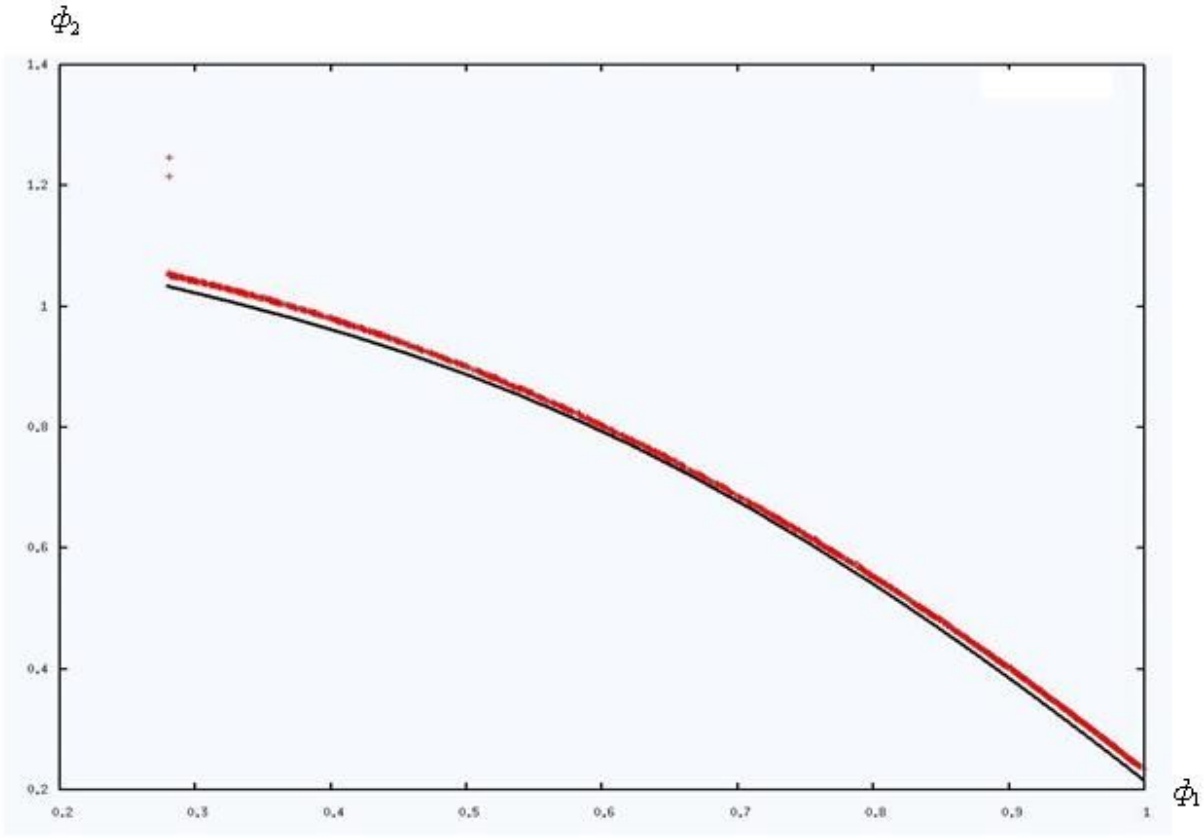

Рис. 3. Результаты тестирования для задачи ZDT6

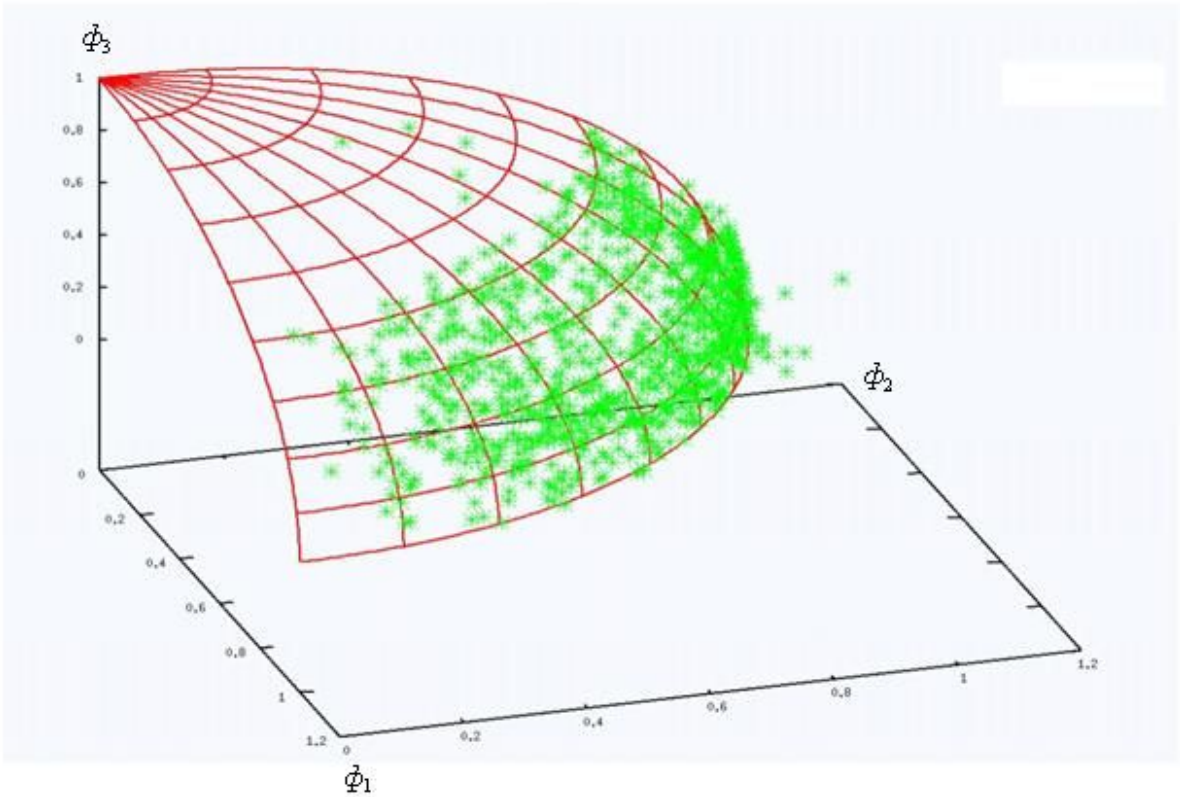

Рис. 4. Результаты тестирования для задачи DTLZ4

Рис. 5. Структурная схема модели

Модель автомобиля в виде схемы редактора Qucs системы PRADIS приведена на Рис. 6, модель на языке PSL - в Приложении 1. Основными элементами модели являются: двигатель DVTBL1; муфта MUFTA1; коробка переключения передач MGEAR11; дифференциал DIFMC1; колеса MUNL1, MUNL2. Кроме того, на схеме присутствует большое количество условных обозначений измерительных приборов, необходимые для расчета вектора выходных параметров. Характеристика двигателя (зависимость момента от частоты вращения вала) задана таблично.

Варьируемыми параметрами модели являются следующие 10 параметров ($n = 10$):

- N1, N2, N3, N4, N5 - передаточные отношения для всех 5 передач;

- T1 – время сброса сцепления;
- T2, T3, T4, T5 – времена включения 2-й, 3-й и т.д. передач.

Параллелепипед Π допустимых значений вектора варьируемых параметров определяется неравенствами

$$1 \le N_i \le 20,\ N_i \in Z,\ i \in [1:5],$$
$$0 \le T_1 \le 1,\ T_1 \in R^1,$$
$$1 \le T_i \le 20,\ T_i \in R^1,\ i \in [2:5],$$

где Z – множество натуральных чисел, R^1 - пространство вещественных чисел.

Множество допустимых значений D формируется неравенствами $T_2 < T_3 < T_4 < T_5$.

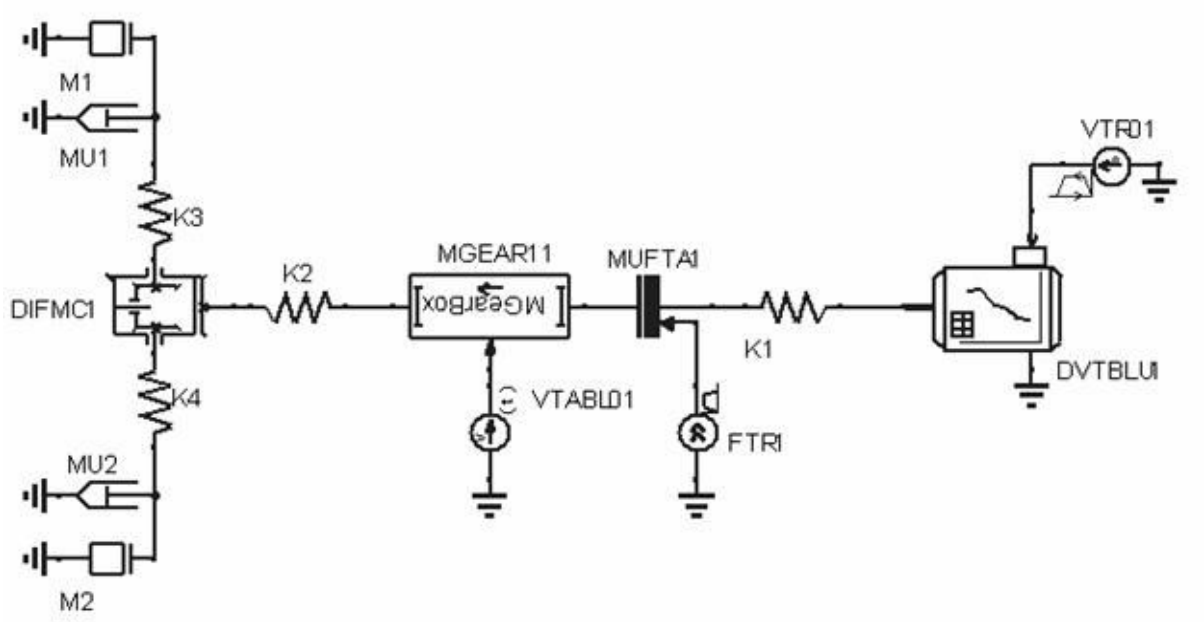

Рис. 6. Схема модели в редакторе Qucs.

Выходные переменные модели:

- Time60 – время разгона до 60 км/ч;
- Time100 – время разгона до 100 км/ч;
- Velocity – скорость автомобиля;
- MaxTorque – максимальный момент в трансмиссии;
- Torque – момент в трансмиссии;
- Torque_Engine – момент на вале двигателя;
- V_Engine – частота вращения вала двигателя;
- Out_Gear – частота вращения выходного вала коробки передач;
- Efficiency – к.п.д. трансмиссии.

На основе указанных выходных параметров модели сформировано 4 следующих критерия оптимальности:

- максимальный момент в трансмиссии – MaxTorque (полежит минимизации);
- максимальная скорость автомобиля - VelocityMax (полежит максимизации);
- время разгона до 100 км/ч - Time100 (подлежит минимизации);
- эластичность автомобиля (время разгона с 60 до 100 км/ч) - Time60_Time100 (подлежит минимизации).

С помощью программной системы PRADIS//FRONT получены аппроксимации множества Парето для 3 следующих задач многокритериальной оптимизации:

1) двухкритериальная задача (ϕ_1=MaxTorque, ϕ_2=VelocityMax);

2) трехкритериальная задача (ϕ_1=MaxTorque, ϕ_2=VelocityMax, ϕ_3=Time100);
3) четырехкритериальная задача задача (ϕ_1=MaxTorque, ϕ_2=VelocityMax, ϕ_3 =Time100, ϕ_4 = Time60_Time100).

Результаты расчетов для двухкритериальной и трехкритериальной задач приведены на Рис. 7, 8, соответственно; результаты расчетов для четырехкритериальной задачи – в приложении 2.

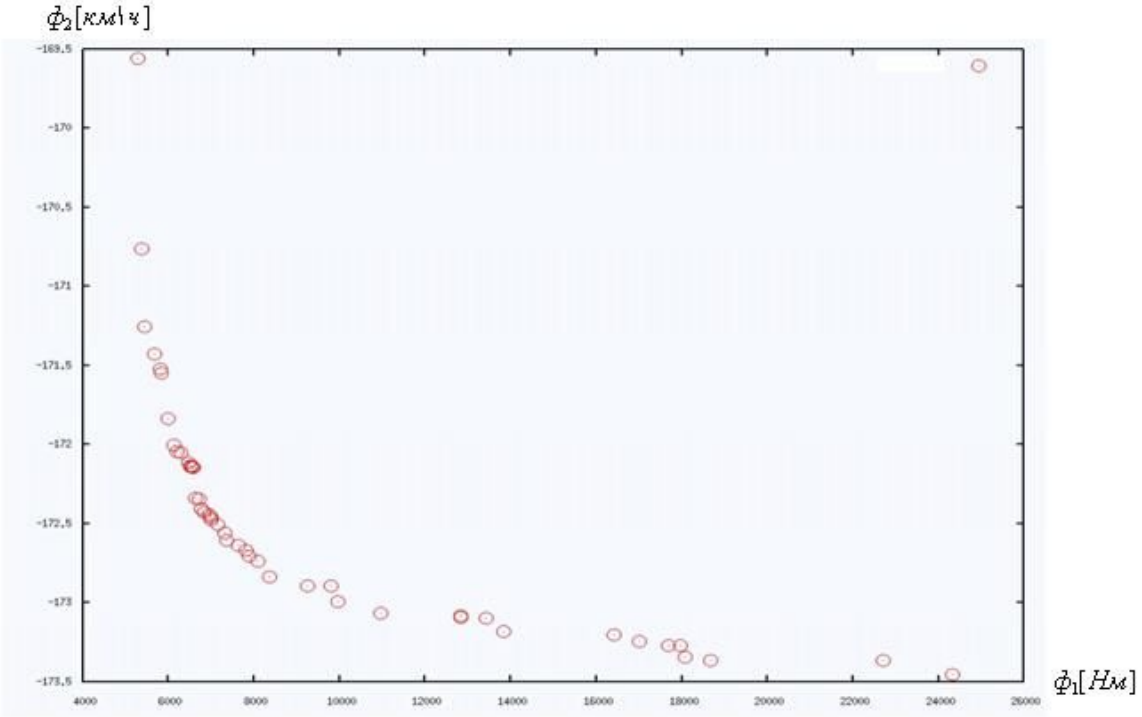

Рис. 7. Аппроксимация фронта Парето для двухкритериальной задачи

Заключение

Работа впервые позволила использовать программный комплекс PRADIS в параллельном режиме.

Результаты работы показывают, что программная система PRADIS//FRONT является удобным и надежным средством решения задач многокритериальной оптимизации сложных динамических систем на основе построения соответствующих множеств Парето.

В развитие работы планируется реализовать запуск заданий на распределенной вычислительной системе не только на языке PSL, но также на языке PPL и в формате схем предпроцессора Qucs. Кроме того, в развитие работы планируется реализация многопоточной серверной части приложения. Идея состоит в том, чтобы организовать дополнительные потоки, которые будут осуществлять обмен сообщениями с разными клиентами одновременно. Планируется также использовать другие методы балансировки загрузки узлов распределенной вычислительной системы [].

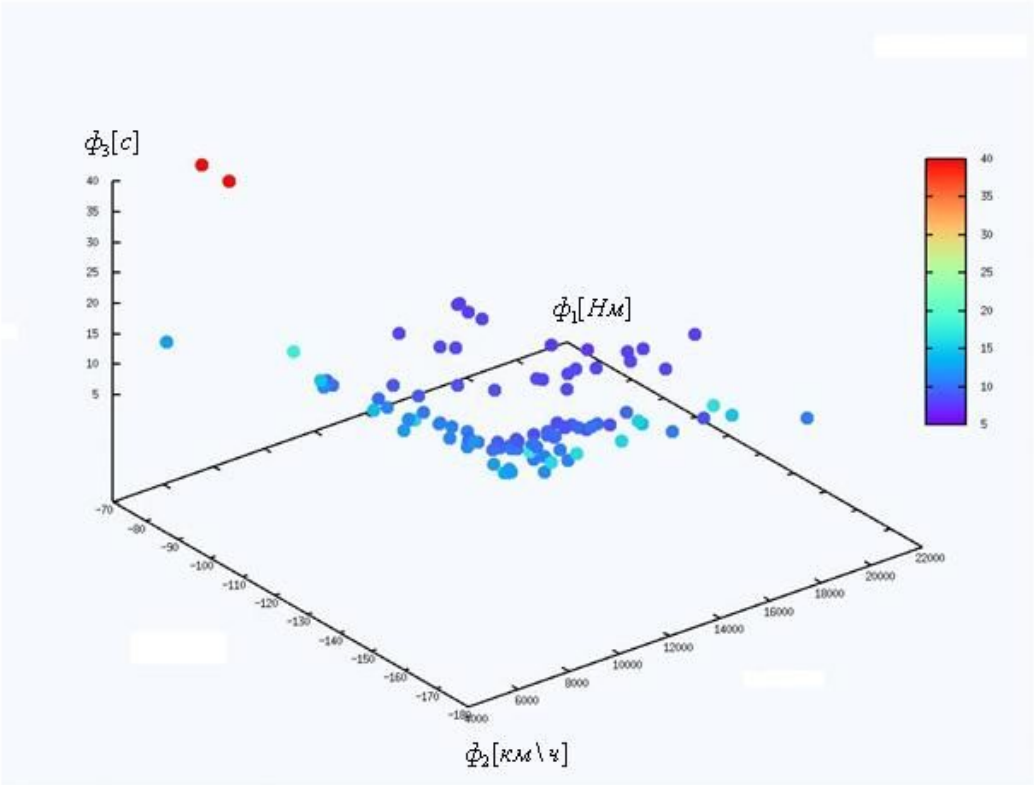

Рис. 8. Аппроксимация фронта Парето для трехкритериальной задачи

Литература

1. Норенков И.П. Основы автоматизированного проектирования: Учеб. для вузов. – М.: Изд-во МГТУ им. Н.Э. Баумана, 2006. – 448 с.
2. Лотов А.В. Введение в экономико-математическое моделирование. – М.: Наука, 1984. -392 с.
3. Соболь И.М., Статников Р.Б. Выбор оптимальных параметров в задачах со многими критериями.- М.: Дрофа, 2006. -175 с.
4. Подиновский В.В., Ногин В.Д. Парето-оптимальные решения многокритериальных задач. –М.: ФИЗМАТЛИТ, 2007. -256 с.
5. Гуменникова А.П. Адаптивные поисковые алгоритмы для решения сложных задач многокритериальной оптимизации: Дис. … канд. техн. наук. - Красноярск, 2006. – 129 с.
6. Cantú-Paz, E., *A Survey of Parallel Genetic Algorithms*, Calculateurs Paralleles, Vol. 10, No. 2. Paris: Hermes, 1998., available via ftp from: ftp://ftp-illigal.ge.uiuc.edu/pub/papers/Publications/ cantupaz/survey.ps.Z.
7. MSC.ADAMS - виртуальное моделирование машин и механизмов. - http://www.mscsoftware.ru/
8. Трудоношин В.А., Трудоношин И.В. Моделирование электромеханических систем с помощью программно-методического комплекса «ПА9»// Информационные технологии, 2006, № 4, с.
9. Погорелов Д.Ю. Компьютерное моделирование динамики технических систем с использованием программного комплекса "Универсальный механизм".- http://www.umlab.ru
10. PRADIS – Руководство к программе. - ttp://www.laduga.ru
11. Воеводин В.В., Воеводин Вл.В. Параллельные вычисления. –Спб.: БХВ-Петербург, 2004. -608 с.

12. Лобарева И.Ф., Черный С.Г., Чирков Д.В., Скороспелов В.А., Турук П.А. Многоцелевая оптимизация формы лопасти гидротурбины. –Вычислительные технологии, Том 11, №5, 2006. С.63 – 76.
13. Филиппов С.Ж. Параметрическая идентификация систем поддержки принятия решений на основе параллельных генетических алгоритмов: Дис. ... канд. техн. наук. -Санкт-Петербург, 2003. - 152 с.
14. Карпенко А.П., Пупков К.А. Моделирование динамических систем на транспьютерных сетях. -М.: Биоинформ, 1995.-73 с.
15. Карпенко А.П., Федорук В.Г., Федорук Е.В. Исследование эффективности балансировки загрузки многопроцессорной системы при распараллеливании одного класса вычислительных задач //"Наука и образование: электронное научное издание. Инженерное образование", № гос. регистрации 0420700025 www.technomag.edu.ru, август, 2007, №0420700025/0034.

Приложение 1. Описание параметрической модели трансмиссии автомобиля на языке PSL

```
$ DATA: tr7_MULTI
Var.Gear1 = {N1}, 0.95, 1000, 1000000.0, 0.001, 0.001
Var.Gear2 = {N2}, 0.9, 1000, 1000000.0, 0.001, 0.001
Var.Gear3 = {N3}, 0.9, 1000, 1000000.0, 0.001, 0.001
Var.Gear4 = {N4}, 0.9, 1000, 1000000.0, 0.001, 0.001
Var.Gear5 = {N5}, 0.9, 1000, 1000000.0, 0.001, 0.001
Var.t2 = {T2}
Var.t3 = {T3}
Var.t4 = {T4}
Var.t5 = {T5}
Var.t1 = {T1}

 $ FRAGMENT: tr7_MULTI
#BASE: 19, 20
#STRUCTURE:
K3'K(1,2;1e+06)
K4'K(8,12;1e+06)
MGEAR11'MGEAR1(13,14,15;
  Var.Gear1, Var.Gear2, Var.Gear3, Var.Gear4, Var.Gear5)
K1'K(16,17;1e+06)
Der'MATH6(3,4;1)
M2'C(12,19;  300)
DIFMC1'DIFMC(18,2,8;  0.1, 1, 1000, 1e+06, 0.0001, 0.0001, 0.0001)
VTR01'VTR0(5,20;  0, 1, 0, 0.01, 100, 1, 1e+06)
Mul'MATH4(12,3;  -1.44)
MUFTA1'MUFTA(16,13,6;  1, 1e+06, 1e+06, 0.01, 1, 10, 0.0001, 0.0001, 0.1)
MU2'MU(12,19;  0.01)
MUNL2'MUNL(20,12;  0.08)
VTABL01'VTABL0(15,20;
  1e+06, 0, 0, Var.t2, 2, Var.t3, 3, Var.t4, 4, Var.t5, 5)
FTR1'FTR(6,20;  0, 1000, Var.t1, 0.01, 100, 1)
DVTBLU1'DVTBLU(17,20,5;
0.0001, 1, -1, 0, -0.0001, 0, 0, 500, 105, 800, 150, 900, 210,    900,420, 800, 630, 500, 700, 0,
800, 0)
K2'K(7,18;1e+06)
```

```
M1'C(1,19;300)
MU1'MU(1,19;0.01)
MUNL1'MUNL(20,1;0.08)
F2Signal1'TSIG1(7,14,9;1)
Abs1'MATH7(9,10;1)
MaxT1'MAXT(10,11;1)

#EXTERNAL:
#OUTPUT:
V_Engine'V(17;1)
V_Clutch'V(16;1)
V_Step'V(15;1)
In_Gear'V(13;1)
Out_Gear'V(14;1)
Velocity'V(3;1)
Time100'TIMERA(4;100)
Time60'TIMERA(4;60)
Torque'V(I:K2(1);1)
Torque_Engine'V(I:DVTBLU1(1);1)
Nin'N(17,I:DVTBLU1(1);1)
In_Diff'V(18;1)
Nout'N(1,I:K3(1);1)
Efficiency'EFFIC(17,I:DVTBLU1(1),1,I:K3(1);0, 2)
Out_Gear1'V(14;1)
MaxTorque'V(11;1)
#MAP
 $ SHOW:
 $ RUN:
Dynamic1'SHTERM(SAVE=1e+10, END=40, OUT=0, SMAX=0.1, SMIN=1e-09,
   DRLTI=0.01, DABSI=0.1, DRLTU=0.01, DABSU=0.01, DRLTX=0.001,
   DABSX=0.01, FLAG=2, ITR=5, PREDICT=1, DEBUG=0, OPTIM=3,
   CONTROL=0.01, SCHEME=0, WEIGHT=1, SECOND=0, IGNORE=0, ATM=0,
   CHECKM=0, OUTPER=1, OUTVAR=1, PRTTIME=30)

 $ PRINT:
DISP1'DISP(  START=0, END=1, FROM=0, SCALE=0;Out_Gear = (-1,1),
  In_Gear = (-1,1),V_Engine = (-1,1))
DISP2'DISP(  START=0, END=1, FROM=0, SCALE=0;Velocity = (-1,1))
DISP3'DISP(  START=0, END=1, FROM=0, SCALE=0;Time100 = (-1,1),
  Time60 = (-1,1))
DISP4'DISP(  START=0, END=1, FROM=0, SCALE=0;Torque_Engine = (-1,1),
  Torque = (-1,1))
DISP5'DISP(  START=0, END=1, FROM=0, SCALE=0;Nin = (-1,1),
  Nout = (-1,1))

$END
```

Приложение 2. Фронт Парето для четырехкритериальной задачи

$\phi_1[Нм]$	$\phi_2[км \setminus ч]$	$\phi_3[с]$	$\phi_4[с]$
11756.109375	-166.209230	7.596370	3.637967
16719.480469	-170.377962	7.957443	3.661078

13848.692383	-171.506975	7.986857	3.878675
18383.888672	-167.778259	7.095534	3.644868
12064.768555	-170.253357	8.701928	3.679001
15854.546875	-171.405472	7.921792	3.730051
16389.140625	-165.536545	7.094394	3.436554
12209.157227	-170.881882	8.666712	3.780350
13341.791016	-167.667221	7.354096	3.491144
16166.315430	-167.401597	7.547771	3.458324
15922.700195	-165.450638	7.597266	3.448369
10675.114258	-169.233812	8.912421	3.746710
12119.811523	-170.867097	9.046435	3.792049
19083.677734	-167.001724	6.657168	3.502646
18557.863281	-169.438034	6.748002	3.481256
15909.229492	-171.186249	7.723738	3.729531
16642.685547	-165.532503	6.951367	3.523507
11813.437500	-169.390259	8.615819	3.647785
13281.957031	-169.170868	8.474910	3.612851
15073.848633	-171.005540	9.103872	3.754503
10220.565430	-165.245286	9.190604	3.707159
10113.371094	-166.923554	9.302829	3.817943
10523.503906	-169.494416	9.243310	3.813464
11181.450195	-169.847002	8.930175	3.661641
9696.871094	-158.921341	9.322958	3.727847
13328.320313	-169.999512	8.326733	3.642395
13981.571289	-170.638048	7.686714	3.687758
11639.289063	-167.824051	8.559426	3.586341
15304.253906	-169.059098	7.947859	3.652743
15638.969727	-168.656662	7.804269	3.665510
9326.653320	-169.557114	9.557758	3.757981
19024.410156	-169.780351	6.672532	3.541213
13514.968750	-171.358658	8.379460	3.969122
16375.173828	-167.648529	7.414255	3.482914
8982.557617	-156.920105	9.547651	3.631566
14944.035156	-168.526385	7.871942	3.575478
8592.459961	-169.254441	9.667379	3.630769
10671.575195	-168.821396	9.187414	3.724718
8444.603516	-169.052140	9.686781	3.867518
9985.257813	-161.237077	9.416264	3.687597
9753.625977	-149.463303	9.471595	3.685851
10258.894531	-166.843539	9.090591	3.560625
9998.609375	-134.230103	9.348485	3.724468
9809.641602	-169.584565	9.595253	3.616181
17664.935547	-165.203421	6.962873	3.505834
15673.209961	-168.844530	7.498390	3.673283
12044.000000	-169.216339	9.458893	3.578897
15515.949219	-171.793643	8.503532	3.918776
8864.609375	-169.770035	11.144691	3.763599
10663.048828	-170.978777	9.453583	3.712622
8232.559570	-170.055435	10.193706	3.897394
18327.734375	-172.529099	7.659643	3.949568
8073.664551	-167.575546	9.887669	3.677644
7000.715820	-164.638827	11.657371	4.269309

7823.234375	-165.202255	11.636192	4.297426
18867.585938	-170.686375	7.255701	3.909310
9981.324219	-168.142273	9.433485	3.987713
7580.853027	-163.930511	11.250679	3.985060
8027.477539	-164.585373	11.109526	4.125300
7948.440430	-169.745363	11.610647	3.984206
11820.487305	-168.671525	8.394582	3.553260
17574.113281	-167.243414	7.299562	3.752908
18724.105469	-170.821291	7.269009	3.971395
14195.145508	-160.386587	7.176186	3.734639
19585.449219	-170.994171	7.101646	3.999689
17100.806641	-157.416534	7.084296	3.497007
17383.169922	-167.226781	7.040079	3.675033
17463.484375	-169.457443	7.398674	3.657277
13270.639648	-169.580583	8.019820	3.748691
11394.761719	-133.372772	8.348872	3.944326
7660.454102	-81.026878	9.822889	3.586096
17787.187500	-169.853027	7.165822	3.752274
17092.751953	-145.560776	6.739779	3.539863
19327.130859	-170.263016	6.888467	3.624422
21141.812500	-163.535266	6.651044	3.754072
16064.337891	-58.796906	40.000000	0.000000
16508.505859	-171.311785	7.858019	3.965745
13534.134766	-171.082047	9.206852	3.828949
16526.654297	-168.116562	7.365165	3.818962
13217.615234	-168.912079	8.550928	3.667556
10826.141602	-169.290039	9.354886	3.799701
15441.402344	-170.058428	8.483262	3.659800
15904.856445	-171.224594	8.954738	3.728241
13068.186523	-168.243485	7.651011	3.647195
13061.282227	-168.720932	7.564181	3.698500
12660.757813	-170.229874	8.642732	3.616443
12523.034180	-170.875549	8.645427	3.666740
15549.768555	-169.292633	7.995219	3.647771
15691.916016	-169.483079	8.101744	3.654084
15330.664063	-170.324219	8.351891	3.665938
15533.776367	-170.161837	8.323863	3.662046
15403.506836	-169.592026	8.145219	3.624434
10830.605469	-170.628846	9.479841	3.676302
10778.659180	-168.494644	9.526069	3.641801
12330.110352	-168.908663	8.527691	3.641683
12463.106445	-169.157243	8.596170	3.599895
11052.288086	-170.034622	9.273525	3.838458
10938.235352	-169.360064	9.188366	3.812195
15188.000977	-170.586442	8.172300	3.687348
15214.217773	-169.207185	8.151094	3.670792