7 МНОГОКРИТЕРИАЛЬНАЯ ОПТИМИЗАЦИЯ[1]

Часто возникают ситуации, когда необходимо оптимизировать не одну функцию качества или приспособленности, а сразу множество функций. Например, представим, что инженер-строитель хочет построить идеальное здание. Т.е. он хочет построить недорогое, высотное, устойчивое к землетрясениям и энергоэффективное сооружение. Не правда ли отличная задумка? К сожалению, такое здание не может быть построено.

Каждая из таких оптимизируемых функций называется **критерием**. Иногда можно найти решение, которое оптимально по всем критериям. Однако гораздо чаще возникает обратная ситуация, когда критерии не согласуются друг с другом. В таких случаях решением является компромисс по различным критериям. Наш инженер-строитель знает, что нельзя построить совершенное здание: недорогое, высотное, устойчивое и «зеленое». Однако он может рассмотреть *наилучшие возможные варианты*. Существует множество способов определить набор «наилучших вариантов», но наиболее известным является **множество Парето**[2] в пространстве возможных решений.

Предположим, что рассматриваются два здания, M и N. Говорят, что M доминирует N по Парето, если M не хуже N по всем критериям и хотя бы по одному критерию превосходит N. Если так оно и есть на самом деле, то, действительно, в выборе N нет никакого смысла. Ведь M по всем параметрам не уступает, а по каким-то и выигрывает N. Если рассматривать всего два критерия (Дешевле, Более энерго-эффективный), то на рис. 46 показана область пространства, доминируемая данным строительным решением A. Эта область «замкнута»: элементы на ее границе также доминируемы A.

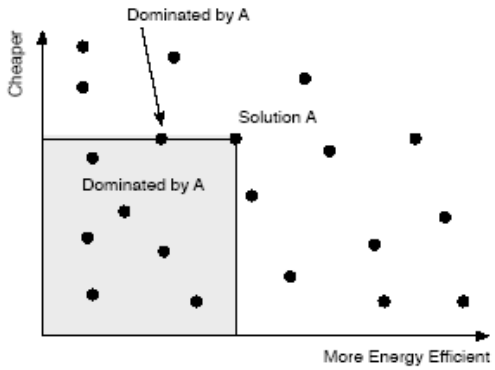

Рис. 46. Область решений, доминируемых по Парето решением A, включая решение на границе. Учтите, что это не изображение пространство фенотипов, а просто результат сравнения по двум критериям

С другой стороны, ни M, ни N не доминируют друг друга, если они равны по всем критериям, либо если N лучше в чем-то одном, а M – в другом. В таких случаях оба решения

---

[1] Перевод раздела из книги Luke S. Essentials of Metaheuristics. A Set of Undergraduate Lecture Notes. Zeroth Edition. Online Version 0.5. October, 2009 (http://cs.gmu.edu/~sean/book/metaheuristics/). Перевел – Юрий Цой, 2009 г.
Любые замечания, касающиеся перевода, просьба присылать по адресу yurytsoy@gmail.com
Данный текст доступен по адресу: http://qai.narod.ru/GA/meta-heuristics_7.pdf

[2] Вильфредо Парето (1848-1923) – итальянский математик, разработавший множество полезных математических концепций для экономики, включая закон Парето для распределения дохода, правило 80-20 (80% событий происходят в 20% случаев, поэтому можно решить большинство проблем, сосредоточившись только на некоторых из них), а также Парето-эффективность и Парето-оптимальность, которые здесь рассматриваются.

M и *N* представляют интерес для нашего инженера. Поэтому одним из способов определения набора «наилучших вариантов» является набор зданий, которые *ничем* не доминируемы. Такие строения называются **недоминируемыми**. Этот набор решений представляет фронт Парето (границу Парето) в пространстве решений. На рис. 47 показана граница Парето для возможных решений в нашем двухкритериальном пространстве. В двумерном случае фронт Парето представляет собой кривую, определяющую нечто вроде внешней границы. В трехмерном случае – это будет что-то, напоминающее оболочку. Если имеется решение, которое превосходит все остальные (эдакий супермен), то фронт сожмется до этого одного решения.

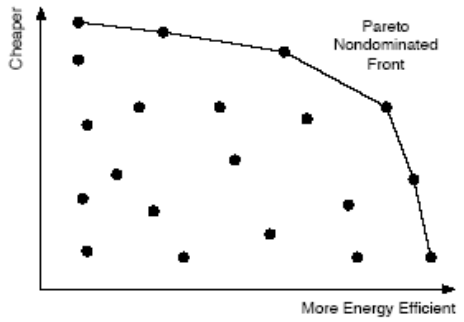

Рис. 47. Граница Парето для недоминируемых решений

Как показано на рис. 48, граница Парето может иметь разный вид. **Выпуклые** границы искривлены в строну лучших решений, а **вогнутые** – в обратную сторону. **Невыпуклые** границы не полностью выпуклы и включают вогнутые участки. Границы также могут быть не непрерывными, что означает наличие областей вдоль фронта, в которых просто не может быть решений: они будут доминироваться другими решениями в «правильных» участках границы. Также существуют и **локально-оптимальные по Парето фронты** в пространстве, когда для определенной точки, не принадлежащей глобальной границе Парето, известно, что она доминирует все точки в некоторой окрестности.

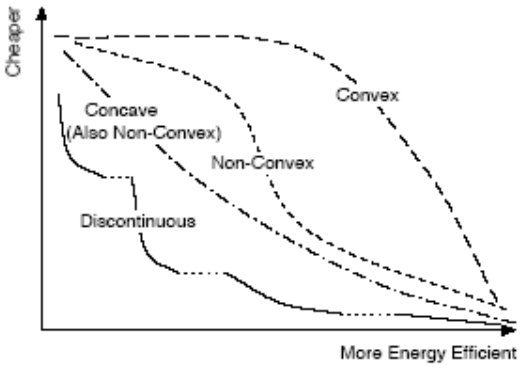

Рис. 48. Четыре вида границ Парето

Разброс (Spread). Недостаточно просто предоставить инженеру 100 точек на границе Парето. Что если все они находятся в дальнем углу фронта? Это мало скажет ему о возможных вариантах решений. Более вероятно, что ему необходимы точки, которые равномерно **распределены** вдоль всей границы. Поэтому многие алгоритмы, которые оптимизируют границу Парето используют меры разнообразия. Интересно то, что меры расстояния, измеряемого в пространстве генотипов или фенотипов, используются очень редко. Вместо этого рассматривается расстояние между приспособленностями, т.е.

насколько удалены друг от друга решения в пространстве многих критериев. Часто такой способ оказывается вычислительно более простым, чем вычисление расстояния между генотипами или фенотипами.

Проблема слишком большого количества критериев. С ростом числа критериев размер популяции, необходимый для достаточно точного описания границы Парето, увеличивается экспоненциально. Все рассматриваемые в данном разделе методы испытывают определенные затруднения при большом количестве критериев (под «большим» количеством я подразумеваю «возможно, больше 4»). Это сложность, порождаемая самой задачей. Для борьбы с ней исследователи в последнее время применяют более экзотические методы, в частности, методы, рассматривающие **гиперобъем**, накрываемый границей Парето. Однако эти методы сложны и требует больших расчетов. Мы же будет рассматривать более простые методы.

Замечание по определению приспособленности. Традиционно в литературе по многокритериальной оптимизации приспособленность определяется аналогично ошибке. Т.е. чем *меньше* значение критерия, тем лучше. Поэтому в большинстве увиденных диаграммах оптимизации по Парето границу будут формировать особь, находящиеся ближе к началу координат. Я, стараясь выдерживать последовательность и целостность изложения, в данном разделе буду полагать, что чем значения критериев *больше*, тем лучше. Это объясняет характер рисунков и описаний алгоритмов данной главы.

7.1. НАИВНЫЕ МЕТОДЫ

Прежде чем перейти к методам, использующим границу Парето, рассмотрим более простые (но иногда весьма эффективные) методы, решающие многокритериальные проблемы в той же манере, что и большинство «традиционных» мета-эвристических алгоритмов.

Простейшим способом сделать это является объединение всех критериев в одну функцию приспособленности, используя линейное соотношение. Например, вы считаете, что доступная стоимость (*Cheap*) в 10 раз значимее, чем высотность (*Height*), в 5 раз значимее, чем устойчивость к землетрясениям (*Earthquake Resistant*) и в 4 раза значимее энергоэффективности (*Energy Efficient*). Тогда можно определить качество решения в виде взвешенной суммы, характеризующей удовлетворенность критериев:

$$\text{Fitness}(i) = \text{Cheapness}(i) + \frac{1}{10}\text{Height}(i) + \frac{1}{5}\text{EarthquakeResistance}(i) + \frac{1}{4}\text{EnergyEfficiency}(i)$$

Мы уже многократно сталкивались с подобным. Например: линейное давление жадности; среднее различных тестовых случаев. Здесь есть три проблемы. Во-первых, необходимо определиться с тем, насколько один критерий важнее другого. Часто это очень сложно сделать, или даже практически невозможно если критерии – нелинейные (т.е. разница между высотой в, скажем, 9 и 10 единиц намного больше, чем разница между высотой в 2 и 3 единицы). Это та же самая базовая проблема, которую мы обсуждали, рассматривая линейное давление жадности в Разделе 4.6 (Раздувание). Во-вторых, понятно, что если M доминирует по Парето над N, то *сразу же* Приспособленность(M) ≥ Приспособленность(N), предполагая, что веса положительны. Таким образом, метод Парето дает в некотором смысле бесплатную информацию. В-третьих, рассмотрение взвешенной суммы не обязательно приводит к границе Парето. Предположим простейший случай, когда мы просто суммируем критерии (т.е., все веса равны 1). Пусть имеется два критерия, граница Парето для которых изображена на рис. 49. Решение A очень близко к границе, и поэтому более привлекательно.

Однако сумма критериев для решения *B* имеет большее значение, и поэтому оно будет выбрано вместо *A* благодаря приспособленности.

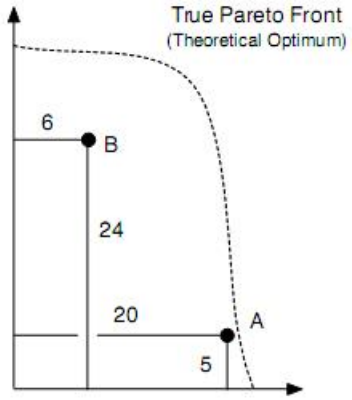

Рис. 49. *A* может рассматриваться как доминирующее решение, однако *B* имеет большую сумму

Для решения первой проблемы (необходимость определения весов), можно отбросить идею линейной комбинации и просто рассматривать критерии как несравнимые функции. К примеру, допустим, что мы задали предпочтения критериев для их упорядочивания: *M* лучше, чем *N*, если имеет большую Высоту. Если значения Высоты равны, то лучше то решение, у которого меньше Стоимость. В противном случае проверяем Устойчивость к землетрясениям. А потом Энерго-эффективность. Можно описать процедуру селекции путем расширения Алгоритма 63 (Лексикографическая турнирная селекция) для случая, когда количество критериев больше двух. Будем просто при сравнении особей перебирать критерии (от более важных к менее важным) до тех пор, пока не установим что одна из особей превосходит другую по очередному критерию. Предполагая, что определена функция `ObjectiveValue` (*objective*, *individual*), которая возвращает качество *особи* для соответствующего *критерия*, турнирную селекцию можно провести следующим образом:

Algorithm 94 *Multiobjective Lexicographic Tournament Selection*

```
1: Best ← individual picked at random from population with replacement
2: O ← {O_1,...,O_n} objectives to assess with        ▷ In lexicographic order, most to least preferred.
3: t ← tournament size, t ≥ 1

4: for i from 2 to t do
5:     Next ← individual picked at random from population with replacement
6:     for i from 1 to n do
7:         if ObjectiveValue(O_i, Next) > ObjectiveValue(O_i, Best) then        ▷ Clearly superior
8:             Best ← Next
9:             break from inner for
10:        else if ObjectiveValue(O_i, Next) < ObjectiveValue(O_i, Best) then   ▷ Clearly inferior
11:            break from inner for
12: return Best
```

Можно также выбирать случайный критерий и использовать в качестве приспособленности для данной селекции:

Algorithm 95 *Multiobjective Ratio Tournament Selection*

```
Best ← individual picked at random from population with replacement
O ← {O_1,...,O_n} objectives to assess with
t ← tournament size, t ≥ 1

j ← random number picked uniformly from 1 to n
for i from 2 to t do
    Next ← individual picked at random from population with replacement
    if ObjectiveValue(O_j, Next) > ObjectiveValue(O_j, Best) then
        Best ← Next
return Best
```

Или можно использовать голосование: особь считается лучше, если она показывает лучшие значения для большего количества критериев:

Algorithm 96 *Multiobjective Majority Tournament Selection*

```
Best ← individual picked at random from population with replacement
O ← {O_1,...,O_n} objectives to assess with, more important objectives first
t ← tournament size, t ≥ 1

for i from 2 to t do
    Next ← individual picked at random from population with replacement
    c ← 0
    for each objective O_j ∈ O do
        if ObjectiveValue(O_j, Next) > ObjectiveValue(O_j, Best) then
            c ← c + 1
        else if ObjectiveValue(O_j, Next) < ObjectiveValue(O_j, Best) then
            c ← c − 1
    if c > 0 then
        Best ← Next
return Best
```

И, наконец, можно расширить Алгоритм 64 (Двойная турнирная селекция) для случая, когда имеется больше двух критериев. Турнир проводится на основании одного критерия. Особи для турнира отбираются с использованием турнира по другому критерию. Особи для этого турнира отбираются с использованием турнира по третьему критерию и т.д. Таким образом, выигрывает турнир часто та особь, которая является «мастером на все руки», т.е. показывает хорошие результаты по *всем* критериям.

Algorithm 97 *Multiple Tournament Selection*

```
O ← {O_1,...,O_n} objectives to assess with
T ← {T_1,...,T_n} tournament sizes for the objectives in O, all ≥ 1        ▷ Allows different weights

return ObjectiveTournament(O, T)

procedure ObjectiveTournament(O, T)
    Best ← individual picked at random from population with replacement
    n ← ||O||                          ▷ O and T change in size. The current last elements are O_n and T_n
    if O − {O_n} is empty then                                  ▷ O_n is the last remaining objective!
        Best ← individual picked at random from population with replacement
    else
        Best ← ObjectiveTournament(O − {O_n}, T − {T_n})        ▷ Delete the current objective
    for i from 2 to T_n do
        if O − {O_n} is empty then                              ▷ This is the remaining objective!
            Next ← individual picked at random from population with replacement
        else
            Next ← ObjectiveTournament(O − {O_n}, T − {T_n})    ▷ Delete the current objective
        if ObjectiveValue(O_n, Next) > ObjectiveValue(O_n, Best) then
            Best ← Next
    return Best
```

7.2. НЕДОМИНИРУЕМАЯ СОРТИРОВКА

Предыдущие алгоритмы пытаются объединять критерии в отдельное значение приспособленности путем определения соотношений между ними. Однако большинство современных алгоритмов вместо этого используют понятия доминирования по Парето, чтобы более точно оценивать «хорошие» решения в многокритериальном смысле.

Простой способ для этого: разработка оператора турнирной селекции, основанной на Парето-доминировании. Но прежде давайте дадим определение. Особь A доминирует по Парето особь B, если A не хуже B по всем критериям, а хотя бы по одному критерию лучше.

Algorithm 98 *Pareto Domination*

```
A ← individual A                                    ▷ We'll determine: does A dominate B?
B ← individual B
O ← {O_1,...,O_n} objectives to assess with

a ← false                                           ▷ A is superior to B somewhere
for each objective O_i ∈ O do
    if ObjectiveValue(O_i, A) > ObjectiveValue(O_i, B) then
        a ← true
    else if ObjectiveValue(O_i, B) > ObjectiveValue(O_i, A) then
        return false
return a
```

Теперь можем построить процедуру бинарного турнирного отбора, основанную на Парето-доминировании:

Algorithm 99 *Pareto Domination Binary Tournament Selection*

```
P ← population
Pa ← individual picked at random from P with replacement
Pb ← individual picked at random from P with replacement
if Pa Pareto Dominates Pb then
    return Pa
else if Pb Pareto Dominates Pa then
    return Pb
else
    return either Pa or Pb, chosen at random
```

К сожалению даже если две особи не доминируют по Парето друг друга, и поэтому одинаково привлекательны для экспериментатора, необходимо выбрать только одну особь в целях оптимизации. В частности, если *A* многократно доминируема другими особями в популяции, а *B* – ни разу, то мы склонны выбрать особь *B*, т.к. нам нужно для нового поколения выбрать особь, лучше чем А. Естественно, *B* не доминирует по Парето *A*. Но А является частью общей кучи.

Чтобы понять этот термин, нам необходимо определить, насколько близко особь располагается к границе Парето. Существует множество различных способов, и один из них (**сила**) будет рассматриваться в следующем разделе. Здесь мы рассмотрим новую концепцию, которая называется Ранг границы Парето. Особи непосредственно на границе имеют ранг 1. Если мы *удалим этих особей из популяции*, а затем посчитаем новую границу, то особи на этой границе будут иметь ранг 2. Если удалим и этих особей и вычислим новую границу, получим ранг 3 и т.д. Это чем-то напоминает чистку лука. На рис. 50 изображены ранги.

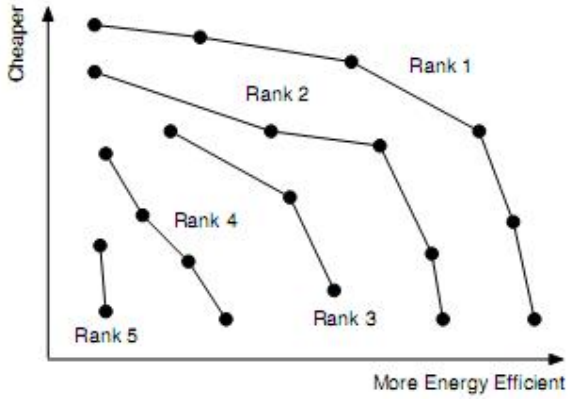

Рис. 50. Ранги Парето

Начнем с того, что рассмотрим, как вычисляется граница Парето. Вся суть заключается в том, что добавляем особь к границе, если эта особь не доминируема другими особями, уже находящимися на границе, и удаляем с границы тех особей, которые оказались доминируемыми этой новой особью. Все достаточно просто:

Algorithm 100 *Computing a Pareto Non-Dominated Front*

```
G ← {G_1,...,G_m} Group of individuals to compute the front among      ▷ Often the population
O ← {O_1,...,O_n} objectives to assess with

F ← { }                                                                ▷ The front
for each individual G_i ∈ G do
    F ← F ∪ {G_i}                                          ▷ Assume G_i's gonna be in the front
    for each individual F_j ∈ F other than G_i do
        if F_j Pareto Dominates G_i given O then   ▷ Oh well, guess it's not gonna stay in the front
            F ← F − {G_i}
            break out of inner for-loop
        else if G_i Pareto Dominates F_j given O then     ▷ An existing front member knocked out!
            F ← F − {F_j}
return F
```

Вычислить ранги легко: находим первую границу, затем убираем особей, снова находим границу и т.д. После того, как этой процедурой будут обработаны все особи, можем использовать Ранг границы Парето для особи в качестве ее приспособленности. Поскольку чем ниже ранг, тем лучше, его преобразование в приспособленность можно сделать следующим образом:

$$\text{Fitness}(i) = \frac{1}{1 + \text{ParetoFrontRank}(i)}$$

Алгоритм для вычисления рангов делает сразу два дела: сначала популяция *P разбивается* по рангам и каждый ранг (группа особей) сохраняется в векторе *F*. Далее алгоритм *присваивает* номер ранга особи (возможно, записывает значение ранга куда-нибудь «внутрь» особи). Таким образом, потом мы всегда сможем спросить: (1) какие особи имеют ранг *i*? (2) какой ранг у особи *j*? Такая процедура разработана Н. Шринивасом и Кальянмой Дебом[3] и называется **Недоминируемая сортировка**:

Algorithm 101 *Front Rank Assignment by Non-Dominated Sorting*

```
P ← Population
O ← {O_1,...,O_n} objectives to assess with

P' ← P                                          ▷ We'll gradually remove individuals from P'
R ← ⟨ ⟩                                  ▷ Initially empty ordered vector of Pareto Front Ranks
i ← 1
repeat
    R_i ← Pareto Non-Dominated Front of P' using O
    for each individual A ∈ R_i do
        ParetoFrontRank(A) ← i
        P' ← P' − {A}                                  ▷ Remove the current front from P'
    i ← i + 1
until P' is empty
return R
```

Разреженность. Желательно сделать так, чтобы особи в популяции были равномерно распределены вдоль границы. Для этого можно ввести некоторую метрику для измерения расстояния между особями с одинаковым Рангом границы Парето. Дадим определение

[3] Впервые опубликовано в Srinivas N., Deb K. Multiobjective optimization using nondominated sorting in genetic algorithms // Evolutionary Computation, 1994, vol. 2, p. 221–248. В этой статье также был описан Алгоритм 100.

разреженности для особи: особь находится в *разреженной области*, если ближайшая к ней особь одного с ней ранга располагается не слишком близко.

На рис. 51 дан иллюстративный пример для этого понятия.

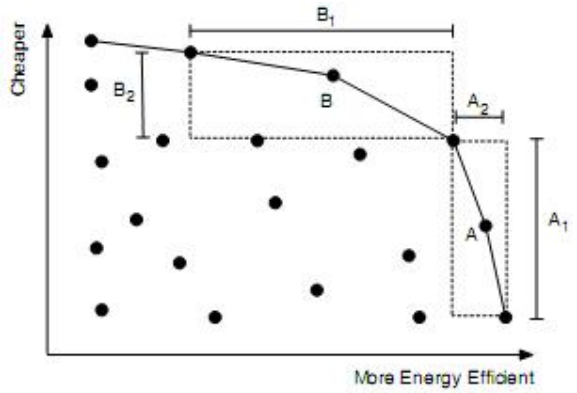

Рис. 51. Разреженность для особи B выше, чем для особи A, поскольку $A_1+A_2 < B_1+B_2$

Будем вычислять разреженность с использования Манхеттеновского расстояния[4] по всем критериям для правого и левого соседа каждой особи для рассматриваемого критерия. Особи на концах Ранга границы Парето считаются находящимися в бесконечно разреженной области.

Algorithm 102 *Multiobjective Sparsity Assignment*

$R \leftarrow \langle ... \rangle$ Provided Pareto Front Ranks of Individuals ▷ To compute a single rank G, pass in $\langle G \rangle$.
$O \leftarrow \{O_1, ..., O_n\}$ objectives to assess with

for each Pareto Front Rank $F \in R$ **do**
  for each individual $F_j \in F$ **do**
    Sparsity$(F_j) \leftarrow 0$
  for each objective $O_i \in O$ **do**
    $F' \leftarrow F$ sorted by ObjectiveValue given objective O_i
    Sparsity$(F'_1) \leftarrow \infty$
    Sparsity$(F'_{||F||}) \leftarrow \infty$ ▷ Each end is always great!
    for j from 2 to $||F'|| - 1$ **do**
      Sparsity$(F'_j) \leftarrow$ Sparsity(F'_j) + ObjectiveValue(O_i, F'_{j+1}) − ObjectiveValue(O_i, F'_{j-1})
return R with Sparsities assigned

Теперь можно использовать разреженность, чтобы реализовать нечто вроде перенаселения, только в многокритериальном пространстве, а не в пространстве генотипов или фенотипов. Определим турнирную селекцию таким образом, чтобы особи прежде всего отбирались по Рангу границы Парето, а неоднозначные ситуации будем разрешать с использованием разреженности. Идея заключается в том, чтобы выбирать особей, которые не только ближе остальных к настоящей границе Парето, но «хорошо» по ней распределены.

[4] На Манхеттен наложена сетка дорог, и вы не можете просто перейти напрямую из точки А в точку Б, если, конечно, не умеете прыгать через крыши. Вместо этого необходимо пройти сколько-то кварталов по горизонтали, а затем еще сколько-нибудь кварталов по вертикали. Сумма пройденных кварталов и будет Манхеттеновским расстоянием.

Algorithm 103 *Non-Dominated Sorting Lexicographic Tournament Selection With Sparsity*

```
P ← Population with Pareto Front Ranks assigned
Best ← individual picked at random from P with replacement
t ← tournament size, t ≥ 1

for i from 2 to t do
    Next ← individual picked at random from P with replacement
    if ParetoFrontRank(Next) < ParetoFrontRank(Best) then          ▷ Lower ranks are better
        Best ← Next
    else if ParetoFrontRank(Next) = ParetoFrontRank(Best) then
        if Sparsity(Next) > Sparsity(Best) then
            Best ← Next                                            ▷ Higher sparsities are better
return Best
```

Этот подход работает хорошо и сам по себе. Но **Non-Dominated Sorting Genetic Algorithm II (NSGA-II)**[5] разработанный Кальянмой Дебом, Амритом Пратапом, Самиром Агарвалом и Т. Меяриваном[6] имеет некоторые улучшения: он сохраняет *всех* хороших особей найденных в процессе поиска, т.е. работает похоже с ($m + l$) эволюционной стратегией и элитаризмом.

Algorithm 104 *An Abstract Version of the Non-Dominated Sorting Genetic Algorithm II (NSGA-II)*

```
P ← {P_1,..., P_m} Build Initial Population
AssessFitness(P)                          ▷ Compute the objective values for the Pareto front ranks
R ← ⟨...⟩ Pareto Front Ranks of P
for each Pareto Front Rank in R_i ∈ R do
    Compute Sparsities of Individuals in R_i
BestFront ← Pareto Front of P
repeat
    Q ← Breed(P), using Algorithm 103 for selection (typically with tournament size of 2)
    AssessFitness(Q)                      ▷ Compute the objective values for the Pareto front ranks
    Q ← Q ∪ P
    P ← {}
    R ← Compute Front Ranks of Q
    BestFront ← Pareto Front of Q
    for each Front Rank R_i ∈ R do
        Compute Sparsities of Individuals in ⟨R_i⟩          ▷ Just for R_i, no need for others
        if ||P|| + ||R_i|| ≥ m then                        ▷ This will be our last front to load into P
            P ← P ∪ the Sparsest m − ||P|| individuals in R_i, breaking ties arbitrarily
            break from the for loop
        else
            P ← P ∪ R_i                                                          ▷ Just dump it in
until BestFront is the ideal Pareto front or we have run out of time
return BestFront
```

Общая идея заключается в использовании *P* как **архива** для лучший найденных особей. Произведя новую популяцию *Q* из *P*, устраивается отбор, чтобы решить, какие особи останутся в *P*. Такие алгоритмы известны, как **алгоритмы с архивом** (*archive algorithms*).

---

[5] *Честно говоря, перевод звучит довольно глупо (Генетический алгоритм с недоминируемой сортировкой II), поэтому в основной части текста перевода нет. Аналогичное замечание и для большинства других алгоритмов с порядковыми номерами. – Прим. перев.*

[6] Deb K., Pratap A., Agarwal S., Meyarivan T. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II // in Marc Schoenauer, et al., editors, Parallel Problem Solving from Nature (PPSN VI), pages 849–858, Springer, 2000. В этой статье также представлен Алгоритм 102.

Обычно такой подход *очень сильно использует* уже найденные решения в ущерб поисковым возможностям. Однако в многокритериальной оптимизации все немного по-другому, т.к. мы ищем не *одну единственную точку* в пространстве поиска, а целую *границу Парето*, которая распределена в пространстве критериев, что и вносит необходимое разнообразие в решение задачи.

Отметим, что разреженность вычисляется только для некоторых Рангов границы Парето, поскольку эта характеристика необходима только соответствующим особям, а другие ранги просто отбрасываются. Хотя можете посчитать значения разреженность для всех особей в *Q*, ничего особенного в этом нет.

7.3 ПАРЕТО-СИЛА

Ранг границы Парето не единственный способ использования границы Парето для вычисления приспособленности. Мы также можем определить **силу** (*strength*) особи, равную количеству особей, которые доминируются по Парето данной особью.

Можно использовать силу особи в качестве ее приспособленности. Однако здесь есть проблема. Сила не обязательно показывает, насколько близко располагается особь к границе Парето. Действительно, особи по краям границы часто будут не такими сильными, как особи достаточно далекие от границы (рис. 52).

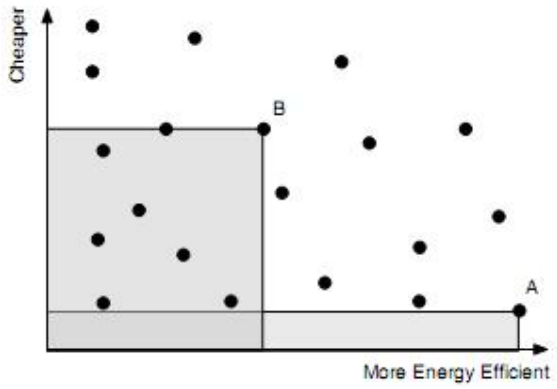

Рис. 52. Особь *A* ближе к границе Парето, но особь *B* – сильнее

Вместо силы мы могли бы определять **слабость** (*weakness*) особи, как *количество особей, доминирующих данную*. Очевидно, что особи на границе Парето имеют слабость равную 0, а особи вдали от границы имеют большую слабость. Немногим более удачной версией слабости особи является ее **хилость** (***wimpiness***)[7]: суммарная общая сила всех особей, доминирующих данную, т.е. для особи *i* и группы *G*:

$$\text{Wimpiness}(i) = \sum_{g \in G \text{ that Pareto Dominate } i} \text{Strength}(g)$$

В идеале желательно, чтобы хилость была как можно *меньше*. Тогда недоминируемая особь имеет хилость равную 0. Мы также могли бы в качестве приспособленности использовать величину «не- хилости». Для этого необходимо преобразовать хилость так, чтобы более хилые особи имели меньшую приспособленность. Например:

[7] *Конечно же,* названия этих терминов придумал я сам.

$$\text{Fitness}(i) = \frac{1}{1 + \text{Wimpiness}(i)}$$

Эккарт Цитцлер, Марко Ломаннс и Лотар Тиеле разработали алгоритм с архивом, использующий понятие силы (или, что более корректно, хилости), и известный как **Strength Pareto Evolutionary Algorithm** (или **SPEA**). На данный момент его улучшенный вариант, **SPEA2**, непосредственно конкурирует с NSGA-II и другими стохастическими многокритериальными алгоритмами оптимизации[8]. Как и NSGA-II, SPEA2 сохраняет архив наилучших найденных особей на границе Парето, равно как и некоторое количество других особей. Однако в SPEA2 используется Парето-мера wimpiness, и степень перенаселения популяции определяется на основе расстояний между особями в многокритериальном пространстве, а не в пространстве рангов.

Мера схожести в SPEA2 вычисляется как расстояние до других особей популяции, в частности, до *k*-й ближайшей особи. Существует довольно много известных эффективных способов вычисления этой величины. Здесь я просто приведу чрезвычайно неэффективный, но зато простой подход[9]. Будем просто вычислять попарные расстояния между особями. Затем для каждой особи в популяции найдем *k*-ю ближайшую особь, путем сортировки по расстояниям до рассматриваемой особи. Вычислительная сложность $O(n^2 \lg n)$, где n – размер популяции. Не очень-то хорошо.

Algorithm 105 *Compute the Distance of the Kth Closest Individual*

$P \leftarrow \{P_1, ..., P_m\}$ Population
$O \leftarrow \{O_1, ..., O_n\}$ objectives to assess with
$P_l \leftarrow$ individual to compute kth closest individual to
$k \leftarrow$ desired individual index (the kth individual from l)

global $D \leftarrow m$ vectors, each of size m ▷ D_i holds an vector of distances of various individuals i
global $S \leftarrow \{S_1, ..., S_m\}$ ▷ S_i will be true if D_i has already been sorted
perform once only
  for each individual $P_i \in P$ **do**
    $V \leftarrow \{\}$ ▷ our distances
    for each individual $P_j \in P$ **do**
      $V \leftarrow V \cup \left\{ \sqrt{\sum_{m=1}^{n} (\text{ObjectiveValue}(O_m, P_i) - \text{ObjectiveValue}(O_m, P_j))^2} \right\}$
    $D_i \leftarrow V$
    $S_i \leftarrow$ false
perform each time
  if S_l is false **then** ▷ Need to sort
    Sort D_l, smallest first
    $S_l \leftarrow$ true
  $W \leftarrow D_l$
  return W_{k+1} ▷ It's W_{k+1} because W_1 is always 0: the distance to the same individual

[8] Их история имеет много пересечений. Алгоритм SPEA был представлен в статье Zitzler E., Thiele L. Multiobjective evolutionary algorithms: A comparitive case study and the strength pareto approach // IEEE Transactions on Evolutionary Computation, 1999, vol., 3, no. 4, pp. 257–271. Затем в 2000 году появился алгоритм NSGA-II, а SPEA2 был описан в статье Zitzler E., Laumanns M., Thiele L. SPEA2: Improving the strength pareto evolutionary algorithm for multiobjective optimization // K. Giannakoglou, et al., editors, Evolutionary Methods for Design, Optimization, and Control, 2002, pp. 19–26.

[9] Да ладно, вычисление приспособленности все равно занимает наибольшее время.

Зная хилость особи и k-го ближайшего соседа мы, наконец, можем посчитать приспособленность Сначала определим предварительную приспособленность G_i:

$$G_i \leftarrow \text{Wimpiness}(i) + \frac{1}{2 + d_i}$$

где d_i – расстояние до k-го ближайшего соседа к i–й особи, $k = \left\lfloor \sqrt{\| P \|} \right\rfloor + 1$. Чем меньше G_i, тем лучше. Идея заключается в том, что при больших расстояниях d_i значение G_i уменьшается (т.к. тогда особь находится далеко от других особей, нам нужно разнообразие!) и, аналогично, малое Wimpiness также уменьшает G_i.

На самом деле в SPEA2 в качестве приспособленности и используется G_i. Но чтобы соблюсти нашу традицию (чем больше приспособленность, тем лучше), будем вычислять приспособленность также как и раньше:

$$\text{Fitness}(i) = \frac{1}{1 + G_i}$$

На каждой итерации SPEA2 составляет архив из особей, находящихся на текущей границе Парето. Размер архива равен n. Если особей на границе недостаточно, чтобы заполнить архив, он дозаполняется следующими по приспособленности особями. Если же наоборот особей на границе *слишком много* для архива, то в SPEA2 производится их отсечение путем последовательного удаления особей с минимальным расстоянием до k-го ближайшего соседа (начиная с k = 1, затем k = 2 и т.д.). Целью является заполнить архив такими особями с границы Парето, которые находятся далеко друг от друга и от других особей в популяции. Алгоритм построения архива выглядит следующим образом:

Algorithm 106 *SPEA2 Archive Construction*

$P \leftarrow \{P_1, \ldots, P_m\}$ Population
$O \leftarrow \{O_1, \ldots, O_n\}$ objectives to assess with
$a \leftarrow$ desired archive size

$A \leftarrow$ Pareto non-dominated front of P ▷ The archive
$Q \leftarrow P - A$ ▷ All individuals not in the front
if $||A|| < a$ **then** ▷ Too small! Pack with some more individuals
    Sort Q by fitness
    $A \leftarrow A \cup$ the $a - ||A||$ fittest individuals in Q, breaking ties arbitrarily
while $||A|| > a$ **do** ▷ Too big! Remove some "k-closest" individuals
    $Closest \leftarrow A_1$
    $c \leftarrow$ index of A_1 in P
    for each individual $A_i \in A$ except A_1 **do**
        $l \leftarrow$ index of A_i in P
        for k from 1 to $m - 1$ **do** ▷ Start with $k = 1$, break ties with larger values of k
            if DistanceOfKthNearest$(k, P_l) <$ DistanceOfKthNearest(k, P_c) **then**
                $Closest \leftarrow A_i$
                $c \leftarrow l$
                break from inner **for**
            else if DistanceOfKthNearest$(k, P_l) >$ DistanceOfKthNearest(k, P_c) **then**
                break from inner **for**
    $A \leftarrow A - \{Closest\}$
return A

И вот теперь мы готовы к тому, чтобы описать SPEA2 на «высоком» уровне. Вообще-то он очень прост. Для данной популяции P и (изначально пустого) архива A, будем формировать новый архив границы Парето из $P \cup A$, удаляя при необходимости из объединения «близких» особей, плюс некоторые приспособленные особи из P, чтобы заполнить пропуски. Затем создаем новую популяцию P размножением особей из A (с течением времени, по мере улучшения границы Парето, этот процесс становится похожим на случайную селекцию). Вроде бы чем-то напоминает $(m + l)$? Так и есть!

Algorithm 107 *An Abstract Version of the Strength Pareto Evolutionary Algorithm 2 (SPEA2)*

$a \leftarrow$ desired archive size

$P \leftarrow \{P_1, ..., P_m\}$ Build Initial Population
$A \leftarrow \{\}$ ▷ Archive ▷ Typically m, that is, the population and archive are the same size
repeat
    AssessFitness(P)
    BestFront $\leftarrow$ Pareto Front of $P \cup A$
    $A \leftarrow$ Construct SPEA2 Archive of size a from $P \cup A$
    $P \leftarrow$ Breed(A), using tournament selection of size 2 ▷ Fill up to the old size of P
until *BestFront* is the ideal Pareto front or we have run out of time
return *BestFront*

SPEA2 и NSGA-II представляют, попросту говоря, вариации $(m + l)$ в многокритериальном пространстве, оснащенные механизмом поддержания разнообразия и процедурой отбора особей наиболее близких к границе Парето. Оба алгоритма, SPEA2 и NSGA-II,[10] весьма впечатляют своими возможностями, хотя NSGA-II несколько проще и имеет меньшую вычислительную сложность в не «навороченных» версиях.

[10] Поверьте мне, я знаю. Збигнев Сколички и я как-то раз разработали массивно-параллельную островную модель для многокритериальной оптимизации. При наличии n критериев, острова располагались на сетке с n узлами, по одному на критерий. К примеру, в случае двух критериев, сетка представляла собой отрезок. Для трех критериев использовалась треугольная сетка. Если критериев было 4, то сетка имела объем и выглядела как тетрагедрон (треугольная пирамида). На каждом острове приспособленность вычислялась как взвешенная сумма критериев. Чем ближе был остров к узлу, тем больше был вес соответствующего критерия. Таким образом, острова в узлах сетки на 100% использовали только один критерий, в то время как (к примеру) острова в центре сетки имели равные веса для каждого критерия. Проще говоря, каждый остров искал свой собственный путь к границе Парето, в результате чего (в случае удачи) получался набор точек, равномерно распределенных вдоль границы. И мы получили вполне неплохие результаты. Однако SPEA2, работая на одном единственном компьютере, разделал нас под орех.