**Версия 2.2 от 01.03.14**

**Алгоритм с учетом фиксации конкретной точки на конкретном маршруте**

Вводится новый признак «жесткой фиксации» gf = номеру маршрута, к которому жестко привязана точка доставки.

**Процедура «встраивания» точки между двумя зафиксированными (Алгоритм VSTR - алгоритм применяется и на этапе первоначального распределения точек по маршрутам и на этапе оптимизации маршрутов)**

1. Зафиксированные точки упорядочиваются по времени, т.е. первой назначается точка с меньшей левой границей временной рамки t1
2. Если левые границы совпадают, то первой назначается точка с меньшей длиной временной рамки t2-t1
3. Накладываем фильтр – находим все точки
   1. неиспользованные в других маршрутах
   2. удовлетворяющих условию загрузки z(i)+zsum < zmax
   3. удовлетворяющих условию попадания во временные рамки (временная рамка обслуживания точки i и попадание во временную рамку точки 2): начиная от времени отъезда от первой точки + время приезда на точку i + время обслуживания точки i + время доезда до точки 2
4. Из найденных точек выбираем точку с минимальным суммарным расстоянием до точек 1 и 2: r(зафикс.точка1,i)+r(i,зафикс.точка2) -> min
5. Производится проверка – при встраивании этой точки в маршрут не происходит ли сдвиг временных рамок более поздних зафиксированных точек из заданных (по аналогии проверки шаг 3, пункт с). Если не происходит, встраиваем точку в маршрут, иначе отбрасываем ее и повторяем шаги 4-5 до тех пор, пока либо не найдем точку, либо свободные точки не закончатся.
6. Проверяем, можем ли еще встроить точки до точки 2 – точке1 переприсваиваем значение найденной точки i, и повторяем шаги 3-5 до тех пор, пока фильтр не окажется пустым

**Формирование начальных маршрутов с учетом фиксации точек**

**Алгоритм FORM**

На входе алгоритма список точек, у некоторых из них уже заполнен ненулевой признак gf=номеру маршрута

1. Выбираем 1ый маршрут с зафиксированной точкой
2. Выполняем процедуру встраивания (алгоритм VSTR) между складом и первой зафиксированной точкой в маршруте
3. Последовательно выполняем процедуру встраивания между j и j+1 зафиксированными точками
4. Как только загрузка ТС превышает 90% (настраиваемый параметр) – выполняем процедуру встраивания между последней зафиксированной точкой и складом (т.е. двигаемся по направлению к складу)

**Оптимизация маршрутов**

**Алгоритм OPTIM**

После начального формирования маршрутов переходим к фазе их оптимизации, как описано в алгоритме 2.0

При этом добавляется новое условие в фильтре – признак фиксации gf, т.е. эти точки остаются привязанными к конкретному маршруту – их исключаем из фильтра при поиске точек в близком кластере

**Уточнение по алгоритму оптимизации**

После выбора 2 маршрутов подаем подмножество заказов только этих двух маршрутов на вход алгоритма FORM (с учетом алгоритма VSTR).

После формирование маршрутов проверяем условия:

1. Все точки использованы в 2 маршрутах (не осталось лишних)
2. Суммарное время маршрутов уменьшилось по сравнению с исходным вариантом.

Если выполняется, считаем маршрут с кратчайшим расстояние из построенных двух сформированным, и переходим к следующему шагу – берем второй маршрут из рассматриваемых, находим к нему ближайший в смысле «центра тяжести» и повторяем Алгоритм OPTIM.

Если не выполняется, оптимизируем маршруты – выбираем по одной точки из маршрута 2 (более длинного) и пытаемся встроить в маршрут 1 (более короткий), при этом если суммарное время двух маршрутов уменьшилось, считаем точку встроенной в 1-ый маршрут, и переходим к следующей.

Если точку встроить не удалось, оставляем ее во 2-ом маршруте и переходим к следующей.

При встраивании точки параллельно исключаем одну точку из первого маршрута (с сопоставимыми временными рамками точки, встраиваемой в 1-ый маршрут) и пытаемся ее встроить во 2-ой, с аналогичным условием – уменьшение суммарного времени 2-х маршрутов.

**Версия 2.1 от 21.11.13**

**Алгоритм с учетом фиксации конкретной точки на конкретном маршруте**

Вводится новый признак «жесткой фиксации» gf = номеру маршрута, к которому жестко привязана точка доставки.

**Процедура «встраивания» точки между двумя зафиксированными**

1. Зафиксированные точки упорядочиваются по времени, т.е. первой назначается точка с меньшей левой границей временной рамки t1
2. Если левые границы совпадают, то первой назначается точка с меньшей длиной временной рамки t2-t1
3. Накладываем фильтр – находим все точки
   1. неиспользованные в других маршрутах
   2. удовлетворяющих условию загрузки z(i)+zsum < zmax
   3. удовлетворяющих условию попадания во временные рамки (временная рамка обслуживания точки i и попадание во временную рамку точки 2): начиная от времени отъезда от первой точки + время приезда на точку i + время обслуживания точки i + время доезда до точки 2
4. Из найденных точек выбираем точку с минимальным суммарным расстоянием до точек 1 и 2: r(зафикс.точка1,i)+r(i,зафикс.точка2) -> min
5. Проверяем, можем ли еще встроить точки до точки 2 – точке1 переприсваиваем значение найденной точки i, и повторяем шаги 3-5 до тех пор, пока фильтр не окажется пустым

**Построение начальных маршрутов с учетом фиксации точек**

На входе алгоритма список точек, у некоторых из них уже заполнен ненулевой признак gf=номеру маршрута

1. Выбираем 1ый маршрут с зафиксированной точкой
2. Выполняем процедуру встраивания между складом и первой зафиксированной точкой в маршруте
3. Последовательно выполняем процедуру встраивания между j и j+1 зафиксированными точками
4. Как только загрузка ТС превышает 90% (настраиваемый параметр) – выполняем процедуру встраивания между последней зафиксированной точкой и складом

**Оптимизация маршрутов**

После начального формирования маршрутов переходим к фазе их оптимизации, как описано в алгоритме 2.0

При этом добавляется новое условие в фильтре – признак фиксации gf, т.е. эти точки остаются привязанными к конкретному маршруту – их исключаем из фильтра при поиске точек в близком кластере

**Версия 2.0 от 13.11.13**

**Алгоритм формирования оптимальных маршрутов**

1. **Формирование начальных маршрутов**

Происходит как в пред. версии

Изменения по сравнению с пред. Версией

* в текущей версии поиск ближайшей точки идет не по направлению «от склада/к складу», а во всей окрестности точки (фильтр по координатам убран)
* изменено условие, момент приезда - попадание во временную рамку за 0,5 часа до ее окончания

1. **Оптимизация маршрутов**

После того, как маршруты сформированы, идет поиск возможностей их улучшения

1. Для поиска близких маршрутов находим «центры» каждого маршрута по формулам

хср=сумма(х(i),i=1..ni)/ni

yср=сумма(y(i),i=1..ni)/ni

ni – кол-во точек в маршруте i

Далее сравниваем маршруты попарно:

1. Находим 2 самых близких маршрута, с минимальным расстоянием между их центрами
   1. Из них находим маршрут с наименьшим суммарным расстоянием и начинаем его улучшать, последовательно убирая по одной точке и добавляя одну точку из 2 го маршрута, и подавая этот массив на вход алгоритма1 – формирование начального маршрута
   2. При улучшении 1-го маршрута также следим за улучшением суммарного маршрута 2 ТС – если суммарный путь не улучшается, этот вариант пропускаем
   3. Количество циклов прогона п.2.1. – параметр, задается заранее (зависит от скорости обработки данных)
2. После нахождения маршрута с кратчайшим суммарным путем, исключаем этот маршрут из обработки, для 2-го оставшегося маршрута пересчитываем его центр, находим ближайший к нему по центру маршрут, и повторяем шаги 2.1-2.3
3. После обработки всех маршрутов повторяем цикл 2.-3. необходимое число раз (количество повторений – параметр, зависит от скорости вычислений)

На выходе алгоритма получаем набор маршрутов с улучшенным суммарным расстоянием (степень улучшения зависит от количества прогонов по шагам 2-3

**Версия 1.0**

**Алгоритм формирования оптимальных маршрутов (базовый)**

Входными данными является массив заказов в виде

(x,y,z,t1,t2,t3)

x ,y – координаты точки

z – объем заказа (в шт. бутылок)

t1,t2 – временные рамки принятия заказа, от и до

t3 – время обслуживания заказа на месте (подъем и пр.)

N – кол-во заказов

**Предподготовка данных**

1. Данные нормируются, чтобы получить нормированные координаты всех точек в квадрате [-1,1], при этом склад имеет координаты (0,0)
2. В массив добавляется точка (0,0) – склад
3. Считается матрица расстояний от точки i до точки j, i=0..N, j=0..N, r(i,j)

**Базовый алгоритм**

Алгоритм построен по принципу «гарантированное обслуживание максимально далеких и максимально срочных заказов».

Второй принцип – движение по направлениям. Выделяются четыре направления, нумеруются по квадрантам

1 – x>0, y>0

2 – x<0, y>0

3 – x<0, y<0

4 – x>0, y<0

Алгоритм

1. Начинаем формировать k-ый маршрут. Добавляем в пустой список точку (0,0)
2. Находим максимально удаленную точку с самой минимальной временной рамкой, с индексом itek, заносим ее в список точек
3. Формируем:
   1. суммарный путь маршрута rsum=0+r(0,itek)
   2. суммарное время в пути tsum1=0+r(0,itek)/vsr, где vsr – средняя скорость
   3. суммарное время с учетом обслуживания tsum2=tsum1+t3(itek)
   4. суммарная загрузка zsum=0+z(itek)
4. определяем направление - (1,2,3,4), дальше будем идти в противоположном направлении, т.е. «к складу», т.е. если точка лежит в направлении 1 x>0, y>0, пойдем в направлении 3 x<0, y<0
5. Выбираем те точки (фильтр), которые
   1. Лежат в направлении «к складу»
   2. Лежат во временной рамке t2>tsum2
   3. Могут быть выбраны по объему z(i)+zsum<zmax, zmax – максимальная загрузка ТС
6. Из выбранных точек выбираем самую близкую к точке itek, min r(imin,itek)
7. Формируем:
   1. суммарный путь маршрута rsum=rsum+r(itek,imin)
   2. суммарное время в пути tsum1=tsum1+r(itek,imin)/vsr, где vsr – средняя скорость
   3. суммарное время с учетом обслуживания tsum2=tsum1+t3(imin)
   4. суммарная загрузка zsum=zsum+z(imin)
8. текущей точкой делаем найденную itek=imin, заносим в список точек
9. переходим к шагу 5
10. если в шаге 5 точек больше не находится (пустой список), меняем направление на противоположное, и опять выполняем шаги 5-9.
11. если в двух направлениях не найдены точки, переходим на два оставшихся (1,3)-(2,4)
12. выполняем цикл 5-9 либо пока ТС не загружено zsum=zmax, либо пока не осталось нераспределенных точек
13. если ТС загружено, дополняем список точек последней точкой (0,0), считаем rsum, tsum1, tsum2, zsum
14. формируем следующий маршрут, возращаемся на шаг 1