No More Guess Work: System Design for Seamless E-Commerce & Store Order Profiles

Sponsored by:

Reddwerks Distribution Science™

Presented by:

Alex Ramirez,

VP National Projects

ProMat 2013
AN MHIA INTERNATIONAL EXPO
McCORMICK PLACE CHICAGO JAN 21-24
www.ProMatShow.com

MHI

Order Profile Defined

Order Profiles: A collection of attributes associated to an order that describes how the order is composed and what fulfillment needs it requires.

The constraints imposed by poorly designed software and rigidly-designed storage material handling devices exacerbate the issue to process varied order profiles.

↑ Cost/Unit Shipped ↓ Throughput ↓ Capacity

Composition

- Number of Lines
- Number of Units
- Velocity Mix of Items
- Affinities of Items

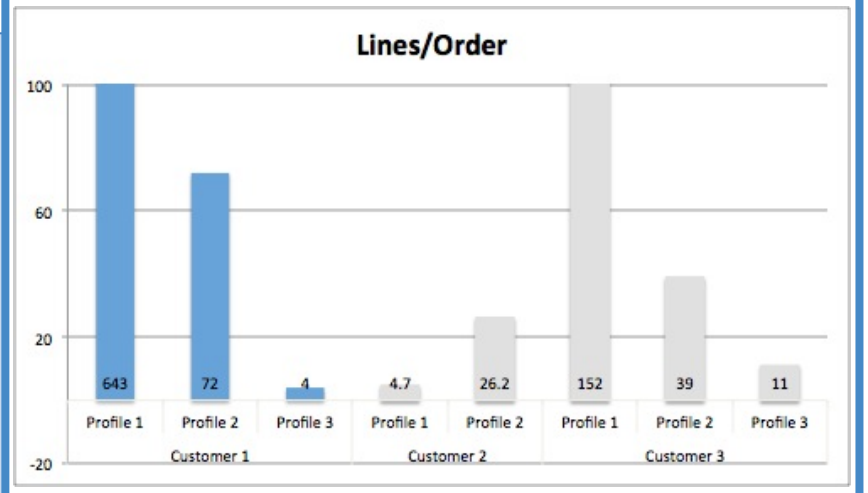
Fulfillment Requirements

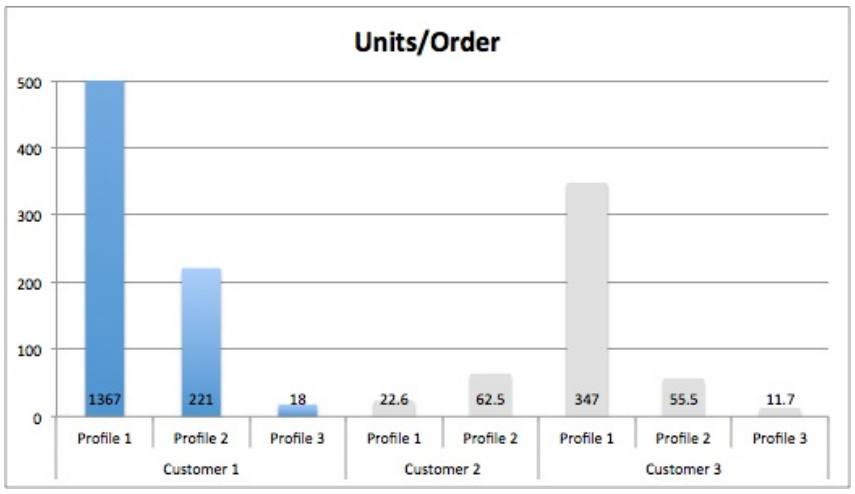
- Dispatch Times
- QC Thresholds
- Value-Added Services
- HAZMAT

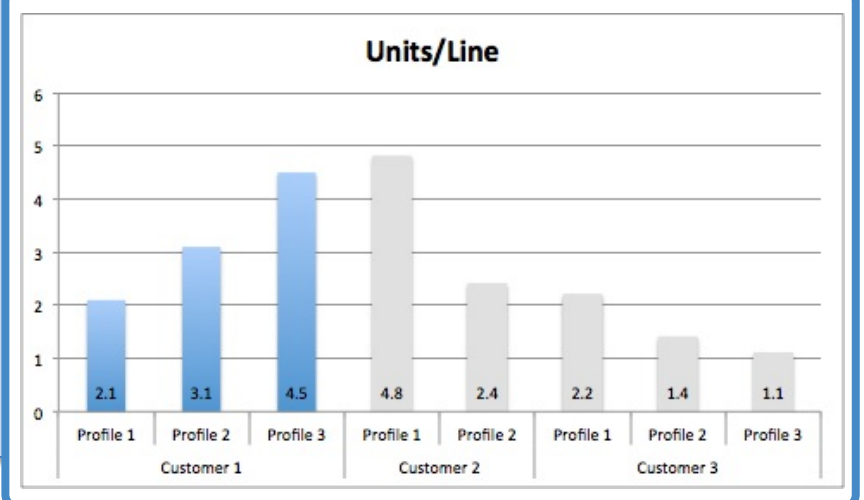
Order Profiles Through Data Models

Customer 1

- Consumer Electronics
- $ 6 Billion Revenue
- Stores, Kiosks
- Order Profiles
  - Brick and Mortar Store Replenishment
  - Mobile Stores
  - Kiosk/Express Stores



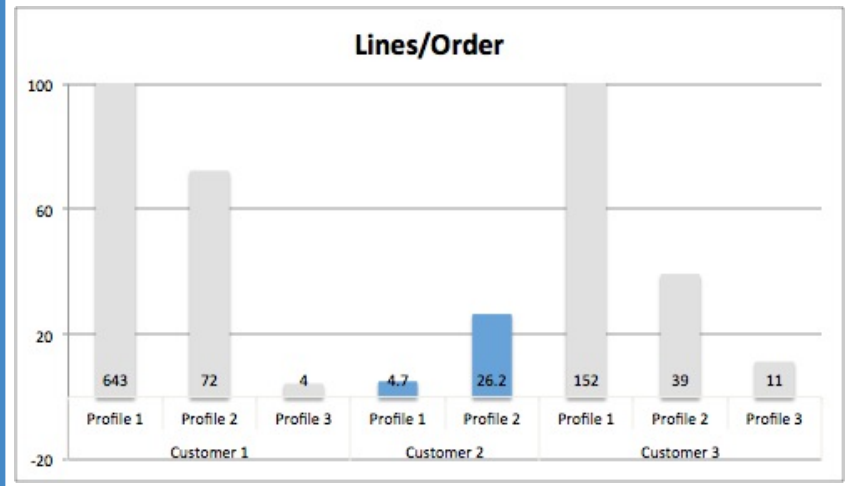


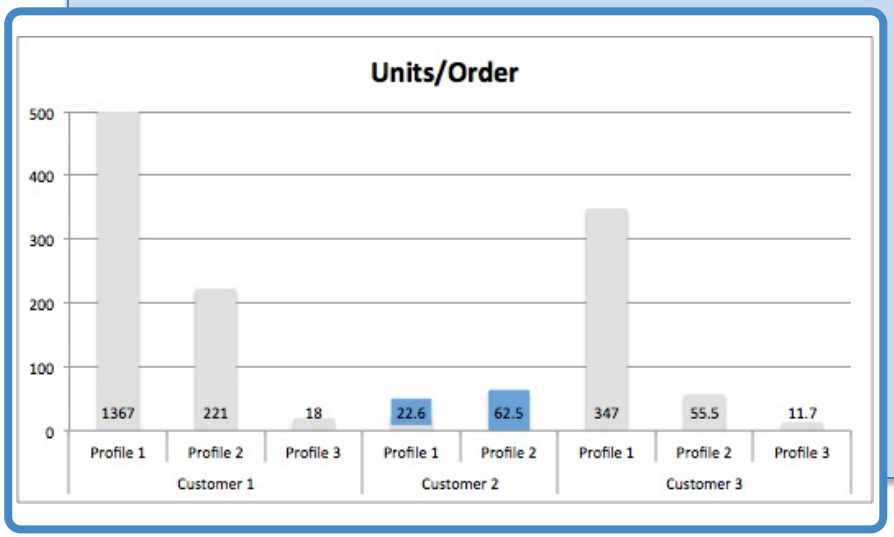


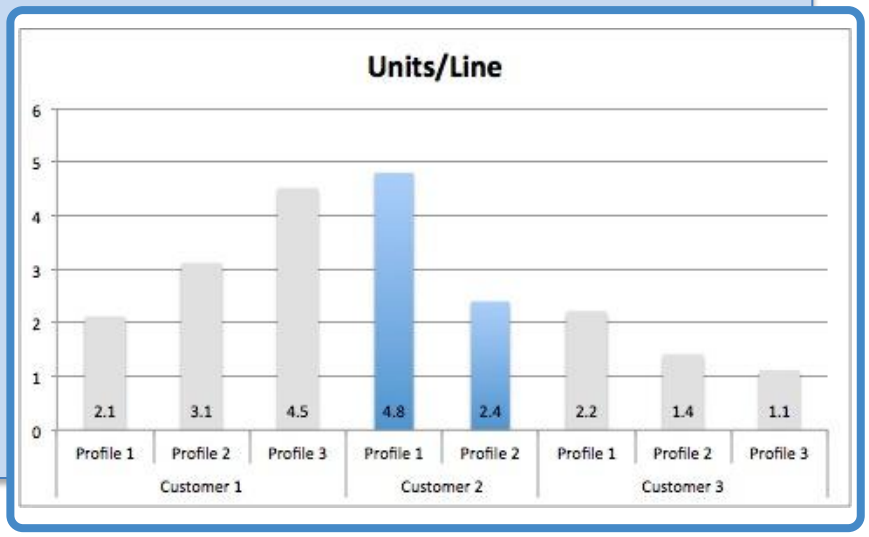
Order Profiles Through Data Models

Customer 2

- Apparel 3PL
- $14 Billion Revenue
- Wholesale, Retail
- Order Profiles
  - Retail Floor sets
  - Retail vs. Wholesale, “Push vs. Pull”
  - Jewelry



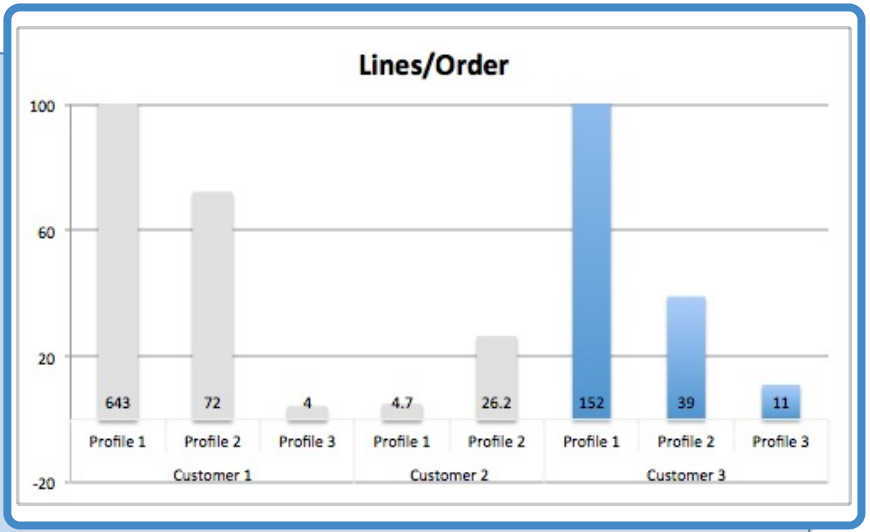


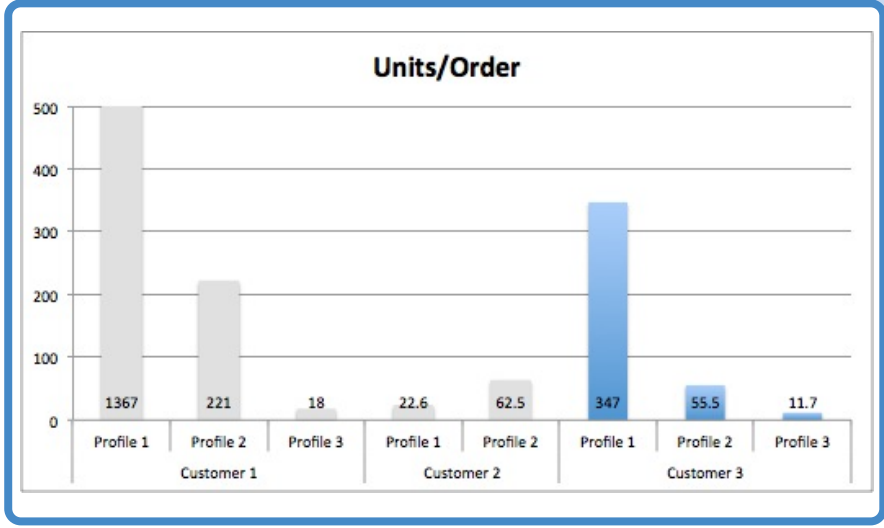


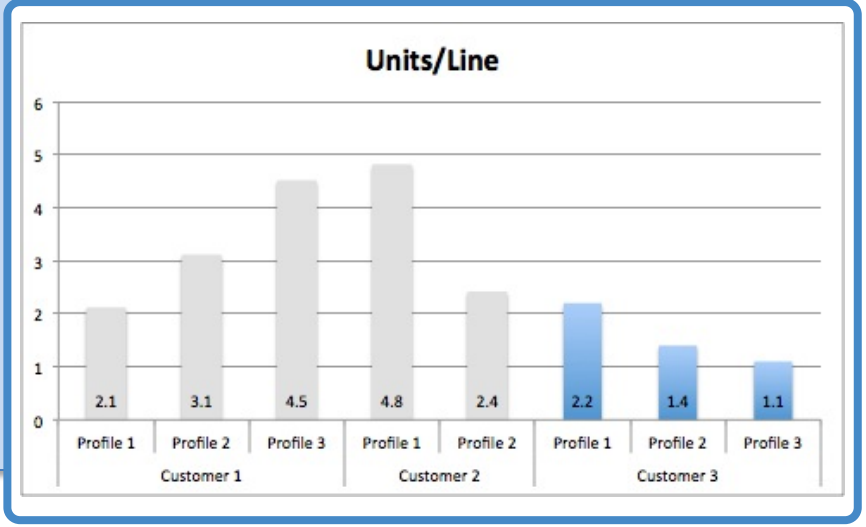
Order Profiles Through Data Models

Customer 3

- Wholesale Distributor
- $ 38 Billion Revenue
- C-Stores/Big Box/Store types
- Order Profiles
  - Split Case & Cigarette Orders
  - Full Case Orders
  - Non Conveyable & Pallet Orders



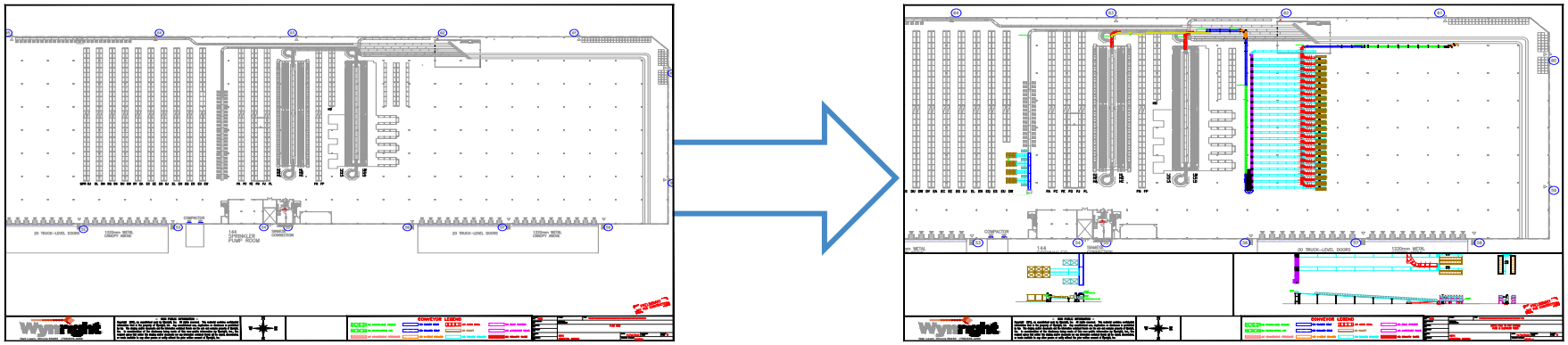




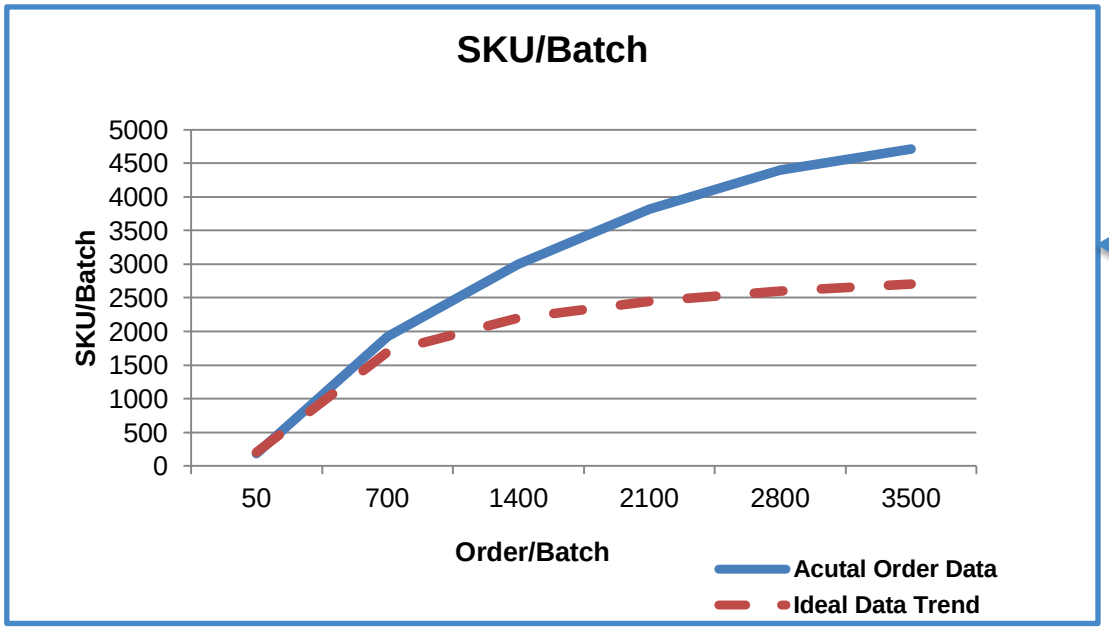
Option A: Put System, Goods-to-Person

Conversion of a Pick System to a Batch Pick to Put System

- Why did G2P work here?
  - Larger store orders in Canada yielded higher percent of full case picks
  - Fewer mixed SKU totes in Canada yielded higher put rates at put stations
  - Outbound totes in Canada were reduced because of totes having to complete in the pick zone where they are started
- Benefits:
  - All but 37 days a year slashed to single-shift days (from two shifts)
  - Flexible, extensible design to enable the site to process web orders and other channels' changing order profiles.
  - Yearly cost reduction over $1M.



Case Study: Pick System, Person-to-Goods



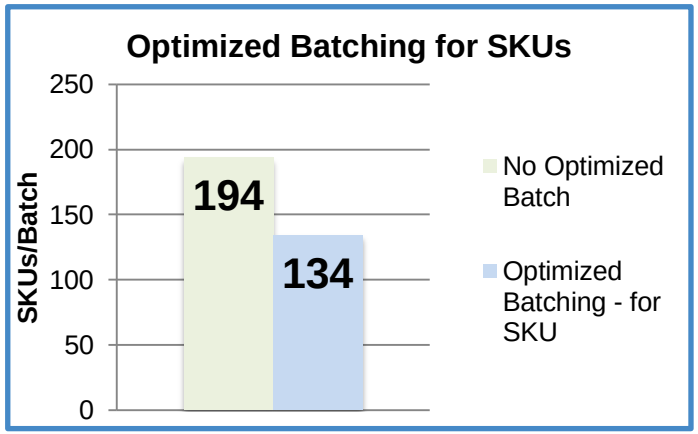
The delta represents the additional residual that would need to be picked per batch

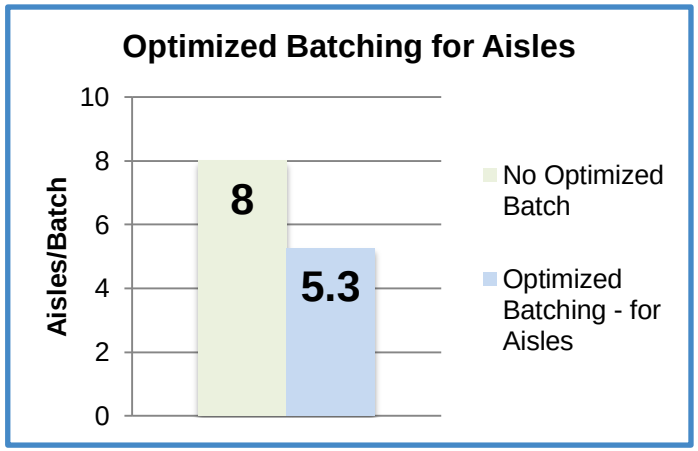
SKU Velocity & Walk Time are critical components in analyzing a Goods to Person Design. **Result = Handling each item at the Put Station adds 7.5 seconds per item** that currently doesn't occur today. With the current walk time per unit being 6 seconds, **you cannot "pay" for the put.**

Case Study: Pick System, Person-to-Goods

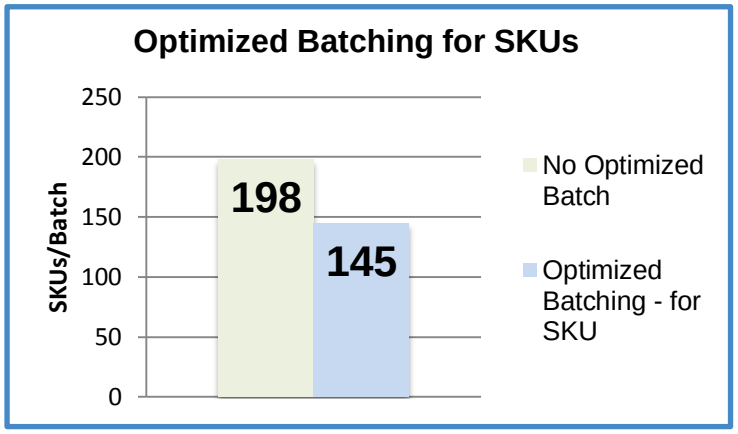
Optimized Batching Algorithms group orders by key attributes allowing for less walk time in Break Pack picking.

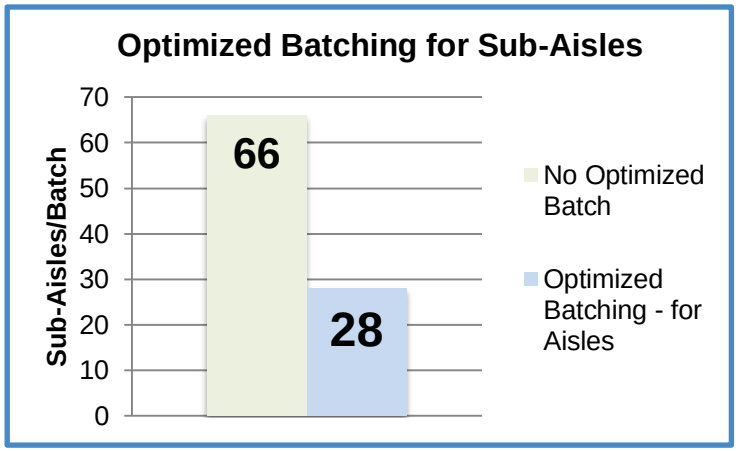
Site 1





Site 2





Don’t Guess…Guessing is Expensive

Objective: To reduce the standard deviation of units per put station.

$$Min \sum_{s \in S} \left(\sum_{o \in O_s} u_o - m_u \right)^2$$

Subject to:

$$\sum_{s \in S} \sum_{o \in O_s} u_o = u_{tot}$$

$$m_u = \left\lfloor \frac{u_{tot}}{n_{stn}} \right\rfloor$$

$$\sum_{s \in S} \sum_{o \in O_s} 1 = o_{tot}$$

$$\sum_{o \in O_s} 1 = \left\lfloor \frac{o_{tot}}{n_{stn}} \right\rfloor \quad \forall s \in S$$

Where:

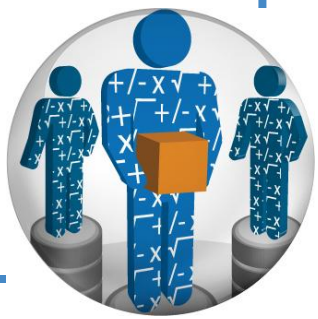
$$u_o \geq 0 \ \forall o \in O_s \ \forall s \in S$$
$$u_o \ integer$$

Where:

S is the set of stations
O_s is the set of orders in station s
u_o is the number of units in order o
u_{tot} is the total number of units
n_{stn} is the number of active stations
o_{tot} is the total number of orders

```
Batch batchObj = Batch.readInputToBatchObj(getCoreToPlanningFilePath(CORE_TO_PLANNING_ORDER_FILE_PREFIX));

Map<String, Order> orderMap = new TreeMap<String, Order>(batchObj.getOrderMap());

Map<String, Integer> ncOrderCartonMap = new HashMap<String, Integer>();
Map<String, Integer> wjOrderCartonMap = new HashMap<String, Integer>();
Map<String, Integer> fcOrderCartonMap = new HashMap<String, Integer>();
Map<String, Integer> ptsOrderCartonMap = new HashMap<String, Integer>();

System.out.println("Total units: " + batchObj.getTotalUnits(orderMap));
Map<String, Order> nonConOrders = batchObj.removeNonCon(orderMap);
System.out.println("Non Con units: " + batchObj.getTotalUnits(nonConOrders) + "\tCon Units: " + batchObj.getTotalUnits(orderMap));

for(String ordID: nonConOrders.keySet()) {
        int numCases = 0;
        for(String pickID: nonConOrders.get(ordID).getPickIDMap().keySet()) {
                int numFullCases = nonConOrders.get(ordID).getPickIDMap().get(pickID).getQty() / nonConOrders.get(ordID).getPickIDMap().get(pickID).getFullCaseQty();
                int numEaches = nonConOrders.get(ordID).getPickIDMap().get(pickID).getQty() % nonConOrders.get(ordID).getPickIDMap().get(pickID).getFullCaseQty();
                numCases +=  numFullCases + numEaches;
        }
        ncOrderCartonMap.put(ordID, numCases);
}

System.out.println("Total units: " + batchObj.getTotalUnits(orderMap));
Map<String, Order> whJwlOrders = batchObj.removeJwl(orderMap,"W");
System.out.println("Wh Jwl units: " + batchObj.getTotalUnits(whJwlOrders) + "\tNon Wh Jwl Units: " + batchObj.getTotalUnits(orderMap));

for(String ordID: whJwlOrders.keySet()) {
        wjOrderCartonMap.put(ordID, whJwlOrders.get(ordID).getExpNumCartons());
}

System.out.println("Total units: " + batchObj.getTotalUnits(orderMap));
Map<String, Order> fullCasePickOrders = batchObj.removeFCPicks(orderMap);
System.out.println("FC Pick units: " + batchObj.getTotalUnits(fullCasePickOrders) + "\tNon FC Pick Units: " + batchObj.getTotalUnits(orderMap));

System.out.println("Total units: " + batchObj.getTotalUnits(orderMap));
Map<String, Order> earlyPickOrders = batchObj.removeEarlyOrders(orderMap);
System.out.println("Early Pick units: " + batchObj.getTotalUnits(earlyPickOrders) + "\tRemaining Non FC Pick Units: " + batchObj.getTotalUnits(orderMap));

List<String> earlyPickIDs = new LinkedList<String>();
for(String ord: earlyPickOrders.keySet()) {
        earlyPickIDs.addAll(earlyPickOrders.get(ord).getPickIDMap().keySet());
}
for(String ordID: fullCasePickOrders.keySet()) {
        int numCases = 0;
        for(String pickID: fullCasePickOrders.get(ordID).getPickIDMap().keySet()) {
                numCases +=  fullCasePickOrders.get(ordID).getPickIDMap().get(pickID).getQty()/fullCasePickOrders.get(ordID).getPickIDMap().get(pickID).getFullCaseQty();
        }
        fcOrderCartonMap.put(ordID, numCases);
}

for(String ordID: orderMap.keySet()) {
        ptsOrderCartonMap.put(ordID, orderMap.get(ordID).getExpNumCartons());
}
```

1. Smooth work across put stations
2. Maximize throughput
3. Reduce cost/unit shipped



PROMAT 2013
AN MHIA INTERNATIONAL EXPO
McCORMICK PLACE CHICAGO JAN 21-24
www.ProMatShow.com

Reddwerks Distribution Science™

Speaker: Aramirez@reddwerks.com

www.reddwerks.com

Visit Promat 2013 Booth # 4281