Lot-Sizing of Several Multi-Product Families

by

Tiffany Bayley

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Management Sciences

Waterloo, Ontario, Canada, 2018

Examining Committee Membership

The following served on the Examining Committee for this thesis. The decision of the Examining Committee is by majority vote.

External Examiner:	Dr. Raf Jans, Professor Dept. of Logistics and Operations Management, HEC Montreal
Supervisor:	Dr. James H. Bookbinder, Professor Dept. of Management Sciences, University of Waterloo
Internal Members:	Dr. Samir Elhedhli, Professor Dept. of Management Sciences, University of Waterloo
	Dr. Fatma Gzara, Associate Professor Dept. of Management Sciences, University of Waterloo
Internal-External Member:	Dr. Paul Calamai, Professor Dept. of Systems Design Engineering, University of Waterloo

Author's Declaration

Statement of Contributions

The work in Chapter 2 was co-authored with Dr. James H. Bookbinder and Dr. Haldun Süral and has been published in *International Journal of Production Research* (Bayley et al., 2017). I was the first author responsible for defining the research problem, model formulation, computational analysis, and preparing the manuscript and its revisions.

Abstract

Production planning problems and its variants are widely studied in operations management and optimization literature. One variation that has not garnered much attention is the presence of multiple production families in a coordinated and capacitated lot-sizing setting. While its single-family counterpart has been the subject of many advances in formulations and solution techniques, the latest published research on multiple family problems was over 25 years ago (Erenguc and Mercan, 1990; Mercan and Erenguc, 1993).

Chapter 2 begins with a new formulation for this coordinated capacitated lot-sizing problem for multiple product families where demand is deterministic and time-varying. The problem considers setup and holding costs, where capacity constraints limit the number of individual item and family setup times and the amount of production in each period. We use a facility location reformulation to strengthen the lower bound of our demand-relaxed model. In addition, we combine Benders decomposition with an evolutionary algorithm to improve upper bounds on optimal solutions. To assess the performance of our approach, single-family problems are solved and results are compared to those produced by state-of-the-art heuristics by de Araujo et al. (2015) and Süral et al. (2009). For the multi-family setting, we first create a standard test bed of problems, then measure the performance of our heuristic against the SDW heuristic of Süral et al. (2009), as well as a Lagrangian approach. We show that our Benders approach combined with an evolutionary algorithm consistently achieves better bounds, reducing the duality gap compared to other single-family methods studied in the literature.

Lot-sizing problems also exist within a vendor-managed-inventory setting, with production-planning, distribution and vehicle routing problems all solved simultaneously. By considering these decisions together, companies achieve reduced inventory and transportation costs compared to when these decisions are made sequentially. We present in Chapter 3 a branch-and-cut algorithm to tackle a production-routing problem (PRP) consisting of multiple products and customers served by a heterogeneous fleet of vehicles. To accelerate the performance of this algorithm, we also construct an upper bounding heuristic that quickly solves production-distribution and routing subproblems, providing a warm-start for the branch-and-cut procedure. In four scenarios, we vary the degree of flexibility in demand and transportation by considering split deliveries and backorders, two settings that are not commonly studied in the literature. We confirm that our upper bounding procedure generates high quality solutions at the root node for reasonably-sized problem instances; as time horizons grow longer, solution quality degrades slightly. Overall costs are roughly the same in these scenarios, though cost proportions vary. When backorders are not allowed (Scenarios 1 and 3), inventory holding costs account for over 90% of total costs and transportation costs contribute less than 0.01%. When backorders are allowed (Scenarios 2 and 4), most of the cost burden is shouldered by production, with transportation inching closer to 0.1% of total costs.

In our fifth scenario for the PRP with multiple product families, we employ a decomposition heuristic for determining dedicated routes for distribution. Customers are clustered through *k*-means++ and a location-alloction subproblem based on their contribution to overall demand, and these clusters remain fixed over the entire planning horizon. A routing subproblem dictates the order in which to visit customers in each period, and we allow backorders in the production-distribution routine. While the branch-and-cut algorithm for Scenarios 1 through 4 quickly finds high quality solutions at the root node, Scenario 5's dedicated routes heuristic boasts high vehicle utilization and comparable overall costs with minimal computational effort.

Acknowledgements

This thesis would not have been possible without the continual motivation and support from those around me. I thank my advisor, Dr. Jim Bookbinder, for his constant encouragement over the years and remaining positive throughout the many turns this dissertation took. I am grateful to my committee members, Professors Raf Jans, Paul Calamai, Samir Elhedhli, and Fatma Gzara, and past committee members Professors David Fuller and Osman Ozaltin for their insightful suggestions that improved this work.

Thank you to Ada Hurst and Jennifer Engels, and their families, for their friendship over the past 10 years. I am indebted to Bissan Ghaddar and Joe Naoum-Sawaya for their generosity of time and knowledge that pushed me forward during the final stages of this degree.

The love and patience from my family is without measure and I cannot express in enough words how fortunate I am to have had their support throughout this journey. Thank you, thank you, thank you.

Matt – We did it! You are the reason this milestone has been reached. Your unwavering belief in me kept me going, even when I doubted myself the most. Together, we overcame some unexpected obstacles, and I know that the strength we gained from these challenges will help us breeze by whatever life throws us next. I love you, Dean, and Joan with all my heart.

Table of Contents

List of Tables

List of Figures

List of Routines

Chapter 1

Introduction

In manufacturing or retail environments, it is typical for certain products to share resources, such as a common machine that stamps sheets of metal or a supplier filling a trailer with items for shipment. Rather than schedule production (or place orders) on an individual basis, items may be grouped into a family and produced (or purchased) together to save on setup times or administrative costs. The coordinated, capacitated multi-item lot-sizing problem (CCLSP) captures this concept by incurring a major setup time whenever any item of a family is produced, and a minor setup time for each item that undergoes production in that period. The objective is to minimize total costs of setting up families of items and holding inventory, while satisfying demand and respecting capacity constraints in every period over a planning horizon.

CCLSP pertains mostly to a single family of items, but *multiple* product groups exist in chemical processing or the manufacture of industrial (versus commercial) items (Mohamed et al., 2004; Karalli and Flowers, 2006). In an automotive setting, the paint process also exhibits this multiple family structure. The limiting resource is time available in the paint shop, while variety of colours dictates the number of product families. Each family comprises a number of items to be painted, such as passenger doors and the frame of the car. Time required to change paint colours is analogous to incurring a major (or family) setup time, and adjusting the positioning of items before painting is similar to a minor (or item) setup time. Consolidation of shipments can be viewed from this perspective as well, as outgoing loads (or orders, consisting of one or many products) are grouped according to customers (families) and shipped on one or more vehicles. For this case, the objective would be to determine the grouping of loads that minimize shipment time or cost, or maximize service levels to customers, or both.

Item characteristics generally dictate family groups: products sharing a machine or a trailer will naturally be "related," and so should their production or shipment schedules. When definitions of families are not as clear, how items are grouped could influence production and shipment patterns. The production-routing problem (PRP) considers lot-sizing, distribution, and vehicle routing decisions simultaneously, usually for a single production facility carrying out all three operations in a vendor-managed inventory setting. It exhibits groupings at both the production and routing stages, and these groupings are not necessarily the same. Depending on aggregate and individual customer demands over the horizon, a vendor ensures that sufficient inventory is maintained at the customer level, and can also coordinate its production and deliveries in such a way that manufacturing and vehicle capacities are efficiently used. It is, therefore, important that the vendor carefully determines how best to coordinate the production and distribution of products to avoid manufacturing and shipping redundancies, and this includes defining the appropriate item clusters.

1.1 An Overview of Lot-Sizing Problems

Lot-sizing problems determine *when* and *how much* of an item, or number of items, to replenish so that demand is satisfied while minimizing total cost. Customer orders dictate demand, which can be deterministic (stationary or time-varying) or probabilistic, and typical costs are attributed to setups and holding inventory. Additional features, such as capacity constraints, time windows or families of products, can greatly complicate the solution of these problems.

The uncapacitated single-item lot-sizing problem with time-varying demand has parameters C_t, H_t, Q_t, and d_t denoting unit costs of production and holding per period, fixed setup cost and demand per period, respectively, where $t = 1, 2, ..., n$. Variables x_t and s_t represent the number of units produced in period t and the number of units carried over from period t to $t+1$. As well, binary variables y_t denote whether a setup for an item occurs in period t.

$$[LSU] \quad \min \quad \sum_{t=1}^{n} \Big(C_t x_t + Q_t y_t + H_t s_t \Big) \tag{1.1}$$

$$\text{s.t.} \qquad s_{t-1} + x_t = s_t + d_t \qquad \forall t \tag{1.2}$$

$$x_t \leq M y_t \qquad \forall t \tag{1.3}$$

$$x_t, s_t \geq 0 \qquad \forall t \tag{1.4}$$

$$y_t \in \{0,1\} \qquad \forall t \tag{1.5}$$

The objective function minimizes the sum of production, setup, and holding costs, while ensuring that demand is met in every period (1.2) and that items are produced only if a setup is incurred (1.3). It is typical to assume initial and ending inventories are zero ($s_0 = s_n = 0$).

When multiple items, $i = 1,2,...,m$, are considered, the formulation changes only slightly to account for different items being produced. All decision variables and parameters will now have an added subscript, i:

$$[M-LSU] \quad \min \quad \sum_{i=1}^{m}\sum_{t=1}^{n}\Big(C_{it}x_{it}+Q_{it}y_{it}+H_{it}s_{it}\Big) \tag{1.6}$$

$$\text{s.t.} \qquad s_{i,t-1}+x_{it} = s_{it}+d_{it} \qquad \forall i,t \tag{1.7}$$

$$x_{it} \leq My_{it} \qquad \forall i,t \tag{1.8}$$

$$x_{it}, s_{it} \geq 0 \qquad \forall i,t \tag{1.9}$$

$$y_{it} \in \{0,1\} \qquad \forall i,t \tag{1.10}$$

Here, we are still minimizing the total production, setup, and holding costs for all items i and over all periods t. Decision variables x_{it}, s_{it}, and y_{it} now respectively denote the amount of product i produced in t, number of units of i held from period t until $t+1$, and whether or not to produce item i in period t. Demand, production cost, setup cost, and holding cost for each item i in period t are respectively denoted by d_{it}, C_{it}, Q_{it}, and H_{it}.

Constraints on production capacity can be added to this formulation in terms of either time or costs required to set up an item. While setup costs can be estimated using an employee's hourly wage and the time required to physically prepare the machine for production, it may not be an accurate reflection of the actual cost to the company, since that same employee would have been paid whether that machine needed attention or not. It maybe just as beneficial to examine setup times, as these can be measured (and even improved upon), and are just as easily incorporated into the model as costs. Letting a_{it} denote the unit time required to produce item i in period t, b_{it} denote the setup time for production of i in period t, and CAP_t be the total time available for production in period t, we can add the following capacity constraint to $[M-LSU]$:

$$\sum_{i=1}^{m}\left(a_{it}x_{it}+b_{it}y_{it}\right)\leq CAP_t \qquad \forall t \tag{1.11}$$

The deterministic coordinated lot-sizing problem has been widely studied. Zangwill's research on multiple product production and inventory models (1966) has since inspired various methods for determining the number and timing of items to be ordered in a way that will minimize system costs. Some of these methodologies include dynamic programming (Kao, 1979), shortest path reformulations (Joneja, 1990), and primal-dual solution algorithms (Kirca, 1995).

Boctor et al. (2004) describe properties of an optimal solution for the uncapacitated coordinated lot-sizing problem $[LSU]$. Using results from the seminal work of Wagner and Whitin (1958), as well as from Silver (1979), Boctor et al. (2004) indicate that:

1. production for an item takes place only when there is no inventory available for that item (or $x^*_{it}\times I^*_{it-1}=0$)
2. the optimal production quantity in a period will fulfill requirements *exactly* for the periods of demand that it covers (i.e., $x^*_{it}=d_{it},d_{it}+d_{i,t+1},\ldots,\sum_{q=t}^{T}d_{iq}$)
3. the optimal inventory level in a period will fulfill requirements *exactly* for the periods of demand that it covers (i.e., $I^*_{it-1}=d_{it},d_{it}+d_{i,t+1},\ldots,\sum_{q=t}^{T}d_{iq}$); and
4. if the cost to hold d_{ir} over an interval $[t,r-1]$ exceeds the setup costs for that item in period r (i.e. $d_{ir}\sum_{t'=t}^{r-1}h_{it'}>q_r$), then production for item i should not take place in period t.

For capacitated (but not coordinated) multi-item lot-sizing problems, the Dixon-Silver heuristic (1981) follows the framework of Silver and Meal (1973) by determining the span of periods for which average setup and production costs is at its minimum, while considering capacity absorption. Unfortunately, a similar heuristic for coordinated *and* capacited lot-sizing problems does not yet exist: making assumptions about the periods in which family setups will take place will greatly impact the remaining capacity available for production, thus creating challenges when adapting this methodology to include additional problem features.

In the literature review that follows, we describe how the coordinated capacitated lot-sizing problem has evolved in terms of its formulations, as well as the solution approaches commonly used to solve them. This survey indicates the potential for solving the multi-family CCSLP with both exact and metaheuristic methods. We conclude with a discussion of research related to production-routing problems, and identify formulation characteristics that deserve more attention, especially in a multiple product-family setting.

1.2 Literature Review

1.2.1 Coordinated Capacitated Lot-Sizing Problems

Trigeiro et al. (1989) provide a foundation for much of the research discussed here. Those authors formulate a capacitated multi-item lot-sizing problem subject to setup costs and setup times, using a Lagrangian heuristic to relax the capacity constraints and decompose the problem by item, to ultimately obtain a lower bound to the optimal solution. To find an upper bound, Trigeiro et al. (1989) develop a smoothing heuristic (TTM) that begins from the Lagrangian subproblem solution and shifts production lots to different periods until a feasible solution is found. That smoothing heuristic, however, does not always succeed at finding feasible solutions, and in those cases, a different initial solution is required.

Li et al. (2012) develop a three-stage approach for a multi-item capacitated lot-sizing problem (LSP). After some pre-processing steps, their heuristic improves upon an initial set of solutions by solving a series of two-item subproblems. This configuration significantly reduces the number of binary variables, though does not optimize production plans for all items simultaneously. While those authors avoid the need for reformulating the LSP, taking this extra step may provide enhanced results.

The TTM heuristic is used in Hindi et al. (2003), who reformulate the CCLSP as a transshipment model and employ variable neighbourhood search to improve the upper bound (see Section 1.2.1 for more on variable neighbourhood search). However, the same issue remains, that the smoothing heuristic may or may not succeed at finding a feasible solution.

Strong Reformulations

$[LSU]$ presented in the previous section is also known as the classical, or *weak*, lot-sizing formulation. Inventory-balance constraints (1.2) establish a link between time periods characterized by given demands, production quantities, and amounts held in inventory, as depicted in Figure 1.1. These links become more interwined as problem sizes grow (either in number of items or time periods), making this representation of the formulation more difficult to solve directly.

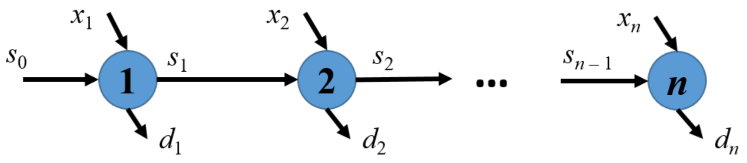

Figure 1.1: Inventory-Balance Constraint for a Weak Formulation

A strong reformulation exploits characteristics of the problem structure to develop models whose LP relaxations produce solutions that are equal to the optimal solution to the weak formulation. Pochet and Wolsey (2006) describe two common reformulations of the single-item uncapacitated lot-sizing problem $[LSU]$: the shortest-path-type (SP) and facility-location-type (FL) (or transportation-type) formulations. Though these each require a redefinition, and thus an increase in the number of production decision variables, their LP-relaxation solutions are identical to one another (Denizel et al., 2008; Solyalı and Süral, 2012). Figure 1.2 shows how the FL reformulation creates a bipartite graph, with u representing periods in which production takes place and t indicating periods in which demand is required. Variables w_{ut} then represent the fraction of d_t produced in u to satisfy demand in t, and $x_u = \sum_{t=u}^{n} d_t w_{ut}$. For the uncapacitated case, the SP and FL formulations exhibit the unimodularity property, and hence their LP relaxations provide integer solutions that are optimal for $[LSU]$.

Reformulations of the CCLSP are studied by Jans and Degraeve (2004) and Süral et al. (2009), who use an SP and FL reformulation, respectively. Solving the linear relaxation of these reformulations results in better linear lower bounds when compared to the typical lot-size-inventory formulation in Trigeiro et al. (1989). Rather than relaxing capacity constraints, these formulations allow demand constraints to be dualized. The structure of the Lagrangian subprob-

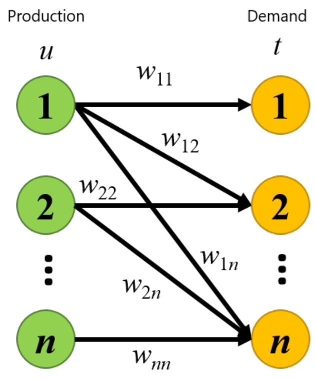

Figure 1.2: Demand Constraint for a (strong) FL Reformulation

lem is such that unimodularity is removed and a better lower bound (than that of the LP-relaxation of the reformulation) is attainable. See Denizel et al. (2008), Gao et al. (2008), and Solyalı and Süral (2012) for further comparisons of the shortest path and transportation reformulations for both the multi-item and coordinated lot-sizing problems.

Metaheuristics in Lot-Sizing Problems

Even with strong formulations, heuristic methods are often needed to find good solutions in a reasonable amount of time. Simulated annealing, genetic algorithms, and variable neighbourhood search have been used to solve multi-item capacitated lot-sizing problems and its variants (Hindi et al., 2003; Jans and Degraeve, 2007; Goren et al., 2010).

Tabu search and simulated annealing are local search metaheuristics that allow some worsening of the objective function value in an attempt to escape a local optimum. In tabu search (TS), a list of previously visited solutions is maintained; returning to those solutions in the list is prohibited, at times forcing the local search to move to areas with less desirable objective function values (Gendreau and Potvin, 2010). With simulated annealing (SA), moving to an improved solution is always accepted, though inferior solutions are at times allowed with a certain probability (Nikolaev and Jacobson, 2010).

Narayanan and Robinson (2010) use a six-phase heuristic to initialize their simulated annealing procedure. Each phase shifts lots either to previous or future periods, attempting to reduce setup and holding costs or else restore feasibility due to the previous phase's adjustments. The fact that the CCLSP has only one family of items means that family setup time can be omitted from the problem by decreasing the capacity accordingly in each period. This, however, cannot be extended directly to a multiple family setting.

Both tabu search and simulated annealing explore a neighbourhood of the solution space, one solution at a time. Evolutionary algorithms (EAs), such as the genetic algorithm, use a population of solutions to quickly search a neighbourhood (Reeves, 2010). Solutions, or a representation of them, follow a sort of biological process: some solutions or *chromosomes* are *selected* to *crossover* and produce *offspring*, who then go on to *replace* other chromosomes in the population. Mutation may occur along the way, which could help move the local search heuristic away from local optima and closer to the global optimal solution.

While TS, SA and EA all remain in the same neighbourhood of a solution space, variable neighbourhood search (VNS) allows a change in neighbourhood if there ceases to be improvement in the objective function value. In this lot-sizing context, a neighbourhood might be defined as a time interval of length k and a change in neighbourhood could be achieved by varying k ($1 \leq k \leq n$, where n is the maximum number of time periods in the planning horizon). The basic steps for VNS involve

1. *shaking*, or finding a solution in a neighbourhood;
2. improving that solution through *local search*; and
3. *changing neighbourhoods* (Hansen et al., 2010).

Randomness can be incorporated in the shaking portion of the algorithm, but a deterministic approach can also be employed, as in Hindi et al. (2003).

In addition to metaheuristics, Buschkühl et al. (2010) describe mathematical programming and decomposition approaches for the capacitated lot-sizing problem. Recently, de Araujo et al. (2015) present a period decomposition approach that combines column generation (Barnhart et al., 1998) and Lagrangian relaxation (Fisher, 2004) to improve lower bounds on the capacitated lot-sizing problem with setup times. They are able to improve the lower bounds found by Süral et al. (2009), and use the smoothing heuristic from Trigeiro et al. (1989) as the initial starting point for finding an upper bound.

Though these problem settings vary from the traditional CCLSP framework, Bollapragada et al. (2011) and Camargo et al. (2014) incorporate mathematical programming and heuristics together in their *matheuristic* approaches to lot-sizing and scheduling problems on multiple machines. Here, a heuristic (or metaheuristic) is embedded within an exact algorithm to help find solutions more quickly. In both articles, the overarching theme is to make the MIP tractable by fixing some of its integer or binary variables, solving this reduced version of the problem to optimality, and then repairing solutions to achieve feasibility or improvement in the objective function value. Toledo et al. (2016) use a similar partitioning approach, but rather than develop a matheuristic, they compare a fix-and-optimize method to a multi-population genetic algorithm and show that both perform at a similar level on medium to large problem instances.

Benders Decomposition

One technique that has not been very widely applied to lot-sizing is Benders decomposition (Benders, 1962). A lot-sizing problem can be decomposed into a master problem containing only binary setup variables, and a subproblem with the remaining production quantity decision variables. Beginning with an initial setup schedule and fixing those binary variables, the subproblem generates a production plan according to the given schedule, along with optimality and feasibility cuts to pass back to the master problem, where a new setup schedule is determined. Iteration between the subproblem and master problem continues until their solutions converge.

After fixing binary setup variables of the weak formulation, Bahl and Zionts (1987) transform the Benders subproblem into a transportation problem that is more easily solved. They are able to show that their Benders approach does solve reasonably-sized problems to optimality. Aardal and Larsson (1990) also use Benders decomposition for their hierarchical production planning problem, but note that the addition of Benders cuts increases the difficulty of the master problem. To overcome this, those authors apply Lagrangian relaxation to the Benders cuts. This relaxed master problem will now provide only a bound, but will not necessarily converge to the optimal solution.

The works by Bahl and Zionts (1987) and Aardal and Larsson (1990) highlight one particular challenge of using Benders decomposition: the master problem becomes increasingly difficult to solve and requires longer computational times. A recent review by Rahmaniani et al. (2017) discusses many improvements to the Benders algorithm that have been studied to tackle that complication (among others). In particular, for a master problem that has no special structure, there may be value in using metaheuristics to quickly explore neighbourhoods and find pools of feasible solutions. Poojari and Beasley (2009) show that using a genetic algorithm to solve the Benders master problem improves both the lower and upper bounds in the majority of their test problems of varying difficulties.

Other emerging trends in practice consider the conservation of energy and reduction of emissions in production planning. Masmoudi et al. (2017) incorporate the cost of energy in their multi-stage flow line problem, emphasizing the impact of off- and on-peak electricity prices on the cost-effectiveness of production. In our work, the coordination of setups can be viewed as an energy-saving measure: fewer setups will be required over the planning horizon, resulting in lower total setup time when machines may be running but idle. From the shipment consolidation viewpoint, fewer setups mean fewer trucks dispatched, leading to reduced carbon emissions.

Robinson et al. (2009) propose that a logical extension to the body of CCLSP research should include multiple families. To the best of our knowledge, the most recent work on MF-CCLSP was by Erenguc and Mercan (1990) and Mercan and Erenguc (1993), who use a branch-and-bound heuristic applied to the lot-size-inventory formulation. There is therefore an opportunity to develop a stronger formulation and solve it with a hybrid decomposition-metaheuristic method. In this work, we explore these improved solution methods in the MF-CCLSP context, and measure the performance of our approach against those of Süral et al. (2009) and de Araujo et al. (2015).

1.2.2 Production-Routing Problem

While plenty of research examining solution approaches to the PRP has been conducted, few studies consider the production (or replenishment) of *multiple* items. The extension to include an additional item or two is not trivial, especially if the production of these items is coordinated. Lot-sizing decisions would now be impacted by the fixed cost of setting up manufacturing for a family of items, and not just by the per-unit production cost. This makes the lot-sizing portion of the PRP even more challenging to solve. Given that PRP is already *NP*-hard due to the embedded traveling salesman problem within the vehicle routing constraints, this added complexity poses challenges in developing appropriate solution methods that can both find near-optimal solutions and do so in a reasonable amount of time.

Production-routing problems combine a number of decisions that have individually been studied extensively, but whose integration is now being treated more thoroughly. There is a vast body of literature on lot-sizing problems (as discussed above) and vehicle routing problems (Braekers et al., 2016; Koç et al., 2016), so our focus in this subsection will be on the trends in PRP research. We discuss the review by Adulyasak et al. (2015) in more detail below and extend its summary in Table 1.1 by further classifying characteristics of the variants of PRP studied in these articles, and adding some recently published research

Chandra and Fisher (1994), Fumero and Vercellis (1999), Armentano et al. (2011), Brahimi and Aouam (2016), and Kang et al. (2017) address the PRP with multiple items, with only Kang et al. (2017) considering the case of coordinated (or joint) replenishment. All of these authors use heuristic solution approaches: Chandra and Fisher (1994) and Brahimi and Aouam

Table 1.1: Classification of PRP literature

	Production					Vehicle Routing				Solution	
Authors	# plts	# items	Cap.	Coord.	BO	Fleet	# veh	SEC/NF	Split	Type	Approach
Chandra and Fisher (1994)	1	mult.	✓			hom.	unlim.	NF	✓	H	Decomposition
Fumero and Vercellis (1999)	1	mult.	✓			hom.	lim.	NF	✓	H	Lag. Relaxation
Lei et al. (2006)	mult.	1	✓			het.	lim.	NF	✓	H	Decomposition
Boudia et al. (2007)	1	1	✓			hom.	lim.	SEC		H	GRASP
Boudia et al. (2008)	1	1	✓			hom.	lim.	SEC		H	Decomposition
Boudia and Prins (2009)	1	1	✓			hom.	lim.	SEC		H	Memetic
Bard and Nananukul (2009b)	1	1	✓			hom.	lim.	NF	✓	H	Tabu search
Bard and Nananukul (2009a)	1	1	✓			hom.	lim.	NF	✓	E	B&P
Bard and Nananukul (2010)	1	1	✓			hom.	lim.	NF	✓	E	B&P
Ruokokoski et al. (2010)	1	1				hom.	1[a]	SEC		E	B&C
Armentano et al. (2011)	1	mult.	✓			hom.	unlim.	NF		H	Tabu search
Archetti et al. (2011)	1	1	✓			hom.	unlim.	SEC		E/H	B&C
Adulyasak et al. (2014b)	1	1	✓			hom.	lim.	SEC		H	ALNS
Adulyasak et al. (2014a)	1	1	✓			hom.	lim.	SEC		E/H	B&C/ALNS
Absi et al. (2015)	1	1	✓			hom.	lim.	SEC		H	Iterative MIP
Brahimi and Aouam (2016)	1	mult.	✓		✓	hom.	unlim.	SEC		H	Decomposition
Solyalı and Süral (2017)	1	1	✓			hom.	lim.	SEC		H	Decomposition
Kang et al. (2017)	1	mult.	✓	✓	[b]	het.	lim.	SEC	[c]	H	PSO

BO: backorders; SEC: subtour elimination constraints; NF: network flow constraints; Split: split deliveries; mult.: multiple; het: heterogeneous; hom: homogeneous; lim.: limited; unlim.: unlimited; E: exact, H: heuristic;
[a] uncapacitated vehicle
[b] overtime and outsourcing allowed
[c] considers pickups from suppliers, not deliveries to customers

(2016) decompose the PRP into production and routing subproblems, while Fumero and Vercellis (1999) employ a Lagrangian relaxation approach. Armentano et al. (2011) and Kang et al. (2017) use tabu search and particle swarm optimization, respectively, which are effective for solving large instances of the PRP. Of these multi-item variants, network flow constraints are used by Chandra and Fisher (1994), Fumero and Vercellis (1999), and Armentano et al. (2011), while subtour elimination constraints (SECs) are modelled by Brahimi and Aouam (2016) and Kang et al. (2017). Though network flow constraints seem to simplify the decomposition of the VRP portion of PRP, Ruokokoski et al. (2010) state that the LP-relaxation of the network flow VRP formulation will provide weaker bounds than when SECs are employed.

Much of the research focuses on solving single-item production problems with a homogeneous fleet of vehicles through heuristics, either decomposition, metaheuristic, or iterative procedure (Lei et al., 2006; Boudia et al., 2007, 2008; Boudia and Prins, 2009; Bard and Nananukul, 2009b; Adulyasak et al., 2014b; Absi et al., 2015; Solyalı and Süral, 2017). Bard and Nananukul (2009a, 2010), Ruokokoski et al. (2010), Archetti et al. (2011), and Adulyasak et al. (2014b) pursue exact approaches based on branch-and-cut algorithms. Lei et al. (2006) are the only authors in this group that model multiple plants producing a single product, and use a heterogeneous fleet of vehicles while considering deterministic travel times.

Ruokokoski et al. (2010) and Brahimi and Aouam (2016) consider strong reformulations for the lot-sizing problem within the PRP. While reformulations have proven successful at tackling the complexities of CCLSPs, they do not seem to provide much benefit in the context of the production-routing problem, as reported by Adulyasak et al. (2014a). Those researchers suggest that valid lot-sizing inequalities proposed by Archetti et al. (2007, 2011) are sufficient to produce good solutions, but this is under the assumption that all demands must be met on time. It is not clear whether the same result would occur when backorders are allowed.

Two of the more recent studies on PRP with multiple items are by Brahimi and Aouam (2016) and Kang et al. (2017). While Brahimi and Aouam (2016) allow backorders, Kang et al. (2017) account for limited production capacity through use of overtime and outsourcing of production. It is interesting to note that the manufacturing facility in Kang et al. (2017) is responsible for picking up materials from multiple suppliers (rather than delivering finished goods to customers), and that they also consider an all-units discount policy.

We also observe emerging research that builds upon the PRP, but is not reflected in Table 1.1. Chitsaz et al. (2017) combine mathematical programming and heuristics in their matheuristic for an assembly routing problem. Here, not only is production modeled, but also specific assembly constraints that dictate the material required in a finished product. More tactical aspects of the PRP are also studied by Nananukul (2013) and Konur and Geunes (2016), who both focus on optimizing the clustering of customers that are serviced by the same fleet of vehicles.

Based on our survey of the literature, there are a few PRP variants that have received little attention. These include the coordinated replenishment of multiple items, allowance for backorders, a heterogeneous fleet of vehicles, and split deliveries. Furthermore, Adulyasak et al. (2015) indicate that there is little research on using exact approaches to solve the PRP with multiple items.

1.3 Organization of Dissertation

There are opportunities to extend both the CCLSP and PRP by considering the presence of multiple product families, and by enhancing existing solution techniques. We begin with a formal presentation of the facility-location reformulation of MF-CCLSP in Chapter 2 and discuss methods to improve both upper and lower bounds through a combination of Benders decomposition, evolutionary algorithm and subgradient optimization. We measure the performance of our approach against the works by Süral et al. (2009) and de Araujo et al. (2015), as well as an exact cutting plane approach. Chapter 3 describes the multi-item PRP subject to capacitated and coordinated replenishments, and studies impacts on inventory, backorder, and routing costs under different business settings. Chapter 4 summarizes the major contributions of this work, and describes avenues of future work to further enrich these research areas.

Chapter 2

Coordinated Capacitated Lot-sizing for Multiple Product Families

2.1 Problem Description

Production planning, or lot-sizing problems generally determine when and how much to produce so that costs of setting up production plus holding inventory are minimized. The multiple-family capacitated coordinated lot-sizing problem with setup times ($MF-CCLSP$) seeks the minimum cost lot-sizing schedule subject to capacity constraints over a planning horizon. There are many ways to express these decisions mathematically, and the structure chosen will impact solution time and quality. In this chapter, we provide an overview of typical mathematical formulations for the coordinated capacitated lot-sizing problem and its variants before detailing solution approaches for the case of multiple product families.

2.1.1 Production-Inventory Formulation

The formulation that most closely reflects the classical definition of lot-sizing problems uses variables to represent production and inventory quantities, respectively. Let $i \in \{1,2,\ldots,m\}$ be the product number index in family $j \in \{1,2,\ldots,l\}$ over the planning horizon, with n time periods indexed by t. Demand, d_{ijt}, is time-varying and deterministic (i.e., dynamic), and the cost of holding one unit from period t to $t+1$ is H_{ijt}. In each period t, there is CAP_t production time available. It is consumed by unit production time a_{ijt}, item setup time b_{ijt}, and family setup time f_{jt}. Setting up a family of items incurs cost R_{jt} and an item setup cost triggers cost Q_{ijt}. We determine how many units, x_{ijt}, to produce, and how many units, s_{ijt}, to hold that will minimize setup plus holding costs over the horizon. Binary decision variables y_{ijt} (and z_{jt}) define whether or not an item (or family) setup occurs in period t.

To avoid infeasibilities in the model, initial inventory s_{ij0}, may be used at a very high cost, K_{ij}. A list of notation can be found in Table 2.1.

$$[P] \quad v_P = \min \quad \sum_{j=1}^{l} \sum_{t=1}^{n} \left[\sum_{i=1}^{m} \left(H_{ijt} s_{ijt} + Q_{ijt} y_{ijt} \right) + R_{jt} z_{jt} \right] + \sum_{i=1}^{m} \sum_{j=1}^{l} K_{ij} s_{ij0} \tag{2.1}$$

$$\text{s.t.} \quad s_{ij,t-1} + x_{ijt} - s_{ijt} = d_{ijt} \qquad \forall i,j,t \tag{2.2}$$

$$x_{ijt} \leq M y_{ijt} \qquad \forall i,j,t \tag{2.3}$$

$$y_{ijt} \leq z_{jt} \qquad \forall i,j,t \tag{2.4}$$

$$\sum_{j=1}^{l} \left[\sum_{i=1}^{m} \left(a_{ijt} x_{ijt} + b_{ijt} y_{ijt} \right) + f_{jt} z_{jt} \right] \leq CAP_t \qquad \forall t \tag{2.5}$$

$$x_{ijt}, s_{ijt} \geq 0 \qquad \forall i,j,t \tag{2.6}$$

$$s_{ij0} \geq 0 \qquad \forall i,j \tag{2.7}$$

$$y_{ijt} \in \{0,1\} \qquad \forall i,j,t \tag{2.8}$$

$$z_{jt} \in \{0,1\} \qquad \forall j,t \tag{2.9}$$

The objective function (2.1) minimizes total setup, holding and initial inventory costs. Constraints (2.2) maintain inventory balance in each period, while inequalities (2.3) and (2.4) ensure that item and family production take place only when a setup is incurred, where $M = min(\sum_{t'}^{n} d_{ijt'}, \frac{CAP_t - b_{ijt} - f_{jt}}{a_{ijt}})$. Available setup plus production time is limited by constraints (2.5), with (2.6) to (2.9) defining non-negativity and integrality restrictions.

Table 2.1: Summary of notation used in Chapter 2

Notation	Description	Indices or Unit of measure
Indices		
i	Index for items	$i = 1,2,\ldots,m$
j	Index for families	$j = 1,2,\ldots,l$
t	Index for production time periods	$t = 1,2,\ldots,n$
u	Index for demand time periods	$u = 1,2,\ldots,n$
Parameters		
d_{ijt}	Demand for item i from family j in period t	units
H_{ijt}	Cost for carrying item (i,j) from period t to $t+1$	\$/unit/time
H^p_{ijt}	Initial inventory and holding cost for item (i,j) to satisfy demand in period t	\$/unit
$\bar{H}_{ijut}$	Holding cost for item (i,j) produced in period u and consumed in period t	\$/unit
CAP_t	Available production time in period t	production time
a_{ijt}	Unit production time for item (i,j) in period t	production time/unit
b_{ijt}	Setup time for item (i,j) in period t	production time
f_{jt}	Setup time for family j in period t	production time
R_{jt}	Family j setup cost in period t	\$
Q_{ijt}	Item (i,j) setup cost in period t	\$
K_{ij}	Cost for using initial inventory of item (i,j)	\$/unit
Decision Variables		
x_{ijt}	Production amount of item (i,j) in period t	Continuous
s_{ijt}	Amount of item (i,j) held from period t to $t+1$	Continuous
w_{ijut}	Fraction of period t demand for item (i,j) satisfied by production from period u	Continuous
p_{ijt}	Fraction of period t demand for item (i,j) satisfied by initial inventory	Continuous
y_{ijt}	=1 if setup for item (i,j) is placed in period t; 0 otherwise	Binary
z_{jt}	=1 if setup for family j is placed in period t; 0 otherwise	Binary

2.1.2 Strong Formulations

A stronger formulation, one whose LP-relaxation will provide a better lower bound on the optimal solution, can be obtained by redefining decision variables of $[P]$ and reformulating the problem as either a shortest-path-type (SP) or facility-location-type (FL) problem. Each reformulation results in an increase in the number of production decision variables, though their LP-relaxation solutions are identical to one another, even in the capacitated and joint replenishment cases (Denizel et al., 2008; Solyalı and Süral, 2012). Furthermore, the projection of the solution space of these reformulations is equivalent to the projection of the convex hull of the weak formulation (Pochet and Wolsey, 2006). In the single-item uncapacitated case, the SP and FL formulations both exhibit the unimodularity property, and hence their LP relaxations provide integer solutions that are optimal. For the capacitated case, these reformulations will provide an improved LP relaxation value, but the LP solution will not necessarily be integer.

We reformulate MF-CCLSP according to a facility-location-type problem, letting w_{ijut} represent the fraction of demand of period t for item i in family j that is satisfied by production in period u and p_{ijt} represents the fraction of demand of period t for item i in family j that is satisfied from initial inventory. The initial inventory and holding cost for unit (i,j) to satisfy demand in period t is denoted by $H^p_{ijt} = (K_{ij} + \sum_{u=1}^{t-1} H_{iju})d_{ijt}$. The holding cost for unit (i,j) is $\bar{H}_{ijut} = \sum_{u'=u}^{t-1} H_{iju'}d_{ijt}$.

The facility location formulation is:

$$[FL] \quad v_{FL} = \min \quad \sum_{i=1}^{m}\sum_{j=1}^{l}\sum_{t=1}^{n} H^p_{ijt}p_{ijt} + \sum_{j=1}^{l}\sum_{u=1}^{n}\left[\sum_{i=1}^{m}\left(Q_{iju}y_{iju} + \sum_{t=u}^{n}\bar{H}_{ijut}w_{ijut}\right) + R_{ju}z_{ju}\right] \tag{2.10}$$

$$\text{s.t.} \quad p_{ijt} + \sum_{u=1}^{t} w_{ijut} = 1 \qquad \forall i,j,t \tag{2.11}$$

$$w_{ijut} \le y_{iju} \qquad \forall i,j, \text{ and } 1 \le u \le t \le n \tag{2.12}$$

$$y_{iju} \le z_{ju} \qquad \forall i,j,u \tag{2.13}$$

$$\sum_{j=1}^{l}\left[\sum_{i=1}^{m}\left(a_{iju}\sum_{t=u}^{n} d_{ijt}w_{ijut} + b_{iju}y_{iju}\right) + f_{ju}z_{ju}\right] \le CAP_u \qquad \forall u \tag{2.14}$$

$$w_{ijut} \ge 0 \qquad \forall i,j,u,t \tag{2.15}$$

$$p_{ijt} \ge 0 \qquad \forall i,j,t \tag{2.16}$$

$$y_{iju} \in \{0,1\} \qquad \forall i,j,u \tag{2.17}$$

$$z_{ju} \in \{0,1\} \qquad \forall j,u \tag{2.18}$$

Here, the objective function (2.10) minimizes the total aggregated holding and initial costs, along with setup costs. Demand constraints (2.2) have been translated to equations (2.11), and state that demand in period t must be satisfied by the sum of initial inventory and production from periods $u \leq t$ destined for period t. Setup constraints (2.12) and (2.13) ensure that item and family production do not occur unless an item or family setup takes place, and constraints (2.14) to (2.18) correspond directly to constraints (2.5) - (2.9).

In Section 2.3, we will also examine a special case where setup costs are not considered, so only holding costs contribute to the objective function value. In practical settings, the setup cost is not typically separated from "regular" costs, such as labour wages or material cost, and it can be cumbersome to quantify. If a setup does not actually occur, the labour wage is still paid, as the worker may be assigned to a different task. Mathematically, the absence of these setup costs produces problems that are more difficult to solve, as no particular family nor item will sway the cost objective (assuming that setup costs are significantly more than holding costs, and that holding costs are somewhat homogenous). See Süral et al. (2009) for further information about the omission of setup costs and impact of setup times.

2.2 Solution Approaches

Any feasible solution to $[FL]$ will provide an upper bound, but finding that solution is no simple feat (Maes et al., 1991). Trigeiro et al. (1989) employ a smoothing heuristic that shuffles lot-sizes between time periods, using both forward- and backward-looking techniques, though they do state that the method does not always succeed. The primal heuristic employed by Süral et al. (2009) solves a period-by-period knapsack problem from the end of the time horizon to the beginning to establish a production schedule, though it too may result in lot-sizes that require excessive initial inventories to be used.

To overcome these difficulties, a better upper bounding procedure is needed. Similar to Süral et al. (2009), we first determine the time periods in which setups will take place. Fixing those binary variables in $[FL]$, we are left with only continuous variables and the resulting problem is a minimum cost network flow problem (MCNFP) that is easy to solve. To find values for the binary variables, we apply Benders decomposition.

2.2.1 Benders Decomposition

When binary variables from $[FL]$ are fixed to $\hat{y}_{iju}$ and $\hat{z}_{ju}$, we obtain the Benders subproblem:

$$[BSP] \quad v_{SP} = \min \quad \omega + \sum_{i=1}^{m}\sum_{j=1}^{l}\sum_{t=1}^{n} H_{ijt}^{p} p_{ijt} + \sum_{i=1}^{m}\sum_{j=1}^{l}\sum_{t=1}^{n}\sum_{u=1}^{t} \bar{H}_{ijut} w_{ijut} \tag{2.19}$$

$$\text{s.t.} \quad p_{ijt} + \sum_{u=1}^{t} w_{ijut} = 1 \qquad \forall i,j,t \tag{2.20}$$

$$w_{ijut} \le \hat{y}_{iju} \qquad \forall i,j, \text{ and } 1 \le u \le t \le n \tag{2.21}$$

$$\sum_{j=1}^{l}\sum_{i=1}^{m} a_{iju} \sum_{t=u}^{n} d_{ijt} w_{ijut} \le CAP_u - \sum_{j=1}^{l}\left[\sum_{i=1}^{m} b_{iju}\hat{y}_{iju} + f_{ju}\hat{z}_{ju}\right] \qquad \forall u \tag{2.22}$$

$$w_{ijut} \ge 0 \qquad \forall i,j,u,t \tag{2.23}$$

$$p_{ijt} \ge 0 \qquad \forall i,j,t \tag{2.24}$$

where $\omega = \sum_{j,u}\left[\sum_i Q_{iju}\hat{y}_{iju} + R_{ju}\hat{z}_{ju}\right]$. The solution to $[BSP]$ gives a production plan based on the setup schedule determined by $[BMP]$. The presence of variables p_{ijt} ensures that this plan will be feasible and hence $[BSP]$ gives an upper bound (UB) on $[FL]$.

Introducing dual variables $\lambda_{ijt}^{(d)}$, $\lambda_{ijut}^{(s)}$ and $\lambda_{u}^{(c)}$ corresponding to constraints (2.20), (2.21), and (2.22) respectively, and dropping the constant term ω, we take the dual of $[BSP]$:

$$[DSP] \quad v_{DSP} = \max \quad \sum_{i=1}^{m}\sum_{j=1}^{l}\sum_{t=1}^{n} \lambda_{ijt}^{(d)} + \sum_{i=1}^{m}\sum_{j=1}^{l}\sum_{t=1}^{n}\sum_{u=1}^{t} \hat{y}_{iju}\lambda_{ijut}^{(s)} + \sum_{u}\left(CAP_u - \sum_{j=1}^{l}\left[\sum_{i=1}^{m} b_{iju}\hat{y}_{iju} + f_{ju}\hat{z}_{ju}\right]\right)\lambda_{u}^{(c)}$$

$$\text{s.t.} \quad \lambda_{ijt}^{(d)} \le H_{ijt}^{p} \qquad \forall i,j,t$$

$$\lambda_{ijt}^{(d)} + \lambda_{ijut}^{(s)} + a_{iju}d_{ijt}\lambda_{u}^{(c)} \le \bar{H}_{ijut} \qquad \forall i,j, \text{ and } 1 \le u \le t \le n$$

$$\lambda_{ijt}^{(d)} \quad \text{urs} \qquad \forall i,j,t$$

$$\lambda_{ijut}^{(s)} \leq 0 \qquad \forall i, j, \text{ and } 1 \leq u \leq t \leq n$$
$$\lambda_{u}^{(c)} \leq 0 \qquad \forall u$$

Introducing variable

$$\theta_B = \max_{r \in R^P} \left\{ \sum_{i=1}^{m} \sum_{j=1}^{l} \sum_{t=1}^{n} \lambda_{ijt}^{(d)r} + \sum_{i=1}^{m} \sum_{j=1}^{l} \sum_{t=1}^{n} \sum_{u=1}^{t} \hat{y}_{iju} \lambda_{ijut}^{(s)r} + \sum_{u} \left(CAP_u - \sum_{j=1}^{l} \left[\sum_{i=1}^{m} b_{iju} \hat{y}_{iju} + f_{ju} \hat{z}_{ju} \right] \right) \lambda_{u}^{(c)r} \right\}$$

where R^P is the set of all extreme points of the feasible solutions to $[DSP]$, the dual subproblem can be reformulated as

$$\begin{aligned} \min \quad & \theta_B \\ \text{s.t.} \quad & \sum_{i=1}^{m} \sum_{j=1}^{l} \sum_{t=1}^{n} \lambda_{ijt}^{(d)r} + \sum_{i=1}^{m} \sum_{j=1}^{l} \sum_{t=1}^{n} \sum_{u=1}^{t} \hat{y}_{iju} \lambda_{ijut}^{(s)r} \\ & \qquad + \sum_{u} \left(CAP_u - \sum_{j=1}^{l} \left[\sum_{i=1}^{m} b_{iju} \hat{y}_{iju} + f_{ju} \hat{z}_{ju} \right] \right) \lambda_{u}^{(c)r} \leq \theta_B \quad \forall r \in R^P \end{aligned}$$

Initial inventory p_{ijt} ensures that the subproblem will always be feasible, so there is no need to generate feasibility cuts. Thus, the Benders master problem is formulated as follows:

$$[BMP] \qquad \min \quad \theta_B + \sum_{j=1}^{l} \sum_{u=1}^{n} \left[\sum_{i=1}^{m} Q_{iju} y_{iju} + R_{ju} z_{ju} \right] \tag{2.25}$$

$$\begin{aligned} \text{s.t.} \quad & \sum_{i=1}^{m} \sum_{j=1}^{l} \sum_{t=1}^{n} \lambda_{ijt}^{(d)r} + \sum_{i=1}^{m} \sum_{j=1}^{l} \sum_{t=1}^{n} \sum_{u=1}^{t} y_{iju} \lambda_{ijut}^{(s)r} \\ & + \sum_{u} \left(CAP_u - \sum_{j=1}^{l} \left[\sum_{i=1}^{m} b_{iju} y_{iju} + f_{ju} z_{ju} \right] \right) \lambda_{u}^{(c)r} \leq \theta_B \quad \forall r \in R^P \end{aligned} \tag{2.26}$$

$$y_{iju} \leq z_{ju} \quad \forall i, j, u \tag{2.27}$$

$$y_{iju} \in \{0, 1\} \quad \forall i, j, u \tag{2.28}$$

$$z_{ju} \in \{0, 1\} \quad \forall j, u \tag{2.29}$$

$$\theta_B \quad \text{urs} \tag{2.30}$$

To initialize the procedure, $(\lambda_{ijt}^{(d)r}, \lambda_{ijut}^{(s)r}, \lambda_{u}^{(c)r})$ are set to 0 for all i, j, u, and t, and the following restricted master problem is solved:

$$
\begin{aligned}
[rBMP] \qquad \min \quad & \sum_{j=1}^{l} \sum_{u=1}^{n} \left[\sum_{i=1}^{m} Q_{iju} y_{iju} + R_{ju} z_{ju} \right] \\
\text{s.t.}\, & y_{iju} \leq z_{ju} \quad \forall i, j, u \\
& y_{iju} \in \{0,1\} \quad \forall i, j, u \\
& z_{ju} \in \{0,1\} \quad \forall j, u
\end{aligned}
$$

As $[rBMP]$ is a relaxation of $[FL]$, the solution to $[rBMP]$ provides a lower bound (LB) to $[FL]$ and is carried to the Benders subproblem, where a Benders optimality cut (2.26) is obtained from the dual solution to $[BSP]$. We iterate between $[BMP]$ and $[BSP]$ until the UB and LB converge, or no improvement in the bounds is found within 50 consecutive Benders cuts.

Strengthening the Benders Master Problem

We generate two types of cuts to strengthen $[BMP]$. The first, Type 1 cuts, ensure that if demand in period $t = 1$ is positive, then a setup must be incurred:

$$
\text{If } d_{ij1} > 0, \text{ then } y_{ij1} \geq 1 \quad \forall i, j \tag{2.31}
$$

Based on given demand and capacity parameters, we can also propose that a minimum number of setups occur over various intervals to ensure feasibility (Maes et al., 1991; Suerie and Stadtler, 2003). These Type 2 cuts are generated by first defining the set $Q_{t,U'}$ of ordered demands between periods t and $t+U'-1$, where demand is expressed in capacity units and $U' = 1,2,3$. For example, $Q_{t,U'} = \left\{ a_{[1]}d_{[1]} \leq a_{[2]}d_{[2]} \leq \ldots \leq a_{[t+U'-1]}d_{[t+U'-1]} \right\}$, where $[x]$ represents an index (i, j, s) with $i = 1, \ldots, m$, $j = 1, \ldots, l$, $s = t, t+1, \ldots, t+U'-1$, and $t = 2, 3, \ldots, n-U'+1$. Let $\bar{Q}_{t,U'}$ be the maximal subset of $Q_{t,U'}$ such that the sum of the demands of the subset can be entirely satisfied from the remaining cumulative slack capacity of periods $s = 2$ to $t-1$, after all previous demands are fulfilled. As in Maes et al. (1991), we also let $\bar{Q}_{t,U'}$ denote the sum of demands in this subset, giving us:

$$\bar{Q}_{t,U'} \subseteq \left\{ Q_{t,U'} \middle| \bar{Q}_{t,U'} + a_{ijs}d_{ijs} > \sum_{t'=1}^{t-1} \left(CAP_{t'} - \sum_{j=1}^{l} f_{jt'} \right) - \sum_{i=1}^{m}\sum_{j=1}^{l}\sum_{t'=1}^{t-1} a_{ijt'}d_{ijt'} - \sum_{\substack{(ij): \\ \sum_{t'=1}^{t-1} d_{ijt'}>0}} b_{ijt'} \right\}$$

$$\forall i,j,s \in Q_{t,U'} \backslash \bar{Q}_{t,U'}$$

Once $\bar{Q}_{t,U'}$ is found for every $U' = 1,2,3$, we add the cut

$$\sum_{i=1}^{m}\sum_{j=1}^{l}\sum_{s=t}^{t+U'-1} y_{ijs} \geq m\,l\,U' - |\bar{Q}_{t,U'}| \tag{2.32}$$

Based on tests conducted on a subset of problem instances, adding cuts (2.31) and (2.32) produced better feasible solutions compared to when they were not present in $[BMP]$, and including them decreased the computational time needed to find feasible solutions.

2.2.2 Evolutionary Algorithm

To speed up convergence of the Benders routine, we use an evolutionary algorithm to quickly search a neighbourhood of solutions with the goal of improving the upper bound. Below, we discuss the settings of our evolutionary algorithm:

i) *Chromosome representation:* We use direct representation, encoding individuals through item setup variables, y_{ijt}. This provides simple manipulation of the individuals when mutation and crossovers occur. Each individual is a vector of length m^*l^*n with binary values, from which we can directly infer family setup values, z_{jt}. These y_{ijt} and z_{jt} values are fixed in $[BSP]$ to solve for production quantities.

ii) *Fitness:* The fitness of each individual is equal to the objective function value of its associated MCNFP, where item and family setup variables are fixed. The fittest individual in the population will produce the lowest objective function value in problem $[BSP]$.

iii) *Population size:* As per the findings in Süral et al. (2010), we restrict our population size to 100 individuals to keep computational times reasonable, but also to allow for better quality solutions to emerge.

iv) *Initialization:* The initial population comprises a proportion of randomly- and heuristically-generated individuals. By using some individuals derived from $[BMP]$, we are adding intelligence to the heuristic and stacking the EA with some good solutions. While Goren et al. (2010) note that most literature uses a completely randomly-generated population, Süral et al. (2010) indicate promising results in their tests carried out on the traveling salesman problem that combine both heuristically- and randomly-generated individuals. The proportion of randomly generated individuals is limited to 45%, based on experiments conducted in Süral et al. (2010), but this does not imply that 45% is the optimal proportion to apply in every problem setting. A discrete uniform distribution is used to create each binary chromosome of the randomly-generated individuals.

   To initialize the entire process, we create 55 vectors to represent 55% of the population determined heuristically through Benders decomposition. Each chromosome is set to zero to indicate that no setups take place over the entire horizon, resulting in a very costly plan to use initial inventory to satisfy all demand. In the first pass of our heuristic, all 55 individuals are carried to Routine 2 (see Section 2.2.4) where 45 newly- and randomly-generated individuals are added to the population. This will bring the total population size to 100.

   In subsequent passes, only the 55 most-fit individuals from $[BMP]$ are carried forward to the evolutionary algorithm, where a new set of 45 randomly-generated individuals are added to the population. Further details on how the population is updated throughout the heuristic are provided in Section 2.2.4.

v) *Crossover and mutation:* The two most-fit individuals, Parent 1 and Parent 2, are chosen for two-bit crossover at the 10th chromosome from the beginning and the 10th chromosome from the end of the parent vectors. For each parent, chromosomes 1 through 10 and $m^{*}l^{*}n-10$ through $m^{*}l^{*}n$ will remain the same, but those chromosomes in between will be switched from Parent 1 to Parent 2, and vice versa, creating two offspring. See Figure 2.1 for a schematic example of two-bit crossover, where $m=2, l=2$, and $n=6$ for a vector with 24 chromosomes.

After the crossover takes place, there is a small probability, p_m, that each chromosome of the offspring vectors mutate. We use $p_m = 1\%$ and employ single-bit flip mutation (i.e., if mutation takes place on a chromosome currently set to 1, we will change its value to 0).

While we use a fixed crossover point, an adaptive crossover point could be implemented to suit the particular needs of the model being solved. We are concerned with larger instances of CCLSP with multiple families, so we expect our vector of setup variables to exceed 20 bits; anything fewer than this would represent a small problem whose solution would be more efficiently found by solving the problem directly with a commercial solver.

vi) *Replacement:* The two offspring replace the two least-fit individuals of the population.

vii) *Termination:* The algorithm terminates after 100 generations, or when there is no improvement in the fitness of the best individual in 25 consecutive generations.

2.2.3 Subgradient Optimization

As $[BMP]$ can be difficult to solve, the value of the lower bound obtained at the end of the Benders-EA heuristic may still be far from the optimal solution. In an attempt to obtain a better lower bound, Lagrangian relaxation is applied, with Lagrange multipliers updated by subgradient

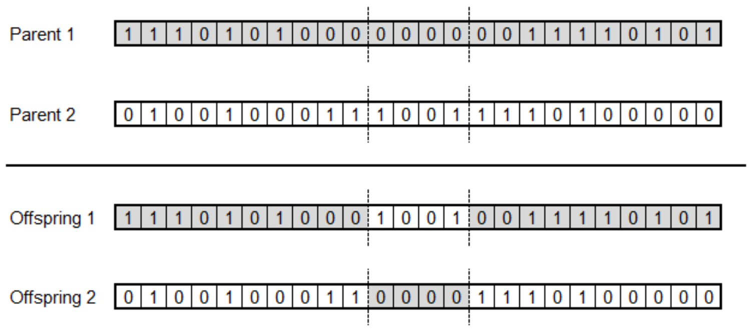

Figure 2.1: Example of two-bit crossover

optimization. Relaxing constraint (2.11) of $[FL]$ results in the following Lagrangian subproblem:

$$\begin{aligned}
[FL_\lambda] \quad v_{FL}(\mathrm{LR}(\lambda)) = \min \quad & \sum_{i=1}^{m}\sum_{j=1}^{l}\sum_{t=1}^{n}\left[\left(H^p_{ijt} - \lambda_{ijt}\right)p_{ijt} + \lambda_{ijt}\right] \\
& + \sum_{j=1}^{l}\sum_{u=1}^{n}\left[\sum_{i=1}^{m}\left(Q_{iju}y_{iju} + \sum_{t=u}^{n}\left(\bar{H}_{ijut} - \lambda_{ijt}\right)w_{ijut}\right) + R_{ju}z_{ju}\right] \\
\text{s.t.} \quad & (2.12) - (2.18)
\end{aligned}$$

The constant term, $\sum_{ijt}\lambda_{ijt}$, is omitted from the objective function during computations, but is used to compute the Lagrangian bound.

Initial inventory variables, p_{ijt}, do not impact any of the constraints in $[FL_\lambda]$. Their values can be determined, as in Jans and Degraeve (2004), by observing the coefficient value for a given λ_{ijt}. Since our objective is minimization, if $H^p_{ijt} - \lambda_{ijt} < 0$, then $p_{ijt} = 1$; otherwise, $p_{ijt} = 0$.

Note that Lagrange multipliers λ_{ijt} are defined by *demand* period t, and that we can separate the Lagrangian subproblem by *production* period u:

$$[FL_\lambda]_u \quad v_{FL}(\mathrm{LR}(\lambda))_u = \min \quad \sum_{j=1}^{l}\left[\sum_{i=1}^{m}\left(Q_{iju}y_{iju} + \sum_{t=u}^{n}\left(\bar{H}_{ijut} - \lambda_{ijt}\right)w_{ijut}\right) + R_{ju}z_{ju}\right] \tag{2.33}$$

$$\begin{aligned}
\text{s.t.} \quad & w_{ijut} \le y_{iju} && \forall i,j,t \ge u && (2.34)\\
& y_{iju} \le z_{ju} && \forall i,j && (2.35)\\
& \sum_{j=1}^{l}\left[\sum_{i=1}^{m}\left(a_{iju}\sum_{t=u}^{n} d_{ijt}w_{ijut} + b_{iju}y_{iju}\right) + f_{ju}z_{ju}\right] \le CAP_u && && (2.36)\\
& w_{ijut} \ge 0 && \forall i,j && (2.37)\\
& y_{iju} \in \{0,1\} && \forall i,j && (2.38)\\
& z_{ju} \in \{0,1\} && \forall j && (2.39)
\end{aligned}$$

Initial multipliers, λ^0_{ijt}, are obtained from the dual of constraint (2.11) of the LP-relaxation of $[FL]$. Using the best upper bound found from the Benders-EA heuristic, UB, the Lagrange multipliers are updated using the equation:

$$\lambda_{ijt}^{k+1} = \lambda_{ijt}^{k} + \alpha^k \left\{ \left[UB - v_{FL}(\mathrm{LR}(\lambda^k)) \right] \Big/ \left[\sum_{ijt} (1 - p_{ijt} - \sum_{u=1}^{t} w_{ijut}) \right]^{\frac{1}{2}} \right\} \times \left[\sum_{ijt} (1 - p_{ijt} - \sum_{u=1}^{t} w_{ijut}) \right]$$

where α^k is a scalar, equal to 0.02 when $k = 0$. If no improvement has been made in five consecutive iterations, α is reduced by 10%, and the entire subgradient routine terminates after 200 iterations.

2.2.4 Overall Heuristic

We refer the reader to Routines 1, 2, and 3, where we present the pseudocode for the overall heuristic and its subroutines, and to Figure 2.2 for a visual representation of the heuristic logic. Benders decomposition, followed by the evolutionary algorithm is repeated five times, or until the solution time exceeds one hour. The best upper bound found so far is carried through to the subgradient routine, where the best lower bound found is updated.

Based on our preliminary computations, good solutions were obtained in a reasonable amount of time when iterating between Benders and EA five times: with fewer iterations, the duality gap remained large, but with more iterations, total CPU time increased greatly with little improvement in the duality gap. The overall time limit of one hour was chosen to give adequate time for our heuristic to find good solutions.

In Routine 1, the following steps occur in one iteration of the outer loop (the outer loop is performed up to five times):

- $[BSP]$ and $[BMP]$ are solved (until convergence, or if there is no improvement in either objective function in 50 consecutive iterations). Setup schedules produced by $[BMP]$ are passed to Routine 2 and used to initialize the population.
- Routine 2 seeks an improved upper bound to the overall problem by quickly searching a

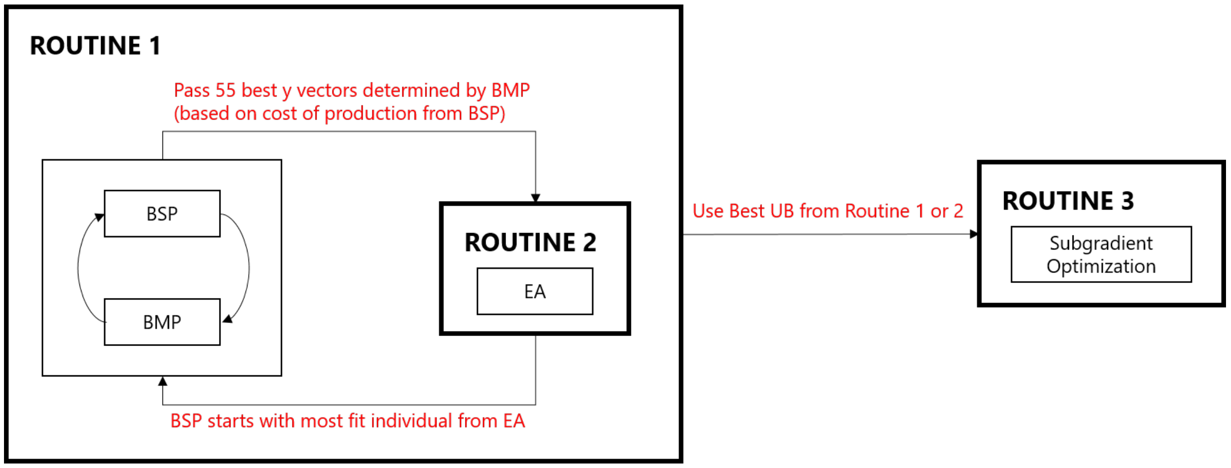
ROUTINE 1
Pass 55 best y vectors determined by BMP (based on cost of production from BSP)
BSP
BMP
ROUTINE 2
EA
BSP starts with most fit individual from EA
Use Best UB from Routine 1 or 2
ROUTINE 3
Subgradient Optimization

Figure 2.2: Benders-EA Heuristic Logic

large neighbourhood of potential solutions. This updated population of solutions is passed back to $[BSP]$ and $[BMP]$, where the most fit individual is used to initialize $[BSP]$.

We exit this outer loop if $[BSP]$ and $[BMP]$ converge, there is no improvement after 50 iterations, or an overall CPU time of 1 hour has been exceeded. If five outer loops are completed, we proceed to Routine 3, which searches for a better lower bound than the one obtained from $[BMP]$.

2.3 Computational Results

We examine two problem settings: problem $[P]$ as it is (*with* setup costs), and $[P]$ *without* setup costs. In each setting, we compare the performance of our approach against the SDW heuristic proposed by Süral et al. (2009) as well as an exact approach, described further in Section 2.3.3. For $[P]$ without setup costs, in addition to the presence of multiple families, we also consider the single-family case ($j = 1$) and include results of the RHB heuristic developed by de Araujo et al. (2015) in our evaluation.

The data sets of Trigeiro et al. (1989) have been modified to include multiple product families. A description of other approaches, against which we will measure our performance, is provided next. All experiments were run on a PC with a dual-core 2.61GHz processor and 39GB RAM.

Routine 1: Benders_EA

```
begin
    /* create preliminary EA population, and initialize problem */
    create 55 individuals with all chromosomes = 0, each with very high (bad) fitness value;
    set all $\lambda^s = 0, \lambda^d = 0, \lambda^c = 0, SP_y = 0, SP_z = 0$;
    set B_UB=10000000, B_LB=-10000000;
    set UB_noimp = 0; LB_noimp = 0; time_flag = 0;
    /* iterate between Benders and Evolutionary Algorithm */
    for $outer\_loop = 1 : 5$ do
        while $|(B\_UB - B\_LB)/B\_LB| > 0.001$ do
            if $time\_flag == 1$ then
                exit while loop
            use $SP_y$ and $SP_z$ to solve [BSP], find SP_obj and update $\lambda^s, \lambda^d, \lambda^c$;
            /* keep 55 best individuals in population */
            if [BSP] solution fitness is better than worst individual in current population then
                replace worst individual with current [BSP] solution
            if $SP\_obj < B\_UB$ then
                set B_UB = SP_obj and set UB_noimp = 0;
            else
                set UB_noimp = UB_noimp + 1;
            generate Benders optimality cut using $\lambda^s, \lambda^d, \lambda^c$ and add to [BMP];
            solve [BMP] to find MP_obj;
            if $MP\_obj > B\_LB$ then
                set B_LB = MP_obj and set LB_noimp = 0;
            else
                set LB_noimp = LB_noimp + 1;
            set $SP_y = y$ and $SP_z = z$ from [BMP] solution;
            if $UB\_noimp > 50$ or $LB\_noimp > 50$ then
                exit while loop;
            if overall CPU time $> 3600$ seconds then
                set time_flag = 1;
        /* do not run Routine 2 if duality gap is small or overall CPU time
           exceeds 3600 seconds */
        if $|(B\_UB - B\_LB)/B\_LB| \leq 0.001$ then
            set EA_UB = B_UB and exit for loop;
        if $time\_flag == 1$ then
            set EA_UB = B_UB and exit for loop;
        extract 55 best individuals and call Routine 2;
        if $time\_flag == 1$ then
            exit for loop;
    set Best_UB = B_UB; Lag_LB = B_LB;
    if $|(B\_UB - B\_LB)/B\_LB| > 0.001$ then
        call Routine 3;
    else
        set EA_UB = B_UB; Best_UB = EA_UB, Lag_LB = B_LB;
```

Routine 2: EA

```
begin
    for $individ = 1 : 45$ do
        randomly generate an individual;
        evaluate fitness of individual by solving $[BSP]$;
        add individual to population;
    set EA_noimp = 0; $p_m = 0.01$; best = best_UB;
    for generation = 1:100 do
        perform 2-point crossover on two most-fit individuals;
        for $offspring = 1 : 2$ do
            for $chromosome = 1 : m^* l^* n$ do
                if random number $< p_m$ then
                    flip chromosome value;
        evaluate fitness of each offspring;
        replace two least-fit individuals with two offspring;
    /* determine whether there is improvement in population fitness */
    set pop_fitness = fitness of most-fit individual;
    if pop_fitness $<$ best then
        set best = pop_fitness and set EA_noimp = 0;
    else
        set EA_noimp = EA_noimp + 1;
    if EA_noimp $>$ 25 then
        exit Routine 2;
    set EA_UB = best;
return new population of 100 individuals to Routine 1;
```

Routine 3: Subgradient

```
begin
    solve LP-relaxation of $[FL]$ and find dual values $\lambda_{LP}$ of demand constraint;
    set $\lambda^0 = \lambda_{LP}$ and set Lag_noimp = 0;
    for $k = 1 : 80$ do
        update coefficients for $[FL_\lambda(\lambda^{k-1})]$;
        solve $[FL_\lambda(\lambda^{k-1})]$ to get LagSPBound;
        if LagSPBound $>$ Lag_LB then
            set Lag_LB = LagSPBound;
        else if Lag_noimp $> 5$ then
            set $\alpha = 0.9\alpha$; Lag_noimp=0;
        else
            set Lag_noimp = Lag_noimp + 1;
        update multipliers to find $[FL_\lambda(\lambda^k)]$;
```

The models were coded in MatlabR2015a and solved by Gurobi Optimizer 5.6.2. Since the evolutionary algorithm may be invoked up to five times during one pass of Routine 1, we execute our overall heuristic only once on each problem instance to measure its performance against other heuristics in the literature.

2.3.1 Overview of SDW heuristic

The SDW heuristic proposed by Süral et al. (2009) iterates between a primal heuristic and a subgradient procedure to find successively better upper and lower bounds. We explain the mechanics of the heuristic using our notation for a multiple family setting, but remind the reader that the original heuristic pertains to a single-family problem.

To begin, the lower bound of $[FL]$ is initialized. This is accomplished by solving the LP-relaxation of original problem, and setting Lagrange multipliers equal to the dual of constraint (2.2). The upper bound is determined by solving n single-period bounded knapsack problems starting from the last period of the horizon and working towards the beginning, whereby the demand required in each period is adjusted to account for any production that takes place in future periods. Each single-period optimization problem maximizes production in a given period (or, conversely, minimizes the inventory to be held).

Once the problem of each period has been solved, a setup schedule has been obtained, indicating in which periods production is to take place. Setup variables y_{ijt} and z_{jt} are fixed accordingly, and $[FL]$ is reduced to a MCNFP. Note that the setup schedule obtained from the previous step may not be feasible, in which case, $d_{ijt}p_{ijt}$ will be greater than zero for some i, j, t.

The lower bound is obtained from the Lagrangian relaxation of constraint (2.11). Lagrange multipliers, λ_{ijt}, are updated through a subgradient procedure that uses the best upper bound obtained from the primal heuristic. These multipliers, along with a parameter β, are used to update cost coefficients of the primal heuristic, and the procedure is repeated 80 times or until the duality gap is small.

2.3.2 Overview of RHB heuristic

de Araujo et al. (2015) also relax (2.11), but employ a column generation approach. The unique feature of their RHB heuristic is its use of Lagrangian relaxation to solve both the restricted master problem of the column generation problem, as well as to generate new columns.

They find an initial upper bound through the smoothing heuristic of Trigeiro et al. (1989), and use a modified subgradient routine that considers past search directions, converging more quickly than traditional subgradient methods. The subproblem is solved with a customized algorithm that tackles single-period knapsack problems. If these subproblem solutions price out, the corresponding columns are added to the restricted master problem and the volume algorithm is applied there to find an upper bound.

The specialized algorithm used to solve the Lagrangian subproblem in this heuristic does not directly scale to the generalized case with multiple product families. For this reason, we will discuss the RHB heuristic only in the context of single-family experiments, specifically with no setup costs in the objective function (see Section 2.3.5).

2.3.3 Overview of the Lagrangian Approach

In this approach, we apply Lagrangian relaxation to $[FL]$ and generate multiple cuts at each iteration to find lower bounds quickly. Feasible solutions are found through a Benders-like integer problem that creates production schedules to be passed to a MCNFP. The combination of results from the Lagrangian and Benders-like problems allow us to measure the gap between upper and lower bounds.

Finding a Lower Bound

Decomposing $[FL_\lambda]$ by production period u, the Lagrangian bound (a lower bound to $[FL]$) can be expressed as:

$$\max_{\lambda} \left\{ \sum_{i=1}^{m}\sum_{j=1}^{l}\sum_{t=1}^{n} \lambda_{ijt} + \min_{h} \sum_{i=1}^{m}\sum_{j=1}^{l}\sum_{t=1}^{n} \left(H_{ijt}^{p} - \lambda_{ijt}\right) p_{ijt}^{h} \right.$$
$$\left. + \sum_{u=1}^{n} \left[\min_{h} \sum_{j=1}^{l} \left(\sum_{i=1}^{m} \left(Q_{iju} y_{iju}^{h} + \sum_{t=1}^{n} \left(\bar{H}_{ijut} - \lambda_{ijt} \right) w_{ijut}^{h} \right) + R_{ju} z_{ju}^{h} \right) \right] \right\}$$

where superscript h on variables $p_{ijt}, y_{iju}, w_{ijut}$, and z_{ju} denotes a feasible solution to the Lagrangian subproblem $[FL_\lambda]$. Introducing new variables

$$\theta_u^w = \min_h \sum_{j=1}^{l} \left(\sum_{i=1}^{m} \left(Q_{iju} y_{iju}^h + \sum_{t=1}^{n} \left(\bar{H}_{ijut} - \lambda_{ijt} \right) w_{ijut}^h \right) + R_{ju} z_{ju}^h \right) \qquad \forall u$$

$$\theta_{ijt}^p = \min_h \left(H_{ijt}^p - \lambda_{ijt} \right) p_{ijt}^h \qquad \forall i, j, t$$

the Lagrangian Master Problem with disaggregated cuts becomes:

$$[LMP - d] \quad \max \sum_{i=1}^{m} \sum_{j=1}^{l} \sum_{t=1}^{n} \lambda_{ijt} + \sum_{u=1}^{n} \theta_u^w + \sum_{i=1}^{m} \sum_{j=1}^{l} \sum_{t=1}^{n} \theta_{ijt}^p \tag{2.40}$$

$$\text{s.t.} \quad \sum_{i=1}^{m} \sum_{j=1}^{l} \sum_{t=1}^{n} w_{ijut}^h \lambda_{ijt} + \theta_u^w \leq \sum_{j=1}^{l} \left(\sum_{i=1}^{m} \left(Q_{iju} y_{iju}^h + \sum_{t=1}^{n} \bar{H}_{ijut} w_{ijut}^h \right) + R_{ju} z_{ju}^h \right) \quad \forall u, h \tag{2.41}$$

$$p_{ijt}^h \lambda_{ijt} + \theta_{ijt}^p \leq H_{ijt}^p p_{ijt}^h \qquad \forall i, j, t, h \tag{2.42}$$

The Lagrangian subproblem, $[FL_\lambda]$, is solved directly, with no decomposition by production period u. Lagrange multipliers are initialized by using the dual of constraints (2.11) from the LP-relaxation solution of $[FL]$. The solution from $[FL_\lambda]$ is then used to generate cuts for the Lagrangian Master Problem, $[LMP - d]$. Unlike Section 2.2.3, here the structure of $[FL_\lambda]$ is exploited, and disaggregated cuts are used to update the Lagrange multipliers. Compared to $[LMP]$, the solution time of $[LMP - d]$ is much faster because the cuts are stronger.

Iterating between $[FL_\lambda]$ and $[LMP - d]$ continues until the gap between (2.33) and (2.40) is less than 1%. Once the Lagrangian upper and lower bounds converge, the Lagrangian Bound (a lower bound to $[FL]$) is found. The corresponding $[FL_\lambda]$ solution is then used to find a better upper bound.

Finding an Upper Bound

Similar to Süral et al. (2009), we first determine the time periods in which setups will take place. Fixing those binary variables in $[FL]$, we are left with only continuous variables and the resulting problem is a MCNFP. To find the values of the binary variables, we apply an approach similar to combinatorial Benders decomposition, as in Naoum-Sawaya et al. (2015), where our "master" problem consists of variables y_{ijt} (z_{jt} can be inferred from y_{ijt}) and the subproblem is a MCNFP.

Since there are no constraints to guide the values of variables y_{ijt}, we must specify a goal for the master problem; otherwise, the setup schedule will not be meaningful. Once we begin to find good setup schedules, we make incremental changes at each iteration to preserve upper bound quality. The master problem objective minimizes the changes made to the previous setup schedule, while satisfying constraints that force at least one setup to change. The premise of these constraints is that only slight adjustments to a feasible setup schedule will be required to achieve savings in holding cost. The master problem is formulated as follows:

$$\begin{aligned} \min \quad & \sum_{i=1}^{m}\sum_{j=1}^{l}\sum_{t=1}^{n} |\bar{y}_{ijt} - y_{ijt}| \\ \text{s.t.} \quad & y_{ijt} \in \{0,1\} \end{aligned}$$

where $\bar{y}_{ijt}$ denotes the current solution. To maintain linearity in this model, we introduce variables $\tilde{y}_{ijt}$ and modify the formulation:

$$[cBMP] \quad \min \sum_{i=1}^{m}\sum_{j=1}^{l}\sum_{t=1}^{n} \tilde{y}_{ijt} \tag{2.43}$$

$$\text{s.t.} \quad \tilde{y}_{ijt} \geq \bar{y}_{ijt} - y_{ijt} \qquad \forall i,j,t \tag{2.44}$$

$$\tilde{y}_{ijt} \geq -(\bar{y}_{ijt} - y_{ijt}) \qquad \forall i,j,t \tag{2.45}$$

$$y_{ijt}, \tilde{y}_{ijt} \in \{0,1\} \qquad \forall i,j,t \tag{2.46}$$

To strengthen $[cBMP]$, we add Type 1 and Type 2 cuts (see Section 2.2.1).

Upon the first iteration of the upper bounding procedure, $[cBMP]$ is solved along with constraint sets (2.31) and (2.32). After the setup schedule is found, y_{ijt} and z_{jt} are fixed and the capacity available in t is adjusted for constraint (2.14) in problem $[FL]$. The resulting MCNFP solves quickly, with infeasibilities in the schedule identified when $d_{ijt}p_{ijt} > 0$ for any i, j, t.

If $\sum_{ijt} d_{ijt} p_{ijt} > 0$, subsequent iterations of this procedure will add a Type 3 cut to $[BMP]$:

$$\sum_{(ijt):\bar{y}_{ijt}=0} y_{ijt} + \sum_{(ijt):\bar{y}_{ijt}=1} (1 - y_{ijt}) \geq 1 \tag{2.47}$$

The best solution is maintained and the process of finding a feasible solution stops when $\sum_{ijt} d_{ijt} p_{ijt} = 0$. This feasible solution, however, can be further improved by continuing to add Type 3 cuts (2.47) until problem $[BMP]$ becomes infeasible. The upper bounding procedure will stop if no improvement in the upper bound is seen within 100 consecutive cuts.

Other cuts were tested in an effort to generate production schedules that would avoid use of initial inventories. An infeasibility is identified when initial inventory must be used to satisfy demand in a particular period. This does not necessarily mean that there is not enough capacity available in that period; rather, that period's capacity could be used to satisfy future demand for a particular t where insufficient capacity exists. Future capacity in t might not be available because an item's associated $y_{ijt} = 0$, or other items' setups whose values are 1, are consuming too much capacity. Applying cuts:

$$\sum_{(ij,t>t'):\bar{y}_{ijt}=0} y_{ijt} + \sum_{(ij,t>t'):\bar{y}_{ijt}=1} (1 - y_{ijt}) \geq 1$$

proved too restrictive. They require that at least one change be made to the current setup schedule over periods $t > t'$, where $t' + 1$ indicates the first period in which initial inventory is used. Due to many interactions across time periods, forcing schedule changes to occur over specific intervals led to substandard results.

2.3.4 Results for Problem *[P] with* setup costs

We test our heuristic on the G51-60.dat problem instances of Trigeiro et al. (1989) that have been modified to include multiple product families. The number of individual products in each family can be either 3 or 6; the number of families varies between 2, 3, or 5; and the number of time periods is 10 or 15. (See Appendix A for a full description of how these data sets were created.) The heuristic will stop when the gap between the upper and lower bounds is within 1% or the

Table 2.2: Average Duality Gap and CPU Time for $[P]$ *With* Setup Costs and Multiple Families

					SDW		Lagrangian		Benders-EA	
m	$\times$	l	$\times$	n	D%	CPU (sec)	D%	CPU (sec)	D%	CPU (sec)
3	$\times$	2	$\times$	10	95.65	22.91	52.26	781.21	**6.02**	10.54
6	$\times$	2	$\times$	10	66.63	28.72	45.64	1471.89	**24.93**	103.02
3	$\times$	2	$\times$	15	116.54	33.40	48.74	878.43	**26.71**	93.25
6	$\times$	2	$\times$	15	84.18	54.75	118.48	2588.98	**28.61**	308.14
3	$\times$	3	$\times$	10	100.36	24.09	73.33	3602.03	**10.28**	28.20
6	$\times$	3	$\times$	10	78.94	37.63	36.83	3602.86	**36.42**	155.33
3	$\times$	3	$\times$	15	114.22	47.07	49.12	2398.41	**35.15**	257.72
6	$\times$	3	$\times$	15	80.25	94.43	83.69	3610.36	**56.43**	138.75
3	$\times$	5	$\times$	10	113.93	34.23	125.32	3604.30	**20.65**	172.45
6	$\times$	5	$\times$	10	77.56	82.45	86.76	3607.33	**31.50**	125.59
3	$\times$	5	$\times$	15	118.09	76.56	81.92	3605.33	**41.00**	619.42
6	$\times$	5	$\times$	15	93.57	252.7	147.05	3625.25	**80.96**	561.04

one-hour time limit is reached. The SDW heuristic was modified to include multiple families, but all other heuristic parameters were maintained.

Table 2.2 shows how the Benders-EA approach compares to the SDW and cutting plane heuristics in terms of duality gap (*D%*) and computational time (CPU). Note that *D%* is computed as (UB – LB)/LB, and the value shown in this table is the average duality gap over the five test instances at each problem size.

In the vast majority of test cases, Benders-EA obtains better lower and upper bounds than SDW and the cutting plane method. Computational time, however, is generally longer for Benders-EA and the cutting plane method than for SDW. This is caused by the difficulty of the master problems: $[BMP]$ for Benders-EA, and $[LMP-d]$ and $[cBMP]$ for the cutting plane approach. The speed of the evolutionary algorithm and subgradient method in the Benders-EA heuristic does give it an advantage over the cutting plane approach, resulting in shorter solution times.

2.3.5 Results for Problem *[P] without* setup costs

Single-Family Experiments

We use data sets from Süral et al. (2009) that are based on those from Trigeiro et al. (1989). The heuristic stops when the gap between upper and lower bounds is within 1% or a one-hour time limit is reached. There are two test beds, one that uses the original inventory holding costs as defined by Trigeiro et al. (1989) (denoted as “het”, for *heterogeneous*), and another that modifies those inventory costs to each take a value of 1 (denoted as “hom”, for *homogeneous*). Each of these test beds contains 50 instances.

Table 2.3 compares our results to those of SDW and RHB, that are both reported in the online supplement by de Araujo et al. (2015). We observe that the lower bound obtained by RHB is always better than SDW, but the same is not true for the upper bound: the majority of the time, SDW reports a better upper bound than RHB and Benders-EA. Note that “Gap” is found by computing $\frac{100\times(UB-LB)}{LB}$.

The experiments show that, in most instances, our heuristic outperforms SDW when it comes to the quality of the lower bound, and we are within 10% of the lower bound reported by RHB, on average. However, the time required to reach such bounds and gaps are much higher than for those heuristics. From these results, we glean that our approach, while time consuming, does produce comparable lower bounds to SDW and RHB. Since better upper bounds are obtained by SDW than by RHB, in the multiple family setting, we will measure the performance of our heuristic against only SDW.

Table 2.3: Performance of SDW, RHB and Benders-EA for $[P]$ *Without* Setup Costs and a Single Family of Items

	SDW				RHB				Benders-EA			
Problem Type	LB	UB	Gap (%)	CPU(s)	LB	UB	Gap (%)	CPU(s)	LB	UB	Gap (%)	CPU(s)
12×10 het	910.70	**1101.10**	31.73	3.06	**980.90**	1108.40	**18.43**	0.42	898.58	1296.10	53.58	113.72
24×10 het	1792.20	**1951.00**	17.42	29.86	**1842.60**	2003.90	**10.85**	1.85	1792.96	2821.80	64.90	654.40
12×15 het	2706.80	3278.90	25.95	6.07	**2515.60**	**2984.30**	**20.16**	0.86	2725.88	4668.70	76.04	566.76
24×15 het	5039.40	**5559.50**	21.26	15.33	**5129.70**	5652.00	**13.44**	3.87	5042.17	10619.70	114.61	3185.32
12×30 het	8758.80	**10901.60**	28.68	24.01	**9098.20**	10961.40	**21.91**	2.99	8677.75	26127.40	208.43	3169.49
24×30 het	9751.60	**11166.00**	32.35	38.93	**9910.80**	11135.20	**23.43**	7.95	9756.47	122374.20	1033.88	2854.75
12×10 hom	595.20	**738.2**0	42.08	2.70	**641.30**	745.10	**22.97**	0.53	586.52	982.40	79.41	111.92
24×10 hom	1196.00	**1339.90**	20.60	53.37	**1236.90**	1360.50	**14.01**	1.97	1195.05	1994.00	74.00	626.45
12×15 hom	**1469.40**	1792.60	27.99	5.64	1447.78	1694.22	**19.62**	1.05	1480.69	2817.80	108.28	546.07
24×15 hom	2794.50	**3070.80**	20.56	11.14	**2840.80**	3117.80	**14.96**	3.46	2796.81	5792.90	110.44	3146.22
12×30 hom	4082.20	**4966.40**	24.33	19.10	**4208.20**	5047.40	**21.69**	2.99	4084.65	18623.80	357.50	3155.36
24×30 hom	5231.20	**5932.60**	30.47	22.91	**5332.80**	6002.40	**21.35**	7.95	5241.60	89929.80	1381.86	2806.25

Table 2.4: Average CPU Time and Improvement over SDW for $[P]$ *Without* Setup Costs and Multiple Families

m	$\times$	l	$\times$	n	SDW CPU (sec)	Benders-EA CPU (sec)	Benders-EA LB imp (%)	Benders-EA UB imp (%)	Benders-EA D% imp (%)
3	$\times$	2	$\times$	10	**7.92**	49.38	**12.19**	-26.08	-10.00
6	$\times$	2	$\times$	10	**11.18**	192.54	**0.71**	**19.09**	**24.76**
3	$\times$	2	$\times$	15	**10.21**	166.80	**0.00**	**13.40**	**18.63**
6	$\times$	2	$\times$	15	**21.65**	794.74	-0.33	-10.31	-13.72
3	$\times$	3	$\times$	10	**9.62**	111.78	-0.64	-21.85	-41.91
6	$\times$	3	$\times$	10	**13.74**	499.79	-0.01	**12.75**	**20.15**
3	$\times$	3	$\times$	15	**17.28**	400.22	**0.14**	**33.14**	**42.07**
6	$\times$	3	$\times$	15	**30.65**	2293.12	-0.20	-2.94	-10.03
3	$\times$	5	$\times$	10	**14.51**	431.06	**4.60**	**15.26**	**24.33**
6	$\times$	5	$\times$	10	**29.91**	1610.69	**2.72**	-38.57	-67.58
3	$\times$	5	$\times$	15	**24.05**	1161.52	**17.22**	**1.81**	**20.89**
6	$\times$	5	$\times$	15	**70.43**	>3600	**2.56**	-119.27	-262.05

Multiple-Family Experiments

The instances described in Section 2.3.4 are used here, but with $R_{jt} = Q_{ijt} = 0$ for all i, j, and t. Table 2.4 shows the average duality gap and computational time for each problem size.

In most test instances, we outperform the SDW heuristic in *both* upper and lower bounds with our hybrid Benders-EA approach, narrowing the duality gap in 37 out of 60 problem instances. This near-tie in the bounds obtained by the heuristics could be due to the randomness of the setup schedules generated by the evolutionary algorithm. Computation time for Benders-EA is higher. It is evident that as the problem size grows, it is much more difficult to find good solutions with either method: the Benders-EA heuristic terminates before $[BMP]$ and $[BSP]$ converge, while SDW ends after 80 iterations with duality gaps exceeding 100%. Note that "LB imp" and "UB imp" are found by dividing the difference between the Benders-EA and SDW lower (upper) bounds by the SDW lower (upper) bound, and that a negative value indicates that the Benders-EA bound is worse, on average, than the SDW bound. Due to the poor performance of the Lagrangian approach with setup costs, it was not tested against SDW or Benders-EA in this more difficult scenario without setup costs.

2.4 Remarks

Benders decomposition, when applied on its own, will converge to the optimal solution, but the difficulty of the master problem will impact the amount of time required for this to occur. In our experiments, we observed that the lower bound obtained at the end of Routine 1 was still quite far from the upper bound determined from either the evolutionary algorithm or $[BSP]$. In the majority of problem instances, the subgradient procedure was able to quickly improve the values of these lower bounds.

An obvious symptom of our hybrid Benders-EA approach is an increase in computational time. Since cuts are added at each iteration, the master problem becomes increasingly difficult to solve. However, given the fact that these big bucket lot-sizing models solve tactical planning problems, in practical settings there will be sufficient time available to apply our heuristic and have confidence that the solutions obtained are closer to optimality than other methods found in the literature.

The structure of capacity constraints (2.14), in particular the presence of family setup times, does not allow us to use the knapsack algorithm developed in de Araujo et al. (2015) to solve our Lagrangian subproblem. These constraints also complicate the Trigeiro et al. (1989) smoothing heuristic applied by Hindi et al. (2003) and de Araujo et al. (2015). Those authors, along with Jans and Degraeve (2004) and Süral et al. (2009), use subgradient optimization to find a lower bound. However, without a good feasible solution (upper bound), there is no guarantee that the subgradient method will converge to the optimal solution. These observations led us towards using a more exact method, namely Benders decomposition, for finding both upper and lower bounds.

In larger instances, specifically as the number of time periods increases, the Lagrangian approach performs poorly, as much of the 1-hour solution time is spent finding the best Lagrangian bound; the algorithm does not advance to improve the upper bound, resulting in poor duality gaps. We varied the stopping criteria for the lower bound, using gaps of 0.01%, 0.1% and 1% between the $[LSP]$ and $[LMP-d]$ solutions, and found that the time for this heuristic does improve with only slight increases in the duality gap. However, much of the computational time is still spent on finding the optimal Lagrangian bound.

Chapter 3

Production Routing for Multiple Product Families

3.1 Problem Description

Coordination within a supply chain involves decisions related not only to production planning, but distribution and routing of materials as well. In a vendor-managed inventory setting, a single supplier simultaneously determines quantities of a product to produce, ship, and transport to a number of customers. By solving this problem, the supplier will effectively meet its customers' demands at the lowest cost.

We tackle this production-routing problem (PRP) for a single supplier serving a number of customers. As discussed in Section 1.2, we address the gap in literature and propose to solve the PRP with multiple product families subject to coordinated, capacitated lot-sizing decisions (PRP-CCLSP). We examine the case where a heterogeneous fleet is used to ship finished goods to customers, and present a few scenarios that consider backordered demand and split deliveries. We develop an exact algorithm with an efficient upper bounding heuristic, along with a decomposition heuristic, to compare solution quality and computational times for both small and large problem instances.

3.2 Problem Formulation

We consider a one-to-many network consisting of a single manufacturing facility producing several items to deliver to multiple customers. The set of the manufacturing facility and customer locations is represented by N and indexed by $i \in \{0,...,n\}$, with $A = \{(i,j) : i,j \in N, i \neq j\}$ as the set of undirected arcs. The manufacturing facility, or plant, is denoted by $i = 0$, and customers are within the set $N_c = N \setminus \{0\}$. The finite planning horizon is indexed by $t = 1,2,...,T$. There are multiple families, $l = 1,2,...,L$, consisting of several items, $k = 1,2,...,K$, whose lot-sizes are coordinated due to the limited production time available in each period (CAP_t). Production of any item in family l will trigger a family setup cost, r_l, and an item-specific setup cost, q_{kl}, along with a per-unit production cost, u_{kl}. Production time is also consumed: f_l for setting up family l, b_{kl} for setting up item (k,l), and a_{kl} for each unit of (k,l) produced. Holding inventory from period t to $t+1$ costs h_{iklt} per unit of item (k,l) per period, and any backordered demand incurs a penalty of $\bar{h}_{iklt}$. Inventory can be held at both the manufacturer and customer locations, but backorders are incurred only by customers.

The manufacturer must determine the appropriate lot sizes to satisfy customer demands, d_{iklt}, along with the routes on which deliveries will travel. The fleet consists of V heterogeneous vehicles, indexed by $v = 1,2,\ldots,V$, whose delivery capacity is denoted by $\overline{CAP}^v$. We let $\bar{a}_{kl}$ denote the size of item (k,l), in terms of vehicle capacity units. Shipment routes must start and end at the manufacturer, and deliveries can be split: a single vehicle will visit a particular customer only once in a period, but a customer may be visited by different vehicles in the same period. There is a cost, c_{ij}, associated with travel between nodes i and j. See Table 3.1 for a complete list of notation used throughout this chapter.

The following decisions are made simultaneously in each period t:

- *lot-sizing*: whether the manufacturer should produce item (k,l), and if so, how much. Associated decision variables are production quantities, P_{klt}; inventory held, S_{iklt}; backordered quantities, B_{iklt}; item setups, Y_{klt}; and family setups, Z_{lt}.

- *distribution*: the quantity of (k,l) on vehicle v to ship from the manufacturer to customers, Q^v_{iklt};

- *routing*: whether customer i is visited by vehicle v in period t (W_{it}^{v}) and the sequence in which customers will be visited by vehicle v in period t (X_{ijt}^{v}).

$$[PRP] \quad \min \sum_{t=1}^{T} \left[\sum_{l=1}^{L} \left\{ \sum_{k=1}^{K} \left(u_{kl}P_{klt} + \sum_{i \in N} h_{iklt}S_{iklt} + \sum_{i \in N_c} \bar{h}_{iklt}B_{iklt} + q_{kl}Y_{klt} \right) + r_l Z_{lt} \right\} + \sum_{v=1}^{V} \sum_{(i,j) \in A} c_{ij}X_{ijt}^{v} \right] \tag{3.1}$$

$$\text{s.t.} \quad P_{klt} + S_{0kl,t-1} - S_{0klt} = \sum_{i \in N_c} \sum_{v=1}^{V} Q_{iklt}^{v} \qquad \forall k,l,t \tag{3.2}$$

$$\sum_{v=1}^{V} Q_{iklt}^{v} + S_{ikl,t-1} - B_{ikl,t-1} - S_{iklt} = d_{iklt} - B_{iklt} \qquad \forall i \in N_c, k,l,t \tag{3.3}$$

$$\sum_{l=1}^{L} \left(\sum_{k=1}^{K} \left[a_{kl}P_{klt} + b_{kl}Y_{klt} \right] + f_l Z_{lt} \right) \leq CAP_t \qquad \forall t \tag{3.4}$$

$$P_{klt} \leq M_{klt}Y_{klt} \qquad \forall k,l,t \tag{3.5}$$

$$Y_{klt} \leq Z_{lt} \qquad \forall k,l,t \tag{3.6}$$

$$Q_{iklt}^{v} \leq \tilde{M}_{it}^{v} W_{it}^{v} \qquad \forall v, i \in N_c, k,l,t \tag{3.7}$$

$$\sum_{k=1}^{K} \sum_{l=1}^{L} \sum_{i \in N_c} \bar{a}_{kl} Q_{iklt}^{v} \leq \overline{CAP}^{v} W_{0t}^{v} \qquad \forall v,t \tag{3.8}$$

$$\sum_{j \in N_c} X_{0jt}^{v} \leq 1 \qquad \forall v,t \tag{3.9}$$

$$\sum_{j \in N} X_{jit}^{v} + \sum_{j' \in N} X_{ij't}^{v} = 2W_{it}^{v} \qquad \forall v, i \in N, t \tag{3.10}$$

$$\sum_{i \in S} \sum_{j \in S} X_{ijt}^{v} \leq |S| - 1 \qquad \forall v,t,S \subseteq N_c : |S| \geq 2 \tag{3.11}$$

$$P_{klt}, S_{iklt}, B_{iklt}, Q_{iklt}^{v} \geq 0 \qquad \forall v, i \in N, k,l,t \tag{3.12}$$

$$Y_{klt}, Z_{lt}, W_{it}^{v} \in \{0,1\} \qquad \forall v, i \in N, k,l,t \tag{3.13}$$

$$X_{ijt}^{v} \in \{0,1\} \qquad \forall v, (i,j) \in A : i \neq 0, t \tag{3.14}$$

$$X_{0jt}^{v} \in \{0,1,2\} \qquad \forall v, j \in N_c, t \tag{3.15}$$

The objective function (3.1) minimizes the sum of production, inventory holding, backorders, setups, and transportation costs in each period. Constraints (3.2) and (3.3) balance inventory both at the plant and customers, respectively. Production capacity is observed in constraints (3.4), while constraints (3.5) and (3.6) ensure that production takes place only when item and family setups have occurred, where M is a very large number (say, $M_{klt} = min(\sum_{i \in N_c} \sum_{t'}^{n} d_{iklt'}, \frac{CAP_t - b_{klt} - f_{lt}}{a_{klt}}))$. The next set of inequalities concerns the routing portion of the formulation. Constraints (3.7) ensure that a delivery is made to customer i only if that customer is visited in period t where $\tilde{M}_{it}^{v} = min(\sum_{kl} \sum_{t'}^{n} d_{iklt'}, \overline{CAP}^{v})$, while constraints (3.8) ensure vehicle capacity is not violated. A vehicle may leave the depot at most once in each period (3.9). Relations (3.10) and (3.11) are the degree and subtour elimination constraints. Non-negativity, binary and integer restrictions are enforced by (3.12), (3.13), (3.14), and (3.15).

3.2.1 Modifications to prohibit split deliveries and backorders

The formulation above permits customers to be served by more than one vehicle in a particular period. At times, customers may prefer that demand arrive in a single shipment in a given period to avoid indirect costs associated with receiving multiple deliveries. In this case, prohibiting split deliveries requires an additional constraint to the formulation, indicating that at most one vehicle visit a customer in a period:

$$\sum_{v} W_{it}^{v} \leq 1 \quad \forall i \in N_c, t \tag{3.16}$$

To ensure that all customer demand is on time, we can remove backorders by omitting variables B_{iklt} and any of its associated cost parameters, $\bar{h}_{iklt}$.

3.3 Valid Inequalities

A number of valid inequalities have been developed to strengthen both the lot-sizing and vehicle routing aspects of PRP. However, their addition to the problem depends on the particular settings being studied.

Table 3.1: Summary of notation

Notation	Description	Indices or Unit of measure
Indices		
i	Index for production facility ($i = 0$) and customers	$i = 0, 1, \ldots, n$
k	Index for products	$k = 1, 2, \ldots, K$
l	Index for families	$l = 1, 2, \ldots, L$
v	Index for vehicles	$v = 1, 2, \ldots, V$
t	Index for time periods	$t = 1, 2, \ldots, T$
Parameters		
u_{kl}	Per unit production cost for item (k,l)	\$/unit
q_{kl}	Item setup for cost (k,l)	\$
r_l	Setup cost for family l	\$
c_{ij}	Cost per unit distance to ship item (k,l) along arc (i,j)	\$
h_{iklt}	Per unit holding cost for item (k,l) at customer i per time	\$/unit/time
$\bar{h}_{iklt}$	Per unit backorder cost for item (k,l) at customer i per time period	\$/unit/time
d_{iklt}	Demand of product (k,l) for location i in period t	units
a_{kl}	Unit production time for item (k,l)	time
b_{kl}	Setup time for item (k,l)	time
f_l	Setup time for family l	time
$\bar{a}_{kl}$	Vehicle capacity absorption for item (k,l)	units
CAP_t	Available production time in period t	time
$\overline{CAP}^v$	Vehicle capacity	units
Decision Variables		
P_{klt}	Amount of item (k,l) manufactured in period t	Continuous
S_{iklt}	Amount of item (k,l) held at location i from period t to $t+1$	Continuous
B_{iklt}	Amount of item (k,l) backordered at location i from period t to $t+1$	Continuous
Y_{klt}	1 if item (k,l) is produced in period t; 0 otherwise	Binary
Z_{lt}	1 if setup for family l takes place in period t; 0 otherwise	Binary
Q^v_{iklt}	Amount of product (k,l) shipped to customer i on vehicle v in period t	Continuous
X^v_{ijt}	number of times vehicle v traverses arc (i,j) in period t	Integer
W^v_{it}	1 if vehicle v visits customer i in period t	Binary

3.3.1 Lot-Sizing Inequalities

When demand must be met in each period, the following lot-sizing inequalities ensure that an appropriate number of setups occur early enough in the horizon, based on initial inventory levels and available capacity. Extending the framework established in Adulyasak et al. (2014a) to account for multiple products and multiple product families, we define

$$
\begin{aligned}
t'_{kl} &= \operatorname*{argmin}_{1\le t\le T}\Big(\sum_{i\in N_c}\max\Big\{0,\sum_{h=1}^{t} d_{iklh}-S_{ikl0}\Big\}-S_{0kl0}>0\Big) \quad \forall k,l\\
t'' &= \min_{\substack{i\in N_c\\ 1\le k\le K\\ 1\le l\le L}} \tau''_{ikl}
\end{aligned}
$$

where $\tau''_{ikl}=\operatorname*{argmin}_{1\le t\le T}\Big\{\sum_{h=1}^{t} d_{iklh}-S_{ikl0}>0\Big\}, \forall i\in N_c, k, l$. These equations imply that t'_{kl} is the earliest period in which the manufacturing facility must produce item (k,l), and t'' is the earliest period in which at least one customer must receive a replenishment for any product (k,l). We let κ be the minimum shipping quantity in period t'', where

$$
\kappa = \sum_{\substack{i\in N_c\\ 1\le k\le K\\ 1\le l\le L}} \max\Big\{0,\sum_{h=1}^{t''} d_{iklh}-S_{ikl0}\Big\}
$$

From these definitions, we add the following inequalities to formulation $[PRP]$:

$$
\sum_{t=1}^{t'_{kl}} Y_{klt} \ge 1 \qquad \forall k,l \tag{3.17}
$$

$$
\sum_{v\in V}\sum_{t=1}^{t''} W_{0t}^{v} \ge \left\lceil \frac{\kappa}{\lambda} \right\rceil \tag{3.18}
$$

Inequality (3.17) ensures that a production setup takes place at least once over the interval $(1,t'_{kl})$, while (3.18) ensures that delivery takes place at least $\left\lceil \frac{\kappa}{\lambda} \right\rceil$ times over the interval $(1,t'')$, where $\lambda = \frac{\sum_v \overline{CAP}^v}{V}$ (the average vehicle capacity).

When backorders are permitted, valid inequalities (3.17) and (3.18) do not apply, as there is no need to force production setups nor minimum shipments to occur in particular periods.

3.3.2 Routing Inequalities

With respect to vehicle routing, the following inequalities will strengthen $[PRP]$:

$$W_{it}^{v} \leq W_{0t}^{v} \qquad \forall v, i \in N_c, t \tag{3.19}$$

$$X_{ijt}^{v} \leq W_{it}^{v} \qquad \forall v, (i,j) \in A, t \tag{3.20}$$

$$X_{i0t}^{v} \leq 2W_{it}^{v} \qquad \forall v, i \in N_c, t \tag{3.21}$$

Inequality (3.19) requires vehicles to return to the depot, while (3.20) will allow a direct trip from i to j by vehicle v only if that vehicle visits node i. Inequality (3.21) indicates that if the manufacturing facility is a successor of node i on vehicle v, then customer i must be visited by the same vehicle. (Recall from equation (3.15) that the manufacturing facility, $i = 0$, is visited either 0, 1, or 2 times in any period t.)

3.4 Solution Approaches

3.4.1 Branch-and-Cut Algorithm

Valid inequalities (3.19), (3.20), and (3.21) are added to $[PRP]$ before solving. Rather than add all subtour elimination constraints (SECs) at once, these cuts are only added as needed, i.e. when a subtour has been identified.

At each node of the branch-and-bound tree, the Floyd-Warshall (Floyd, 1962; Warshall, 1962) algorithm is called to examine the relevant subgraphs for each vehicle v in period t, as given by the current values of W and X (denoted by $\bar{W}_{it}^{v}$ and $\bar{X}_{ijt}^{v}$). Note that we modify the Floyd-Warshall algorithm slightly to account for the undirected nature of our graph. For a particular vehicle v and time period t, the graph will consist only of those customers whose corresponding $\bar{W}_{it}^{v} > 0$. For any pair of nodes i, j (where $j \neq i$) in the graph, if $\bar{X}_{ijt}^{v} > 0$, then an edge (i, j) will be created with a weight of 1. The Floyd-Warshall algorithm finds the shortest path between every pair of nodes in the graph (see Routine 4). If the length of the shortest path from a particular customer to any other visited customer is infinite, then a subtour has been identified. The corresponding SEC (3.11) can then be added to the formulation.

Routine 4: Finding Subtours

```
begin
    /* Construct subgraph for a given $\bar{Z}_{lt}, \bar{X}^v_{ijt}$ */
    set $S = \emptyset$
    for $i = 1 : N$ do
        if $\bar{W}^v_{it} > 0$ then
            $S = S \cup i$
    for $i$ in $S$ do
        for $j$ in $S$: $j \neq i$ do
            if $\bar{X}^v_{ijt} > 0$ then          // add edge $(i, j)$ to $S$ with weight = 1
                set $c_{ij} = 1$
    /* Execute the Floyd-Warshall Algorithm for an undirected graph */
    for $u$ in $S$ do
        for $k$ in $S$ do          // Initialize distances and predecessors
            $d_{uk} = 1000$
            $pred_{uk} = -1$
        $d_{uu} = 0$
    for each edge $(u, k)$ do          // Set distance to current edge weight
        $d_{uk} = c_{uk}$
        $pred_{uk} = u$
    for each edge $(k, u)$ do          // Add symmetric distances to create an undirected graph
        $d_{ku} = c_{uk}$
        $pred_{ku} = k$
    for $p$ in $S$ do          // Update distances if a shorter path is found
        for $u$ in $S$ do
            for $k$ in $S$ do
                newdist = $d_{up} + d_{pk}$
                if $newdist < d_{uk}$ then
                    $d_{uk}$ = newdist
                    $pred_{uk} = pred_{pk}$
    /* Determine if subtour exists */
    set $\bar{S_1} = S$ and $\bar{S_2} = S$
    while $|\bar{S_1}| > 0$ do
        set $next = \emptyset$
        for $j$ in $\bar{S_1}$ do
            if $d_{s_1, j} > 999$ then
                $next = next \cup j$
                $\bar{S_2} = \bar{S_2} \setminus j$
        if $|\bar{S_2}| > 1$ then
            add SEC for all edges and nodes in subtour $\bar{S_2}$
        set $\bar{S_1} = next$ and $\bar{S_2} = next$
```

3.4.2 Finding an Upper Bound

While the branch-and-cut procedure is considerably faster than solving $[PRP]$ directly, there is still a certain amount of difficulty in generating good solutions within a reasonable amount of time. Aside from challenges presented by the vehicle-routing portion of the problem, the presence of coordinated lot-sizing with multiple families also adds a new layer of complexity.

We take an approach similar to Adulyasak et al. (2014b) in developing a good upper bound. The original problem is split into a production-distribution subproblem, $[PD]$, and a routing subproblem, $[R]$. The goal is to solve each of these subproblems quickly and use them as a "warm start" for the branch-and-cut algorithm.

Production-Distribution Subproblem

This subproblem identifies production quantities and customers visited in each period t. We assume direct shipments, and estimate these costs through the function

$$\sigma_i = \min\left[2c_{0i},\ \min_{j,k\in N, j\neq k}(c_{ij}+c_{ik})\right]$$

We determine allocation of shipments to vehicles and keep index v on the quantities shipped and customer visited. This simplifies the routing subproblem, since we will know which vehicle from the heterogeneous fleet will be used. In contrast, Adulyasak et al. (2014b) omit the vehicle index in their production-distribution subproblem, and instead use the Clarke-Wright algorithm to assign shipment quantities to their fleet of identical vehicles. While our method does increase the number of binary variables in $[PD]$, solutions times are still fairly quick.

$$[PD] \quad \min \quad \sum_{t=1}^{T}\left[\sum_{l=1}^{L}\left\{\sum_{k=1}^{K}\left(u_{kl}P_{klt}+\sum_{i\in N}h_{iklt}S_{iklt}+\sum_{i\in N_c}\bar{h}_{iklt}B_{iklt}+q_{kl}Y_{klt}\right) + r_l Z_{lt}\right\} + \sum_{i\in N_c}\sum_{v=1}^{V}\sigma_i W_{it}^{v}\right]$$

$$\text{s.t.} \qquad (3.2)-(3.8), (3.12), (3.13)$$

$[PD]$ is solved using a fix-and-optimize approach. First, we set all $W_{it}^{v} = 1$ to indicate that all customer locations will be visited by every truck. Then, a production schedule, Y_{klt}, is found. Those Y values are fixed, and $[PD]$ is re-solved to determine production quantities and new W values. This final solution is fixed and passed along to subproblem $[R]$. Since optimality is not necessary here, the solution tolerance is set to 1% and the branch-and-bound process terminates after 5000 nodes.

Routing Subproblem

Now that production quantities and customers visited have been decided, routing decisions are made. In keeping the vehicle index on variables Q and W, the allocation of shipments to vehicles has already been made, so we are left to determine the order in which customers are visited.

For each vehicle v in period t, we solve a traveling salesman problem (TSP). Though the entire customer network consists of n customer nodes, it is unlikely that a particular vehicle will be assigned to visit all customers in one trip, so solving a few TSPs on these subgraphs is manageable. We are not seeking an optimal solution at this stage, and only solve a TSP to its first feasible solution using the formulation below:

$$\begin{aligned} [R]_t^v \quad & \min 0 \\ \text{s.t.} \quad & (3.9), (3.10), (3.11), (3.14), (3.15), (3.20), (3.21) \end{aligned}$$

3.5 Computational Settings

A few tests were performed to determine overall formulation and computational settings to carry forward to five problem scenarios. The format for subtour elimination constraints and lot-sizing valid inequalities, as well as computational time limits were compared in Experiments A, B, and C. A brief discussion of results is below, and a complete description of how data sets are generated is provided in Section 3.8.1:

A. **Lot-sizing Valid Inequalities:** Inequalities from Adulyasak et al. (2014a) base minimum production setup decisions on the demands and initial inventory only, while inequalities

(2.32) from Section 2.2.1 consider remaining slack capacity after satisfying all demands in previous periods.

Experiments using data sets g1.dat – g27.dat with no backorders and no split deliveries allowed and using weak SECs, inequalities (3.17) and (3.18) outperformed cuts specified by (2.32), as the latter failed to tighten the formulation in any way. It is believed that demand patterns play a role in the impact those inequalities will have, and further testing should be carried out to confirm these suspicions in future work.

B. **Routing Inequalities and Constraints:** Adulyasak et al. (2015) note that SECs (3.11) can be re-stated as:

$$\sum_{i \in S} \sum_{j \in S} X_{ijt}^{v} \leq \sum_{i \in S} W_{it}^{v} - W_{et}^{v} \qquad \forall S \subseteq N_c : |S| \geq 2, \forall e \in S, v, t \tag{3.22}$$

In the presence of multiple vehicles, constraints (3.22) provide better root node solutions in less time, while they reduce to (3.11) when only a single vehicle is considered. A comparison of these two forms of SECs was conducted on instances g1.dat – g27.dat. Using SECs (3.22) improved the root gap % in 19 out of 27 instances, and solved to optimality more quickly in 21 out of 27 instances (Figure 3.1).

C. **Graph time-gap data.** Gap % versus elapsed time is plotted in Figure 3.2 for instances g1.dat – g27.dat using the best settings determined from Experiments A and B (no backorders, no split deliveries, lot-sizing valid inequalities, and strong SECs). For the most part, solution quality plateaus by 1200 seconds, but to ensure adequate solution time for larger instances, a CPU time limit of 1800 seconds has been imposed moving forward.

3.6 Scenario Descriptions

The PRP with multiple families is tested under a number of inventory and routing scenarios to study the impact of these constraints on overall cost, computational time, and vehicle utilization. We refer the reader to subsection 3.2.1 for a description of constraint and variable modifications.

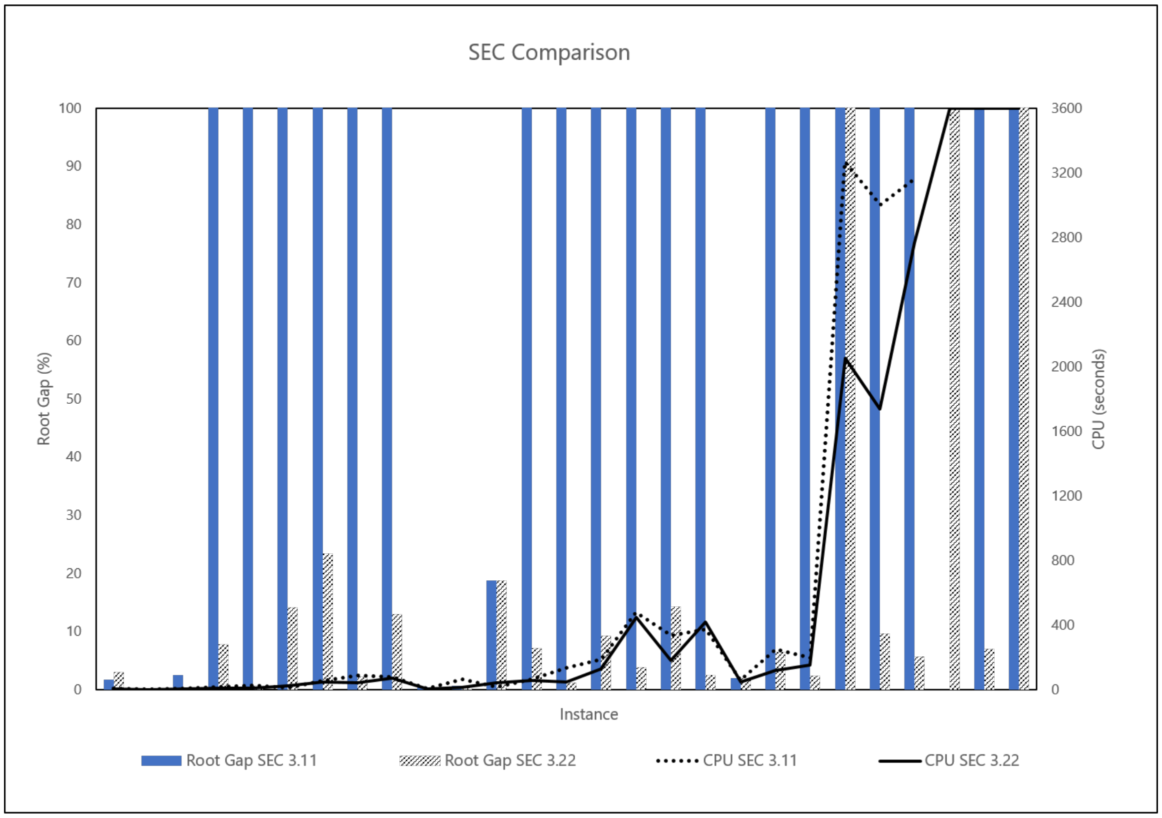

Figure 3.1: Root Gap and CPU Performance using SECs (3.11) and (3.22)

1. **Traditional (S1):** No backorders and no split deliveries allowed. This represents a typical PRP model, where demands must be met on time and each customer is visited by at most one vehicle in each period. The benefits of this setting are that customers receive at most one delivery in a period and demand is always fulfilled.

2. **Flexible Transportation and Demand (S2):** Backorders and split deliveries are allowed. Here, the manufacturer tries to maximize vehicle utilization by allowing customers to be visited by more than one vehicle in a given period. Demand may not necessarily be met on time, also in an effort to maximize truck utilization – delaying delivery of some demand may better fill a truck.

3. **Flexible Transportation (S3):** Split deliveries allowed, but backorders are not. Here,

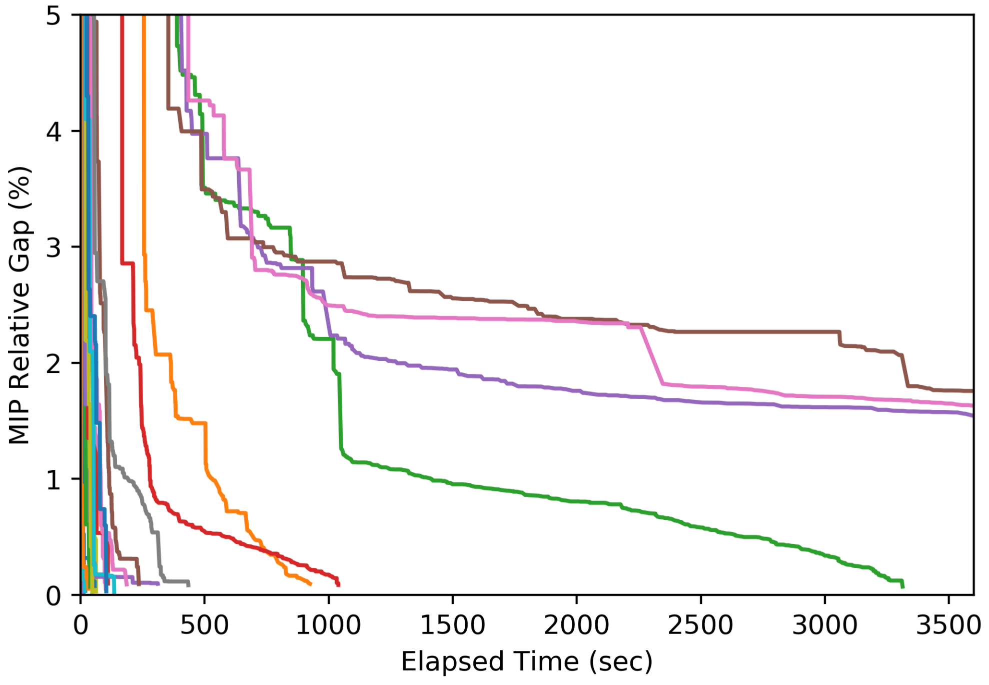

Figure 3.2: Time versus Gap % Data

demand must be met on time but customers can receive more than one shipment per period.

4. **Flexible Demand (S4):** Backorders are allowed, but split deliveries are not. This aims to improve vehicle utilization (and even capacity utilization at the production-level), without impacting the customer's receiving process. Customers will receive at most one shipment per period, but it may not necessarily satisfy all of their demands on time.

5. **Dedicated Routes (S5):** In scenarios S1 through S4, all production, distribution, and routing decisions are made simultaneously, with customers potentially being served by a different vehicle in each period. With dedicated routes, customers will be clustered into districts and assigned a specific vehicle that will serve that cluster for the entire horizon. Allocation and routing decisions will be made only once, while production and distribution decisions will continue to be made on a period-by-period basis.

Scenarios 1 through 4 are implemented by making slight adjustments to constraints and variables in the original [PRP] formulation, and are solved using the branch-and-cut algorithm coupled with the upper bounding heuristic. Scenario 5 calls for routing and production decisions to be carried out sequentially, and is solved by the Dedicated Routes Heuristic, which is detailed next.

3.7 Scenario 5: Dedicated Routes Heuristic

PRP-CCLSP with dedicated routes first solves a districting problem to determine customer clusters (or districts). Each cluster remains unchanged over the time horizon, and so only production and shipments are determined in the second phase.

3.7.1 Phase 1: Clustering Customers

Kalcsics (2015) describes a number of criteria that are useful in creating districts, such as balancing demand across districts, contiguity of districts, and compactness. Here, we choose to cluster customers based on geographical proximity and total demands over the planning horizon. Once grouped together, the order in which customers are visited is determined, and is fixed during Phase 2 when production and shipment quantities are found.

Districting Formulation

Let λ_{ij}^{v} indicate whether or not customer j is assigned to district center i served by vehicle v, and let $\sigma_i = 1$ if customer i is designated as a district center and 0 otherwise. This designation of "district center" is only needed for the purposes of the objective function, which minimizes the number of customers in each district. Since we have a heterogeneous fleet of vehicles, variable γ_i^v represents whether or not district i is serviced by vehicle v (for a homogeneous fleet, this variable would not be necessary). Parameter D_j denotes customer j's total demand over the horizon and is equal to $\sum_{klt} d_{jklt}$.

$$\min \quad \sum_{(i,j)\in N_c} \sum_{v\in V} D_j c_{ij}^2 \lambda_{ij}^v \tag{3.23}$$

$$\text{s.t.} \quad \sum_{i\in N_c} \sum_{v\in V} \lambda_{ij}^v = 1 \qquad \forall j \in N_c \tag{3.24}$$

$$\sum_{i\in N_c} \sigma_i \leq V \tag{3.25}$$

$$\sum_{v\in V} \gamma_i^v \leq \sigma_i \qquad \forall i \in N_c \tag{3.26}$$

$$\sum_{i\in N_c} \gamma_i^v = 1 \qquad \forall v \in V \tag{3.27}$$

$$\sum_{j\in N_c} D_j \lambda_{ij}^v \leq T \cdot \overline{CAP}^v \gamma_i^v \qquad \forall i \in N_c, v \in V \tag{3.28}$$

$$\sum_{j\in N_c} D_j \lambda_{ij}^v \geq \alpha^v T \cdot \overline{CAP}^v \gamma_i^v \qquad \forall i \in N_c, v \in V \tag{3.29}$$

$$\lambda_{ij}^v, \gamma_i^v, \sigma_i \in \{0,1\} \qquad \forall i,j \in N_c, v \in V \tag{3.30}$$

The objective function minimizes the weighted "moment of inertia" of the district. Equation (3.24) assigns customer j to a single district center i and constraint (3.25) limits the number of district centers according the number of vehicles in the fleet. Vehicle v is assigned to district center i only if that location has been chosen as a center (3.26), and a vehicle must be assigned to exactly one district center (3.27). Inequalities (3.28) and (3.29) specify the maximum and minimum quantities that can be shipped within the district. The maximum threshold is based on the total vehicle capacity over the horizon, while the minimum threshold is linked to a measure of vehicle utilization that is quantified by α^v. These α^v may vary depending on vehicle size; to encourage smaller vehicles to be filled first, their α values may be larger in relation to those α for larger vehicles in the fleet. If a more balanced approach is desired (i.e. the proportion that each vehicle is filled should be roughly equal, regardless of its nominal capacity), the same value of α may be assigned to each vehicle.

As discussed by Kalcsics (2015), there are other expressions aside from objective function (3.23) that could be used to achieve a compact district. While each may result in a different district, there is no single measure that dominates the others.

Districting Solution Approach

This location-allocation formulation is typically solved by decomposition (Kalcsics, 2015). An initial set of district centers i is fixed and the resulting subproblem allocates customers to these centers. Using these allocations, new district centers are determined by solving, for each cluster, a single facility location problem.

The clustering heuristic will:

1. Initialize district centers σ_i,
2. Fix σ_i and solve the resulting allocation subproblem to obtain λ_{ij}^{v} and γ_i^{v}.
3. Compute a weighted center of gravity using the allocations from step (2) to find a new district center σ_i for each district.
4. Repeat steps (2) and (3) until the solution converges.

Initialization

To initialize district centers, the k-means++ algorithm (Arthur and Vassilvitskii, 2007) assigns cluster centers based on a weighted probability. In this problem, total horizon demands (D_j) are used as the weights. Vector c_{ib} represents the minimum distance from customer i to any location b that is already designated as a cluster center. Routine 5 shows the pseudocode:

Routine 5: Initializing Cluster Centers

```
1 begin
      /* Initialize variables                                          */
2     set $\bar{C} = \emptyset$, $c_{ib} = \emptyset$, num_centers = 0
      /* Execute $k$-means++ algorithm                                  */
3     while $num\_centers < V$ do
4         if $\bar{C} = \emptyset$ then
5             set $\sigma_i = 1$ w.p. $D_i / \sum_j D_j$
6         else
7             set $\sigma_i = 1$ w.p. $D_i c_{ib}^2 / \sum_j D_j c_{jb}^2$
```

Once assigned, the set of cluster centers can be denoted as $\bar{C} = \{\sigma_1, \sigma_2, ..\sigma_v\}$ where $|\bar{C}| = V$ and $\bar{C} \subset N_c$.

Allocating Customers to Clusters

Fixing district center variable values as $\bar{\sigma}_i$, the allocation subproblem is:

$$\begin{aligned}
\min \quad & \sum_{i\in\bar{C}}\sum_{j\in N_c}\sum_{v\in V} D_j c_{ij}^2 \lambda_{ij}^v && && (3.31)\\
\text{s.t.} \quad & \sum_{i\in\bar{C}}\sum_{v\in V} \lambda_{ij}^v = 1 && \forall j \in N_c && (3.32)\\
& \gamma_i^v \leq \bar{\sigma}_i && \forall i \in \bar{C}, v \in V && (3.33)\\
& \sum_{i\in\bar{C}} \gamma_i^v = 1 && \forall v \in V && (3.34)\\
& \sum_{j\in N_c} D_j \lambda_{ij}^v \leq T \cdot \overline{CAP}^v \gamma_i^v && \forall i \in \bar{C}, v \in V && (3.35)\\
& \sum_{j\in N_c} D_j \lambda_{ij}^v \geq \alpha^v T \cdot \overline{CAP}^v \gamma_i^v && \forall i \in \bar{C}, v \in V && (3.36)\\
& \lambda_{ij}^v, \gamma_i^v \in \{0,1\} && \forall i \in \bar{C}, j \in N_c, v \in V && (3.37)
\end{aligned}$$

Using a vector of minimum thresholds based on vehicle size is just one way to influence the allocation of customers to cluster. Alternative methods may include constraints specifying a particular number of customers that must be served by each vehicle, or assigning a penalty for using certain vehicles, e.g. an extra unit charge if a customer is served by a small vehicle will influence the model to allocate customers to a larger vehicle first.

Updating Cluster Centers

Using the clusters developed from the allocation subproblem, new cluster centers are computed. Letting C_v be the set of customers in cluster v (served by vehicle v) and $j, j' \in C_v$ be customers in cluster v, we compute

$$\arg\min_{j\in C_v} \sum_{j'\in C_v} D_{j'} c_{j'j}^2 \tag{3.38}$$

to find a new cluster center j, or σ_j. This value is passed to the allocation subproblem to determine new customer allocations, and the process repeats until no changes in the allocations occur (or for an arbitrary number of iterations).

3.7.2 Phase 2: Determining Vehicle Routes

Once customers are assigned to clusters, vehicle routes are determined. These routes stay fixed over the horizon, regardless of demand patterns. This could mean that in certain periods, a customer may not be visited by a vehicle, but the sequence of customer visits will remain unchanged.

The vehicle routing portion of PRP reduces to V TSP problems, solved only once:

$$[TSP]^v \qquad \min \sum_{i\in\{0\cup C_v\}} \sum_{j\in\{0\cup C_v\}} c_{ij}X_{ij}^v \tag{3.39}$$

$$\text{s.t.} \quad \sum_{j\in C_v} X_{ij}^v + \sum_{j'\in C_v} X_{ij}^v = 2 \qquad \forall i \in \{0 \cup C_v\} \tag{3.40}$$

$$\sum_{i\in C_v} \sum_{j\in C_v} X_{ij}^v \leq |S| - 1 \qquad \forall S \in C_v : |S| \geq 2 \tag{3.41}$$

$$X_{ij}^v \in \{0,1\} \qquad \forall (i,j) \in \{0 \cup C_v\} \tag{3.42}$$

Objective function (3.39) minimizes the total distance traveled in one period when all customers in cluster v are visited. (Recall, C_v denotes the set of customers served by vehicle v.) Degree constraints (3.40) ensure there is exactly one arc into and one arc out of a particular node, and subtours are eliminated through constraints (3.41). Note that subtour elimination constraints (SECs) from the original formulation include variables W_{it}^v; in each cluster's TSP, every $W_{it}^v = 1$ and so those SECs reduce to constraints (3.41).

3.7.3 Phase 3: Determining Production and Distribution Quantities

Since routing and production/distribution decisions have been separated, Phases 2 and 3 can be solved in parallel, as their outcomes do not impact one another.

From Phase 1, each vehicle has enough total capacity to meet aggregate demand for its cluster over the planning horizon. In a particular period, however, this might not be the case. To account for any periods of demand exceeding vehicle capacity, backlogging (or outsourcing) at a high cost must be allowed to prevent infeasibility, and this is reflected in the production-distribution formulation:

$$[PD2]^{C_v} \quad \min \sum_{t=1}^{T}\sum_{l=1}^{L}\left\{\sum_{k=1}^{K}\left(u_{kl}P_{klt}+\sum_{i\in N}h_{iklt}S_{iklt}+\sum_{i\in N_c}\bar{h}_{iklt}B_{iklt}+q_{kl}Y_{klt}\quad\right)+r_lZ_{lt}\right\} \tag{3.43}$$

$$\text{s.t.} \quad P_{klt}+S_{0kl,t-1}-S_{0klt}=\sum_{i\in C_v}\sum_{v=1}^{V}Q^v_{iklt} \qquad \forall k,l,t \tag{3.44}$$

$$Q^v_{iklt}+S_{ikl,t-1}-B_{ikl,t-1}-S_{iklt}=d_{iklt}-B_{iklt} \qquad \forall v\in V, i\in C_v, k,l,t \tag{3.45}$$

$$\sum_{l=1}^{L}\left(\sum_{k=1}^{K}\left[a_{kl}P_{klt}+b_{kl}Y_{klt}\right]+f_lZ_{lt}\right)\le CAP_t \qquad \forall t \tag{3.46}$$

$$P_{klt}\le MY_{klt} \qquad \forall k,l,t \tag{3.47}$$

$$Y_{klt}\le Z_{lt} \qquad \forall k,l,t \tag{3.48}$$

$$\sum_{k=1}^{K}\sum_{l=1}^{L}\sum_{i\in C_v}\bar{a}_{kl}Q^v_{iklt}\le\overline{CAP}^v \qquad \forall v,t \tag{3.49}$$

$$P_{klt},S_{iklt},B_{iklt},Q^v_{iklt}\ge 0 \qquad \forall v,i\in N,k,l,t \tag{3.50}$$

$$Y_{klt},Z_{lt}\in\{0,1\} \qquad \forall k,l,t \tag{3.51}$$

This model is similar to $[PD]$, though routing variables W^v_{it} and costs σ_i are omitted. Constraints (3.45) to (3.49) are analogous to constraints (3.3) to (3.6) and (3.8).

3.8 Computational Results

Scenarios 1 through 4 are solved via the branch-and-cut algorithm, which is tested on its own, as well as with warm-starting from the upper bounding heuristic. Scenario 5 is solved with the Dedicated Routes Heuristic. Both solution methods are coded in Python 2.7 and call CPLEX 12.7 to solve $[PRP]$ with tolerance set to 0.1%. The branch-and-cut algorithm was run on a PC with dual-core 2.61GHz processor and 39GB RAM. The upper bounding procedure was run on a Lenovo laptop with Intel Core i7 2.9GHz processor and 8GB of RAM. We limit our study to the case of two product families only ($L = 2$), though our formulation and solution approach can be extended to any value of L.

3.8.1 Data Set Creation

We use the 50-customer data instances developed by Boudia et al. (2005) (Set B50), modifying them to include multiple product families and a heterogeneous fleet of vehicles. For each problem size considered, three instances (a, b, and c) are generated: customer coordinates are taken from instances 1, 11, and 21 of Set B50; demand for instance a corresponds to the first 9 periods of demand of instances 1 through 10; instance b uses the first 9 periods of demand from instances 11 through 20; and instance c uses the first 9 periods of demand of instances 21 through 30.

To illustrate how demands from Set B50 are assigned in the multi-family case: demand from B50 instance 1 corresponds to demand for item $(k,l) = (1,1)$ of instance a; demand from B50 instance 2 corresponds to item $(k,l) = (1,2)$ of instance a; and demand from B50 instance 3 corresponds to item $(k,l) = (2,1)$ of instance a.

All other problem parameters are generated randomly as follows: unit production costs u_{kl} are randomly generated over the discrete uniform distribution $(20, 30, 40, ..., 80)$; item setup costs $q_{kl} = 10u_{kl}$; family setup costs $r_l = \frac{10\sum_k q_{kl}}{K}$; travel cost c_{ij} is the Euclidean distance between nodes i and j; holding cost at the plant $h_{0kl} = \lceil 0.05u_{kl} \rceil$; holding costs at customer locations h_{ikl} are randomly generated over the discrete distribution $U(6,9)$ for $i \in N_c$; backorder costs $\bar{h}_{ikl} = 15h_{ikl}$ for $i \in N_c$; unit production times are $a_{kl} = 1$ for all k,l; item setup times b_{kl} are randomly generated over the discrete uniform distribution $(10, 20, ..., 100)$; family setup times $f_l = \frac{10\sum_k b_{kl}}{K}$; production capacity $CAP = \left\lfloor \frac{2\sum_{iklt} d_{iklt}}{T} \right\rfloor$, and unit vehicle absorption $\bar{a}_{kl} = 1$ for all k,l.

Similar to Adulyasak (2017), vehicle capacity, $\overline{CAP}^v$, is based on the number of customers in each instance, as well as the maximum of the maximum allowable inventory over all customers in N_c. Note that we do not restrict the amount of inventory allowed at each customer in our formulation, but this value (equal to 1200) is provided in the data sets of Boudia et al. (2005). Since we are considering a heterogeneous fleet of vehicles as well as multiple items, these characteristics must be factored into the vehicle capacity calculation. We use the following equation to set vehicle capacity:

$$\overline{CAP}^{v} = \frac{\alpha\beta^{\lambda}\gamma(1200)}{V}$$

where λ = S (small), M (medium), L (large), $\alpha = \left\lfloor \frac{n}{10} \right\rfloor + 1, \beta^{S} = 2, \beta^{M} = 3, \beta^{L} = 6, \gamma = \frac{KL}{2}$ and V corresponds to the total number of vehicles in the fleet. Problem sizes are listed in Table 3.2. With three instances (a,b,c) at each problem size, a total of 108 problems are tested.

Boudia et al. (2005) initialize their model by setting beginning inventory to zero at all customer locations and allowing production to begin only in period $t = 2$. As such, initial inventory at the manufacturing facility is established to satisfy period 1's demand exactly for all customers.

Table 3.2: Problem Sizes

n	K	L	T	V	Fleet composition
10	1	2	3/6/9	3	SSM
10	2	2	3/6/9	4	SSSM
10	5	2	3/6/9	5	SSSMM
20	1	2	3/6/9	3	SSM
20	2	2	3/6/9	4	SSSM
20	5	2	3/6/9	5	SSSMM
30	1	2	3/6/9	4	SSSM
30	2	2	3/6/9	5	SSSMM
30	5	2	3/6/9	6	SSSMMM
40	1	2	3/6	4	SSML
40	2	2	3/6	5	SSMML
40	5	2	3/6	6	SSSMML
50	1	2	3	4	SSML
50	2	2	3	5	SSMML
50	5	2	3	6	SSMMML

S: small truck, M: medium truck, L: large truck
e.g. "SSM" indicates that the fleet consists of two small trucks and one medium truck

3.8.2 Impact of the Upper Bounding Heuristic

For each of Scenarios 1 through 4, $[PRP]$ is solved with only the branch-and-cut algorithm, and then solved again but with the algorithm warm-started using results of the upper bounding heuristic. As instances become larger and more difficult to solve, we encountered many computing crashes within the Python environment, and so only problems g1.dat–g72.dat have been solved without the UB heuristic. (These difficulties did not occur when the branch-and-cut algorithm was warm-started.) Average results over problem sets for each scenario are shown in Tables 3.3 – 3.6, where %LB indicates the final lower bound as a percentage of the final upper bound and CPU denotes the computational time in seconds. Similarly, %RLB shows the lower bound obtained at the root node as a percentage of the upper bound at the root node, and RCPU is computational time at the root node.

For all scenarios, performances without and with the UB heuristic are nearly identical when comparing %LB values. The benefit of the UB heuristic is seen more evidently in the smaller root gap (%RLB), particularly for Scenarios 2, 3, and 4 when transportation and demand flexibility are introduced. The improvement there speeds overall solution time for these three scenarios, compared to when no warm-start is used.

Since solution time is limited to 30 minutes, it is likely that further improvement in the overall gap would occur without and with the UB heuristic. Moreover, as solution tolerance is currently set to 0.1%, it is also highly probable that warm-starting the branch-and-cut algorithm provides even more benefit than what is reported in Tables 3.3 – 3.6, as CPLEX terminated early in many instances.

Table 3.7 shows the average breakdown of computational time required to solve each subproblem within the upper bounding heuristic. Overall time to compute a good upper bound is fairly low for problem instances across all scenarios. Differences lie in the ratio of time spent on solving $[PD]$ versus $[R]$. For Scenarios 1 and 4, most effort is focused on determining production-distribution decisions, while for Scenarios 2 and 3 (when split deliveries are allowed), the routing decisions become more time-consuming. In general, subproblem $[R]$ solves quickly since we search only for the first feasible solution and are not concerned with its distance from the optimal solution. Even with solution tolerance set to 1%, the longer solution time can be attributed to allocating products to vehicles (i.e. having more binary variables W_{it}^{v} due to the vehicle index).

Table 3.3: Scenario 1 Summary Without and With UB Heuristic

					Without UB Heuristic					With UB Heuristic				
n	*K*	*L*	*T*	*V*	%LB	Nodes	CPU	%RLB	RCPU	%LB	Nodes	CPU	%RLB	RCPU
10	1	2	3	3	99.96	65.33	3.09	98.88	1.56	99.97	42.33	5.40	99.33	4.23
10	1	2	6	3	99.95	654	15.80	92.48	4.51	99.93	72.33	46.19	98.76	43.22
10	1	2	9	3	99.92	1781	58.27	87.12	6.47	99.93	714.67	175.50	98.15	146.43
10	2	2	3	4	99.93	1328.67	21.84	93.41	3.76	99.92	106.67	17.16	99.36	1.99
10	2	2	6	4	99.94	2346.67	82.98	94.12	15.68	99.93	220	190.36	99.02	171.67
10	2	2	9	4	99.92	4915	372.81	93.15	95.51	99.94	6511.33	606.89	93.15	270.56
10	5	2	3	5	99.92	1984.67	110.94	96.38	24.80	99.93	1036.67	147.67	98.08	82.19
10	5	2	6	5	99.74	3610.67	1782.85	inf	88.80$^{(2)}$	99.53	2981	1661.94	94.29	164.94$^{(2)}$
10	5	2	9	5	98.22	2346	1800.17	inf	210.99$^{(3)}$	98.15	2376	1800.18	inf	330.08$^{(3)}$
20	1	2	3	3	99.97	340	28.64	99.16	10.81	99.97	74	55.31	98.79	48.14
20	1	2	6	3	99.96	894	125.91	98.90	53.06	99.95	316	221.32	99.08	168.54
20	1	2	9	3	99.93	3626.67	470.27	97.02	137.69	99.97	2930.33	658.22	97.15	296.00
20	2	2	3	4	99.94	500.5	136.42	99.84	66.75	99.94	635.67	147.77	98.02	80.13
20	2	2	6	4	99.91	4002.33	645.77	inf	103.01	inf	1205.33	514.08	inf	213.62
20	2	2	9	4	99.08	2543	1822.69	inf	166.57$^{(3)}$	inf	2689.67	1800.32	inf	298.64$^{(3)}$
20	5	2	3	5	99.94	879.33	432.41	97.92	120.22	97.97	941	637.82	97.97	221.85
20	5	2	6	5	98.98	1318.67	1800.17	96.24	397.11$^{(3)}$	96.24	1172	1800.23	96.24	491.30$^{(3)}$
20	5	2	9	5	97.55	648.5	1800.21	inf	652.45$^{(2)}$	inf	562	1800.20	inf	704.16$^{(2)}$
30	1	2	3	4	99.95	1913	458.83	97.25	57.15	98.13	1572.67	592.90	98.13	220.44
30	1	2	6	4	99.96	1247	657.30	inf	177.02	inf	1762.33	1098.92	inf	354.52
30	1	2	9	4	97.82	1574.33	1800.39	inf	437.29$^{(3)}$	inf	1325.33	1800.38	inf	706.97$^{(3)}$
30	2	2	3	5	99.95	1248.67	772.04	96.31	267.38	96.31	1317	990.79	96.31	370.77
30	2	2	6	5	99.20	1090	1632.05	inf	525.47$^{(1)}$	inf	1297	1733.97	inf	616.33$^{(2)}$
30	2	2	9	5	92.05	74	1800.23	91.93	1385.73$^{(1)}$	–	–	–	–	–
30	5	2	3	6	–	–	–	–	–	97.98	824	1800.17	97.98	378.27$^{(2)}$
30	5	2	6	6	–	–	–	–	–	96.47	50	1801.21	96.47	612.77$^{(2)}$
30	5	2	9	6	–	–	–	–	–	–	–	–	–	–
40	1	2	3	4	–	–	–	–	–	95.69	2466.33	1210.42	95.69	232.03
40	1	2	6	4	–	–	–	–	–	inf	1444	1577.36	inf	498.16$^{(1)}$
40	2	2	3	5	–	–	–	–	–	inf	696.67	1295.74	inf	304.26
40	2	2	6	5	–	–	–	–	–	95.38	300	1806.61	95.38	913.89$^{(1)}$
40	5	2	3	6	–	–	–	–	–	–	–	–	–	–
40	5	2	6	6	–	–	–	–	–	–	–	–	–	–
50	1	2	3	4	–	–	–	–	–	inf	1242.67	1517.47	inf	434.68$^{(2)}$
50	2	2	3	5	–	–	–	–	–	inf	430.33	1574.73	inf	483.90$^{(1)}$
50	5	2	3	6	–	–	–	–	–	–	–	–	–	–

(#) Indicates number of instances (out of 3) that did not solve to optimality
– Indicates that no incumbent solution was found at the root node within the 1-hour time limit for any instance

Table 3.4: Scenario 2 Summary Without and With UB Heuristic

					Without UB Heuristic					With UB Heuristic				
n	*K*	*L*	*T*	*V*	%LB	Nodes	CPU	%RLB	RCPU	%LB	Nodes	CPU	%RLB	RCPU
10	1	2	3	3	99.98	15.33	3.29	99.14	2.69	99.98	22	4.41	99.16	3.82
10	1	2	6	3	99.95	229.33	13.74	98.34	6.07	99.97	139.33	19.79	98.64	14.27
10	1	2	9	3	99.94	2007.33	63.78	91.52	11.31	99.94	311	98.68	98.28	82.22
10	2	2	3	4	99.96	80	7.76	95.44	1.41	99.96	1	7.30	99.64	3.81
10	2	2	6	4	99.96	916.67	69.28	94.14	21.00	99.94	187	71.02	98.84	53.81
10	2	2	9	4	99.93	1839.33	193.40	85.17	34.27	99.95	691	220.74	97.77	122.34
10	5	2	3	5	99.95	439.33	73.53	95.85	24.85	99.94	251	68.52	97.55	43.57
10	5	2	6	5	99.58	1739	1515.58	80.94	105.55(1)	99.67	1218.67	1728.93	96.75	205.23(1)
10	5	2	9	5	98.03	2062.67	1800.17	62.77	140.04(3)	97.69	2123	1800.18	91.57	179.21(3)
20	1	2	3	3	99.96	133.5	26.55	99.08	15.79	99.98	156.5	54.18	99.10	43.12
20	1	2	6	3	99.96	336.67	92.95	98.68	48.16	99.93	317	170.04	98.75	120.04
20	1	2	9	3	99.92	873.33	263.46	82.60	64.76	99.91	1255.33	439.92	93.29	161.33
20	2	2	3	4	99.95	321	72.24	97.02	29.23	99.96	44	83.76	99.48	72.84
20	2	2	6	4	99.94	456.67	282.49	87.98	100.55	99.93	442.67	310.41	98.01	150.05
20	2	2	9	4	98.46	1747.33	1800.30	43.46	207.59(3)	98.18	1492	1842.33	74.28	249.61(3)
20	5	2	3	5	99.92	1843.67	1338.74	86.36	117.93(1)	99.66	1372	1131.15	93.34	158.64(1)
20	5	2	6	5	99.16	879.67	1800.19	60.40	347.71(3)	98.82	925.33	1800.27	74.31	420.85(3)
20	5	2	9	5	97.63	637	1800.26	inf	759.76(3)	77.17	570	1800.42	inf	794.32(3)
30	1	2	3	4	99.98	926	289.53	97.73	90.42	99.96	643.67	241.87	99.07	133.33
30	1	2	6	4	99.96	1142.67	796.42	90.56	171.42	99.96	1260.67	897.94	90.56	216.93
30	1	2	9	4	98.21	944.33	1800.87	61.17	448.69(3)	98.25	1005.33	1801.10	61.17	510.71(3)
30	2	2	3	5	99.96	1237.33	820.95	95.35	128.10	99.97	1260	832.09	95.48	150.74
30	2	2	6	5	99.73	825	1507.06	36.14	374.53(1)	99.69	808.67	1435.20	36.14	366.00(1)
30	2	2	9	5	77.26	370	1800.27	inf	1067.47(2)	77.26	390	1800.22	inf	989.44(2)
30	5	2	3	6	–	–	–	–	–	99.15	534.67	1800.24	39.06	334.96(3)
30	5	2	6	6	–	–	–	–	–	–	–	–	–	–
30	5	2	9	6	–	–	–	–	–	–	–	–	–	–
40	1	2	3	4	–	–	–	–	–	99.92	1328	738.06	97.88	155.69
40	1	2	6	4	–	–	–	–	–	99.98	934.5	1288.35	54.75	244.90
40	2	2	3	5	–	–	–	–	–	99.18	669.5	1352.52	74.19	285.98(1)
40	2	2	6	5	–	–	–	–	–	99.59	500	1800.42	inf	640.23(1)
40	5	2	3	6	–	–	–	–	–	27.92	30	1800.19	27.89	1219.45(1)
40	5	2	6	6	–	–	–	–	–	–	–	–	–	–
50	1	2	3	4	–	–	–	–	–	98.86	1300	1800.27	83.52	318.08(1)
50	2	2	3	5	–	–	–	–	–	99.93	221	1718.12	inf	519.93
50	5	2	3	6	–	–	–	–	–	36.96	20	1800.32	36.85	1479.09(2)

(#) Indicates number of instances (out of 3) that did not solve to optimality
– Indicates that no incumbent solution was found at the root node within the 1-hour time limit for any instance

Table 3.5: Scenario 3 Summary Without and With UB Heuristic

					Without UB Heuristic					With UB Heuristic				
n	*K*	*L*	*T*	*V*	%LB	Nodes	CPU	%RLB	RCPU	%LB	Nodes	CPU	%RLB	RCPU
10	1	2	3	3	99.97	13.33	2.04	99.79	1.35	99.98	6.33	9.04	99.81	8.44
10	1	2	6	3	99.95	430	15.79	97.14	4.96	99.95	202.67	64.21	99.30	58.73
10	1	2	9	3	99.91	891.67	40.44	84.82	8.23	99.95	621	100.21	98.34	77.19
10	2	2	3	4	99.96	0.33	2.62	99.98	1.01	99.96	0.33	7.42	99.98	5.79
10	2	2	6	4	99.94	610.33	47.14	85.65	12.23	99.96	256.67	81.53	99.14	61.98
10	2	2	9	4	99.91	3036	229.17	86.94	24.11	99.91	1427.33	216.79	97.34	101.08
10	5	2	3	5	99.97	605.33	58.25	94.27	19.06	99.98	200.67	51.82	97.86	27.00
10	5	2	6	5	99.67	1516	1534.32	92.89	118.11$^{(1)}$	99.79	806.33	981.60	97.35	286.08$^{(1)}$
10	5	2	9	5	98.08	1462.33	1800.17	92.71	168.08$^{(3)}$	98.24	971.33	1800.19	96.35	226.22$^{(3)}$
20	1	2	3	3	99.98	51	19.65	99.66	15.12	99.98	51	41.17	99.66	36.41
20	1	2	6	3	99.97	242	69.89	94.22	36.11	99.96	309.67	157.11	99.20	115.84
20	1	2	9	3	99.91	585	243.99	85.90	64.18	99.92	1501.67	326.98	93.23	112.36
20	2	2	3	4	99.97	394	78.47	99.79	27.30	99.93	349.5	135.32	99.89	82.87
20	2	2	6	4	99.95	898.67	287.58	96.13	114.65	99.94	333.33	298.14	97.15	159.91
20	2	2	9	4	98.95	1785.33	1800.20	inf	185.18$^{(3)}$	98.84	1637.67	1812.62	inf	213.99$^{(3)}$
20	5	2	3	5	99.95	943.67	725.80	95.30	142.84	99.95	980.67	706.01	97.26	223.66
20	5	2	6	5	99.06	995.33	1800.20	96.25	359.68$^{(3)}$	99.02	906.67	1800.29	96.25	409.23$^{(3)}$
20	5	2	9	5	97.35	501.67	1800.22	inf	812.04$^{(3)}$	97.43	424.33	1800.34	inf	868.06$^{(3)}$
30	1	2	3	4	–	–	–	–	–	–	–	–	–	–
30	1	2	6	4	–	–	–	–	–	–	–	–	–	–
30	1	2	9	4	–	–	–	–	–	97.35	1171.33	1812.55	inf	446.95$^{(3)}$
30	2	2	3	5	–	–	–	–	–	–	–	–	–	–
30	2	2	6	5	–	–	–	–	–	99.81	968	1688.18	inf	379.84$^{(1)}$
30	2	2	9	5	–	–	–	–	–	–	–	–	–	–
30	5	2	3	6	–	–	–	–	–	99.65	386	1800.17	inf	403.69$^{(1)}$
30	5	2	6	6	–	–	–	–	–	–	–	–	–	–
30	5	2	9	6	–	–	–	–	–	–	–	–	–	–
40	1	2	3	4	–	–	–	–	–	–	–	–	–	–
40	1	2	6	4	–	–	–	–	–	–	–	–	–	–
40	2	2	3	5	–	–	–	–	–	99.98	301	732.41	99.06	357.08
40	2	2	6	5	–	–	–	–	–	–	–	–	–	–
40	5	2	3	6	–	–	–	–	–	–	–	–	–	–
40	5	2	6	6	–	–	–	–	–	–	–	–	–	–
50	1	2	3	4	–	–	–	–	–	–	–	–	–	–
50	2	2	3	5	–	–	–	–	–	–	–	–	–	–
50	5	2	3	6	–	–	–	–	–	–	–	–	–	–

(#) Indicates number of instances (out of 3) that did not solve to optimality
– Indicates that no incumbent solution was found at the root node within the 1-hour time limit for any instance

Table 3.6: Scenario 4 Summary Without and With UB Heuristic

					Without UB Heuristic					With UB Heuristic				
n	*K*	*L*	*T*	*V*	%LB	Nodes	CPU	%RLB	RCPU	%LB	Nodes	CPU	%RLB	RCPU
10	1	2	3	3	99.96	308	7.87	92.32	2.79	99.96	88	7.90	99.08	5.71
10	1	2	6	3	99.92	747	27.35	93.42	8.70	99.94	680.33	44.54	97.68	30.16
10	1	2	9	3	99.94	946.67	42.76	94.77	10.24	99.93	867	82.38	96.48	50.03
10	2	2	3	4	99.92	1400	33.81	88.08	5.19	99.95	3446.67	57.80	92.07	10.00
10	2	2	6	4	99.91	5379	175.99	92.09	21.28	99.92	6265.33	328.56	93.76	137.04
10	2	2	9	4	99.94	3315.33	301.63	87.47	40.81	99.91	5717.33	566.06	91.82	180.73
10	5	2	3	5	99.94	2873.67	202.94	81.45	14.24	99.94	1303	102.71	82.96	16.62
10	5	2	6	5	98.85	1919	1800.19	76.02	99.33$^{(3)}$	98.87	1899	1800.19	86.81	265.76$^{(3)}$
10	5	2	9	5	98.01	1628.33	1800.20	72.16	178.02$^{(3)}$	97.61	1703.67	1800.19	76.43	252.81$^{(3)}$
20	1	2	3	3	99.94	355	32.92	98.90	16.50	99.94	173	44.89	99.30	30.92
20	1	2	6	3	99.96	587.33	116.74	96.34	61.70	99.97	1040	244.49	98.83	163.97
20	1	2	9	3	99.91	3621.33	451.25	85.75	78.80	99.92	2899	555.4	96.75	202.25
20	2	2	3	4	99.94	406.5	80.73	98.32	27.40	99.95	2991.67	345.83	97.63	164.93
20	2	2	6	4	99.91	697	331.44	88.60	107.48	99.92	1781.67	697.01	89.98	200.84
20	2	2	9	4	98.71	2313.33	1800.41	46.54	206.93$^{(3)}$	98.56	2413.33	1800.16	54.75	253.66$^{(3)}$
20	5	2	3	5	99.83	2417	1258.45	87.95	85.01$^{(1)}$	99.67	2976.33	1190.83	89.13	135.67$^{(1)}$
20	5	2	6	5	99.33	1297	1800.21	inf	245.51$^{(3)}$	99.30	1182	1800.20	inf	284.96$^{(3)}$
20	5	2	9	5	75.69	696.67	1800.20	inf	682.67$^{(3)}$	71.53	573.33	1800.48	inf	296.09$^{(3)}$
30	1	2	3	4	99.92	2934.75	489.72	94.23	70.46	99.96	2768	666.17	95.41	235.90
30	1	2	6	4	99.93	2220.67	1090.98	86.95	211.75	99.94	1681.33	1019.22	87.00	289.68
30	1	2	9	4	98.47	1442.33	1800.31	43.12	396.31$^{(3)}$	98.47	1482.33	1800.34	43.12	427.38$^{(3)}$
30	2	2	3	5	99.95	2075.67	1208.89	85.39	211.43	99.64	1749.33	1183.04	85.78	272.41$^{(1)}$
30	2	2	6	5	99.64	877.33	1501.64	47.52	513.60$^{(2)}$	99.64	978	1555.24	47.52	484.84$^{(2)}$
30	2	2	9	5	95.97	200	1801.13	41.71	1031.32$^{(2)}$	37.70	166.67	1800.39	37.55	1142.69$^{(3)}$
30	5	2	3	6	–	–	–	–	–	96.90	486.33	1560.38	inf	632.74$^{(2)}$
30	5	2	6	6	–	–	–	–	–	94.78	187	1800.20	inf	777.97$^{(1)}$
30	5	2	9	6	–	–	–	–	–	–	–	–	–	–
40	1	2	3	4	–	–	–	–	–	99.81	2046.5	1483.39	91.08	241.80$^{(1)}$
40	1	2	6	4	–	–	–	–	–	99.94	1421	1413.19	inf	375.39
40	2	2	3	5	–	–	–	–	–	99.10	970.67	1739.90	72.02	352.33$^{(2)}$
40	2	2	6	5	–	–	–	–	–	83.41	353	1800.27	inf	899.76$^{(2)}$
40	5	2	3	6	–	–	–	–	–	78.41	97	1800.18	inf	765.47$^{(2)}$
40	5	2	6	6	–	–	–	–	–	–	–	–	–	–
50	1	2	3	4	–	–	–	–	–	99.67	1600	1814.05	88.53	280.65$^{(2)}$
50	2	2	3	5	–	–	–	–	–	99.93	401.33	1531.50	61.95	498.83
50	5	2	3	6	–	–	–	–	–	–	–	–	–	–

(#) Indicates number of instances (out of 3) that did not solve to optimality
– Indicates that no incumbent solution was found at the root node within the 1-hour time limit for any instance

Table 3.7: Average Solution Time for UB Heuristic (in seconds)

					Scenario 1			Scenario 2			Scenario 3			Scenario 4		
n	K	L	T	V	R CPU	PD CPU	Total CPU	R CPU	PD CPU	Total CPU	R CPU	PD CPU	Total CPU	R CPU	PD CPU	Total CPU
10	1	2	3	3	0.56	0.38	0.93	0.56	0.26	0.81	0.59	0.28	0.86	0.54	0.39	0.93
10	1	2	6	3	1.00	0.33	1.33	1.11	0.43	1.55	1.16	0.33	1.49	0.98	0.77	1.74
10	1	2	9	3	1.42	0.62	2.04	1.63	0.55	2.18	1.58	0.49	2.07	1.37	1.03	2.40
10	2	2	3	4	0.64	0.65	1.29	0.81	0.28	1.09	0.77	0.24	1.01	0.68	0.83	1.51
10	2	2	6	4	1.41	1.04	2.44	1.67	0.66	2.33	1.52	0.39	1.90	1.35	1.62	2.97
10	2	2	9	4	1.97	2.47	4.44	2.47	1.32	3.79	2.50	0.81	3.31	2.01	2.62	4.63
10	5	2	3	5	0.83	2.17	3.01	0.90	0.54	1.44	1.10	0.57	1.67	0.81	1.67	2.47
10	5	2	6	5	1.47	7.70	9.18	1.92	1.68	3.59	2.05	1.21	3.25	1.80	9.34	11.14
10	5	2	9	5	2.22	13.23	15.45	2.63	2.36	4.98	2.8	1.67	4.47	2.29	20.19	22.47
20	1	2	3	3	0.55	0.45	0.99	1.01	0.25	1.26	1.03	0.27	1.30	0.54	0.60	1.15
20	1	2	6	3	0.98	0.53	1.51	2.41	0.47	2.89	2.34	0.44	2.78	1.08	0.94	2.02
20	1	2	9	3	1.54	0.88	2.42	3.55	0.73	4.28	3.50	0.67	4.17	1.57	1.38	2.95
20	2	2	3	4	0.71	0.83	1.54	1.61	0.41	2.02	1.56	0.28	1.84	0.68	1.29	1.97
20	2	2	6	4	1.44	1.77	3.21	3.43	1.06	4.48	4.06	0.88	4.94	1.47	2.42	3.89
20	2	2	9	4	2.30	6.01	8.31	5.73	2.28	8.01	5.12	1.42	6.54	2.38	6.27	8.64
20	5	2	3	5	0.96	4.65	5.61	1.84	1.08	2.92	1.69	1.13	2.82	0.83	4.61	5.45
20	5	2	6	5	1.57	9.51	11.07	4.06	2.29	6.35	3.18	1.89	5.07	1.62	16.80	18.42
20	5	2	9	5	2.41	41.96	44.37	4.82	4.91	9.73	6.18	4.09	10.27	2.63	58.35	60.98
30	1	2	3	4	0.73	0.90	1.63	1.92	0.30	2.22	2.00	0.32	2.32	0.70	0.83	1.54
30	1	2	6	4	1.43	2.03	3.46	3.92	0.61	4.53	3.84	0.60	4.44	1.46	2.18	3.65
30	1	2	9	4	2.17	3.77	5.94	5.87	1.39	7.26	4.60	1.33	5.93	2.24	5.93	8.17
30	2	2	3	5	0.96	2.59	3.55	2.41	0.66	3.06	2.39	0.45	2.84	0.92	2.39	3.31
30	2	2	6	5	1.67	2.09	3.76	4.85	1.39	6.24	4.97	1.00	5.97	1.81	4.77	6.58
30	2	2	9	5	2.61	13.42	16.03	7.13	2.92	10.05	5.80	2.29	8.09	2.58	38.70	41.28
30	5	2	3	6	1.11	18.23	19.35	2.42	1.39	3.81	3.02	1.01	4.03	1.11	22.73	23.84
30	5	2	6	6	2.17	69.50	71.67	5.90	3.40	9.29	5.92	3.32	9.24	2.19	62.88	65.07
30	5	2	9	6	3.34	190.78	194.12	8.72	8.93	17.65	8.68	7.47	16.15	3.00	203.54	206.54
40	1	2	3	4	0.94	1.40	2.34	4.67	0.35	5.02	4.71	0.37	5.08	0.99	0.94	1.93
40	1	2	6	4	1.72	2.08	3.80	9.20	0.79	9.99	9.28	0.74	10.03	1.80	1.59	3.39
40	2	2	3	5	1.02	4.78	5.80	5.76	0.64	6.40	4.29	0.69	4.98	1.07	3.77	4.84
40	2	2	6	5	1.92	11.74	13.66	11.84	1.63	13.47	11.48	1.38	12.86	1.92	7.98	9.91
40	5	2	3	6	1.12	54.98	56.10	7.05	2.08	9.12	6.86	1.95	8.81	1.14	67.89	69.03
40	5	2	6	6	2.15	150.31	152.45	13.86	6.47	20.33	13.65	6.37	20.02	2.25	164.72	166.97
50	1	2	3	4	1.20	1.28	2.47	10.81	0.44	11.24	10.82	0.39	11.20	1.05	1.26	2.31
50	2	2	3	5	1.07	6.11	7.18	13.64	0.91	14.54	13.52	0.77	14.28	1.25	4.31	5.56
50	5	2	3	6	1.29	66.21	67.51	17.87	2.57	20.44	18.11	2.11	20.22	1.33	66.90	68.23

Table 3.8: Cost and CPU Breakdown for Different Values of α^v

	Cost Proportions (%)							CPU Breakdown (seconds)			
α^v	Prod.	Inv.	Back.	Item	Family	Trans.	Avg Vehicle Util (%)	Cluster	TSP	PD	Total
all 0.0	86.64	4.60	1.20	1.25	6.15	0.16	80.44	1.49	1.13	6.39	8.92
all 0.1	86.6	4.51	1.25	1.26	6.22	0.16	80.05	1.54	1.13	5.83	8.42
all 0.2	86.66	4.61	1.19	1.24	6.14	0.16	80.35	1.65	1.13	6.62	9.28
all 0.3	86.76	4.59	1.08	1.25	6.16	0.16	80.47	1.44	1.19	5.86	8.39
all 0.4	86.76	4.60	1.12	1.24	6.12	0.16	80.41	1.60	1.15	7.32	9.95
all 0.5	86.72	4.59	1.17	1.24	6.11	0.16	80.74	1.67	1.16	5.65	8.33
all 0.6	86.77	4.56	1.15	1.24	6.12	0.17	81.1	1.71	1.11	5.60	8.11
all 0.7	86.95	4.51	0.93	1.25	6.19	0.17	81.43	2.58	1.12	5.45	8.82
[0.7, 0.3, 0.0]	86.76	4.48	1.13	1.26	6.20	0.17	82.35	2.03	1.12	5.71	8.54
[0.0, 0.3, 0.7]	86.79	4.59	1.05	1.25	6.16	0.16	79.87	1.52	1.13	6.36	8.92

3.8.3 Impact of Problem Settings on Costs and Vehicle Utilization

Aside from computational performance, cost breakdown for each scenario is impacted by the rigidity expressed through inventory and routing constraints. We can now examine results from the heuristic approach of Scenario 5 alongside those exact solutions obtained from Scenarios 2 and 4 (backorders are allowed), as well as observe their differences compared to Scenarios 1 and 3 (backorders not allowed).

Setting Minimum Vehicle Thresholds for Scenario 5

For Scenario 5, we ran a number of experiments varying α^v, the minimum threshold capacity for each vehicle, between 0.0 and 0.7 in increments of 0.1, with results summarized in Table 3.8. Overall average cost ratios are shown, detailing the proportion of total costs attributed to production, inventory holding, backorder, item setup, family setup, and transportation across all instances. Average vehicle utilization and a breakdown of computation time is also presented.

When α^v is set to 0.7 for all values of v, this provides high average vehicle utilization across all instances. When α^v is varied across different vehicle sizes, we see that vehicle utilization improves when smaller vehicles are filled first ($\alpha^S = 0.7$), though backorder costs now increase in proportion to other costs. When larger vehicles are encouraged to be filled first, average vehicle utilization and the backorder cost ratio both deteriorate. Moving forward, we use only the results with all values of $\alpha^v = 0.7$ when discussing Scenario 5's performance in relation to Scenarios 2 and 4.

Comparing Proportion of Costs

Figure 3.3 shows the average proportion of costs for all scenarios. For Scenarios 1 and 3, the emphasis is on ensuring customer demand is met on time. The majority of costs in each of these scenarios is focused on production (roughly 7% of total costs) and inventory holding (over 90%). Family and item setups account for just under 1% of total cost, while transportation costs contribute under 0.01% in both scenarios. Scenario 3 experiences slightly lower transportation costs, with split deliveries allowing for more streamlined routes.

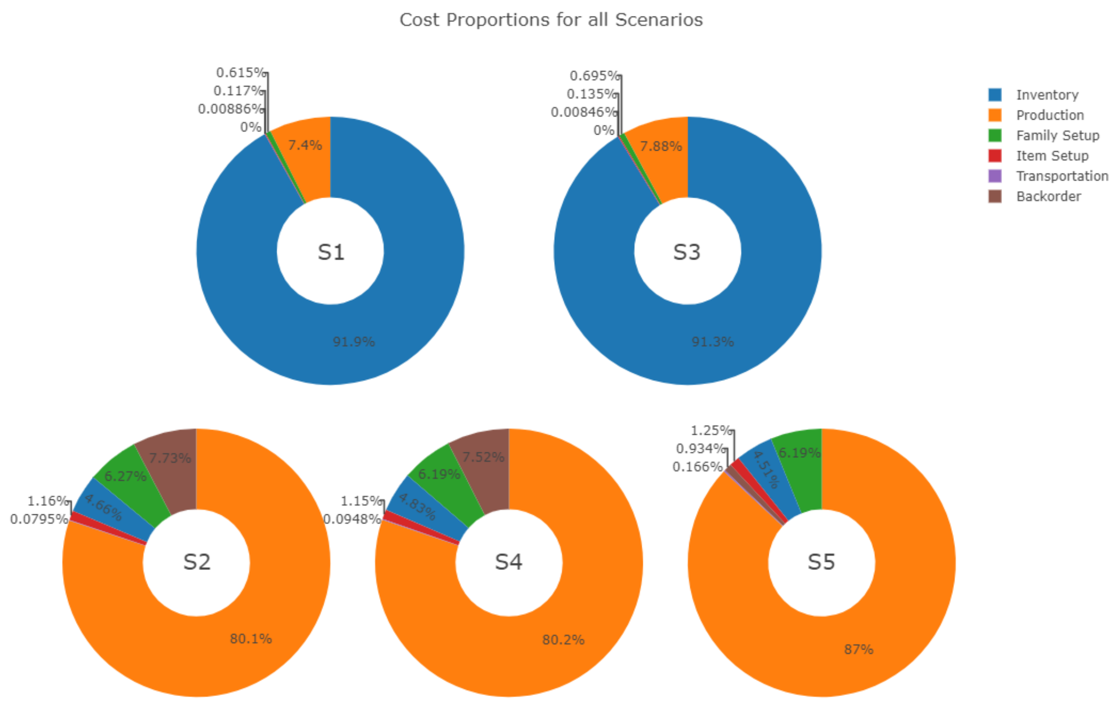

Figure 3.3: Cost Proportions for all Scenarios

It is difficult to compare total costs across scenarios, as certain instances for each scenario may not have been solved due to computational issues, and this impacts the calculated total average costs. What can be confirmed is that, for the same instances that have been solved, Scenario 3 always results in lower total costs, but only by roughly 0.01% (Table 3.9).

When backorders are allowed in Scenarios 2, 4, and 5, the proportion of total cost differs from that of Scenarios 1 and 3. While production still dominates, with an average of 80% for Scenarios 2 and 4 and 86% for Scenario 5, now inventory holding cost proportions reduce dramatically to between 4.5% and 5%. Demand at times is backordered, contributing between 7.5 – 7.7% towards total costs in Scenarios 2 and 4, but just under 1% for Scenario 5 (due to increased production). Family setup costs now account for over 6% of total costs in all cases, and transportation costs rise to make up just under 0.1% of all costs for Scenarios 2 and 4, but 0.17% for Scenario 5.

While Scenario 5 seems better at meeting customer demands, recall that both production and transportation costs are higher compared to Scenarios 2 and 4. When examining total cost for individual instances, it is clear that Scenario 5 does indeed cost more overall in the majority of instances, but not by much (Table 3.10).

Comparing Vehicle Utilization

From Figure 3.3, it is apparent that allowing backorders has a bigger impact on transportation costs than do split deliveries. Changes in overall average vehicle utilization are not as significant. Vehicle utilization for Scenario 1 (78.5%) is better than in Scenario 3 (75.5%), but transportation cost ratio is just slightly higher (Table 3.9). When omitting any problem sizes that did not solve for both Scenarios 1 and 3, Scenario 1's overall average vehicle utilization drops to 75.9%. The heuristic approach of Scenario 5 boasts average utilization of 81.4%, compared to 74.9% and 76.5% for Scenarios 2 and 4 (Table 3.10). The poor performance of Scenarios 2 and 4 is due to many instances not solving to optimality (or at all) as problem size increased; Scenario 5 was able to provide a heuristic solution for all instances.

Table 3.9: Scenarios 1 and 3 Vehicle Utilization, CPU and % Change in Total Costs

					Avg Veh Util. (%)		Total CPU (seconds)		S3 % Change from S1 Total Cost
n	*K*	*L*	*T*	*V*	S1	S3	S1	S3	
10	1	2	3	3	71.30	72.10	5.40	9.04	-0.001
10	1	2	6	3	72.04	72.04	46.19	64.21	-0.001
10	1	2	9	3	72.48	71.33	175.50	100.21	-0.006
10	2	2	3	4	75.86	75.12	17.16	7.42	-0.002
10	2	2	6	4	76.66	76.21	190.36	81.53	-0.004
10	2	2	9	4	76.93	76.80	606.89	216.79	-0.004
10	5	2	3	5	70.03	69.00	147.67	51.82	-0.002
10	5	2	6	5	70.79	69.82	1661.94	981.60	-0.014
10	5	2	9	5	72.19	70.40	1800.18	1800.19	-0.010
20	1	2	3	3	71.57	72.51	55.31	41.17	-0.553
20	1	2	6	3	73.03	72.07	221.32	157.11	-0.003
20	1	2	9	3	75.88	75.23	658.22	326.98	0.001
20	2	2	3	4	76.57	76.65	147.77	135.32	3.629(a)
20	2	2	6	4	76.45	76.63	514.08	298.14	-0.005
20	2	2	9	4	80.39	79.86	1800.32	1812.62	-0.006
20	5	2	3	5	71.54	73.91	637.82	706.01	-0.001
20	5	2	6	5	72.93	71.03	1800.23	1800.29	-0.072
20	5	2	9	5	75.28	74.96	1800.20	1800.34	-1.770(b)
30	1	2	3	4	87.44	–	592.90	–	–
30	1	2	6	4	88.13	–	1098.92	–	–
30	1	2	9	4	92.22	92.28	1800.38	1812.55	-0.039
30	2	2	3	5	82.12	–	990.79	–	–
30	2	2	6	5	83.86	83.66	1733.97	1688.18	0
30	2	2	9	5	88.39	–	1800.95	–	–
30	5	2	3	6	78.85	76.95	1800.17	1800.17	6.076(a)
30	5	2	6	6	82.82	–	1801.21	–	–
30	5	2	9	6	–	–	–	–	–
40	1	2	3	4	79.31	–	1210.42	–	–
40	1	2	6	4	81.11	–	1577.36	–	–
40	2	2	3	5	82.32	82.24	1295.74	732.41	-2.552(b)
40	2	2	6	5	82.11	–	1806.61	–	–
40	5	2	3	6	–	–	–	–	–
40	5	2	6	6	–	–	–	–	–
50	1	2	3	4	85.31	–	1517.47	–	–
50	2	2	3	5	86.67	–	1574.73	–	–
50	5	2	3	6	–	–	–	–	–
		Overall Average			78.52	75.49	1027.76	746.55	

(a) Indicates fewer instances for Scenario 3 solved than for Scenario 1
(b) Indicates fewer instances for Scenario 1 solved than for Scenario 3
– Indicates no data available

Table 3.10: Scenarios 2, 4, and 5 Vehicle Utilization, CPU and % Change in Total Costs

					Avg. Veh. Util. (%)			Total CPU (seconds)			% Change from S5 Total Cost	
n	*K*	*L*	*T*	*V*	S2	S4	S5	S2	S4	S5	S2	S4
10	1	2	3	3	70.64	71.38	74.07	4.41	7.90	3.29	-0.18	-0.15
10	1	2	6	3	72.35	71.36	73.51	19.79	44.54	3.24	0.00	0.02
10	1	2	9	3	70.35	70.64	73.43	98.68	82.38	3.51	0.03	0.07
10	2	2	3	4	75.83	75.96	77.97	7.30	57.80	4.04	-0.21	-0.20
10	2	2	6	4	76.70	76.61	78.50	71.02	328.56	4.79	-0.03	0.02
10	2	2	9	4	76.54	77.29	78.73	220.74	566.06	7.46	-0.07	-0.02
10	5	2	3	5	68.84	69.76	72.58	68.52	102.71	2.82	-0.16	-0.12
10	5	2	6	5	69.76	70.58	72.97	1728.93	1800.19	7.66	-0.21	-0.04
10	5	2	9	5	71.19	71.23	74.12	1800.18	1800.19	21.25	0.27	0.43
20	1	2	3	3	70.12	78.94	72.32	54.18	44.89	4.24	-0.17	-0.16
20	1	2	6	3	73.33	71.91	74.27	170.04	244.49	3.98	-0.01	0.01
20	1	2	9	3	76.03	75.22	76.24	439.92	555.40	6.16	0.00	0.05
20	2	2	3	4	75.93	76.61	78.06	83.76	345.83	4.15	-0.21	-0.18
20	2	2	6	4	76.54	76.61	77.12	310.41	697.01	7.07	-0.15	-0.10
20	2	2	9	4	79.62	80.11	80.34	1842.33	1800.16	11.72	0.10	-0.06
20	5	2	3	5	71.39	71.61	73.66	1131.15	1190.83	8.82	-0.50	-0.40
20	5	2	6	5	72.00	71.75	73.96	1800.27	1800.20	18.97	0.16	-0.02
20	5	2	9	5	71.48	57.50	77.42	1800.42	1800.48	9.24	58.00	26.97
30	1	2	3	4	86.71	85.92	85.64	241.87	666.17	3.88	-0.58	-0.60
30	1	2	6	4	87.49	87.63	87.7	897.94	1019.22	5.02	-0.38	-0.35
30	1	2	9	4	91.59	91.82	91.67	1801.10	1800.34	15.39	-0.43	-0.36
30	2	2	3	5	82.59	83.41	83.05	832.09	1183.04	5.30	-0.35	-0.32
30	2	2	6	5	82.08	82.55	83.69	1435.20	1555.24	11.73	-0.17	-0.04
30	2	2	9	5	75.76	52.67	87.58	1800.22	1800.39	25.83	-9.58	61.26
30	5	2	3	6	80.08	79.48	80.89	1800.24	1560.38	26.48	-0.19	2.41
30	5	2	6	6	–	78.72	82.22	–	1800.20	7.34	–	3.86
30	5	2	9	6	–	–	84.85	–	–	10.53	–	–
40	1	2	3	4	75.88	76.48	85.77	738.06	1483.39	2.87	-0.48	-0.93
40	1	2	6	4	75.05	80.62	85.72	1288.35	1413.19	6.44	-0.06	-0.55
40	2	2	3	5	81.53	82.02	87.62	1352.52	1739.90	6.33	-0.92	-0.39
40	2	2	6	5	79.12	75.35	88.59	1800.42	1800.27	15.96	-1.16	15.59
40	5	2	3	6	42.49	89.04	92.21	1800.19	1800.18	6.28	257.27	29.64
40	5	2	6	6	–	–	93.17	–	–	8.91	–	–
50	1	2	3	4	80.31	77.99	89.39	1800.27	1814.05	5.24	-1.09	-0.9
50	2	2	3	5	85.04	86.19	90.42	1718.12	1531.50	6.99	-0.43	-0.38
50	5	2	3	6	47.26	–	91.96	1800.32	–	14.55	182.95	–
		Overall Average			74.90	76.51	81.43	992.70	1098.09	8.82		

– Indicates no data available

Comparing Computational Performance

As expected, Scenario 5 provides good solutions very quickly when compared to the exact approaches of Scenarios 1 through 4, with overall average CPU time at 8.82 seconds. The restrictions imposed in Scenario 1 make larger problems very difficult to solve, as evidenced by its average of 1072.76 seconds. What is surprising is that Scenario 4, with backorders allowed, appears to require more computational effort than Scenario 1. This discrepancy may be due to differences in solvable instances by each scenario. Scenarios 2 and 3 solve more quickly than Scenarios 1 and 4, suggesting that allowing for split deliveries reduces the time needed to allocate demand and route vehicles in each period.

3.9 Remarks

For Scenarios 1 through 4, the upper bounding heuristic improves the root node solution at a slight increase in computational time when compared to solving problems directly with the branch-and-cut algorithm. Computing issues became troublesome as instances grew in size, resulting in many unsolvable problems when the upper bounding heuristic was not present. Using the upper bounding heuristic is indeed beneficial, even if just solving to the root node, as solution quality is quite good and computational time is within reason.

The structure of subtour elimination constraints has an impact on solution quality and time, as seen in Figure 3.1. Further experimentation was carried out using SECs in the form of (3.11) for Scenario 2, when backorders and split deliveries are allowed. The gap (overall and root node), number of nodes explored, and solution time (overall and root node) are compared when the branch-and-cut algorithm is applied on its own and with warm-starting (Table B.1 in Appendix B). The advantages of the upper bounding heuristic in reducing the overall duality gap are more pronounced there than in Table 3.4, due to the poor solution quality provided by the weak form of SECs (3.11).

Our heuristic approach in Scenario 5 divides the production-routing-problem into clustering, routing, and production-distribution phases, and produces total costs that are close to those of Scenarios 2 and 4. At the current problem settings, a reduction in backorder cost is offset by increased production to meet customer demands on time along with an increase in transportation cost. Vehicle utilization is much higher than in the exact approaches, and computational time is significantly reduced.

The problem settings expressed by these five scenarios can be useful to managers when deciding upon both customer service and routing policies. It is difficult to compare computational effort across all scenarios; when considering only the common instances that did solve, there are few differences, but allowing more flexibility in inventory and routing does improve overall CPU time. Total costs, on the other hand, are not impacted as much. Scenario 5 seems to provide the best holistic solution when management's focus is on customer service and maintaining regularity in driver schedules, with the highest vehicle utilization amongst all scenarios at just slightly greater costs.

Chapter 4

Conclusion

We examine formulations for a multi-family capacitated coordinated lot-sizing problem, along with its incorporation in a production-routing problem. This is a model that has not seen much attention in the literature since the work by Erenguc and Mercan (1990) and Mercan and Erenguc (1993). To the best of our knowledge, we are the first to solve this problem using strong reformulations.

In Chapter 2, we propose a hybrid Benders-Evolutionary Algorithm approach, combined with subgradient optimization, to find improved upper and lower bounds for the multi-family CCLSP with setup times. The combination of these methods improves upon the quality of the lower bounds found solely by subgradient-optimization-based methods. Our heuristic produces better bounds than those found by Süral et al. (2009), reducing the average duality gap in all 60 instances.

The nature of this method is such that the Benders master problem becomes increasingly difficult as more cuts are added. However, the evolutionary algorithm is able to inject some speed into the heuristic by quickly generating a number of feasible solutions at certain intervals of the process. The subgradient technique helps to close the gap once a good upper bound has been obtained, and adds little to the overall solution time of this method.

As noted by Süral et al. (2009), the CCLSP with setup times and without setup costs is more difficult to solve than its counterpart where setup costs are present. So, while our proposed heuristic performs well when setup costs are present, there is room for improvement in the special case when setup costs are set to zero. In terms of solution time, we do not outperform Süral et al. (2009) or de Araujo et al. (2015), but this should not deter the user from applying our technique:

as the multi-family CCLSP is a tactical planning problem, there is sufficient time available to find good quality solutions. Given that the typical single-family CCLSP duality gaps reported in literature range between 7% and over 100% (Süral et al., 2009; de Araujo et al., 2015), we can relax certain tolerances of our heuristic to speed up solution time, with the understanding that this will reduce solution quality.

In addition to strengthening both the lower and upper bounds, we contribute a standard test bed for the MF-CCLSP that consists of both small and large problems of varying difficulty. While in this work we study a problem without setup costs, that cost information (from the Trigeiro et al. (1989) problem sets) is included in our data files so that problem extensions can be examined in the future.

Chapter 3 incorporates coordinated, capacitated lot-sizing for multiple product groups within the production-routing problem served by a heterogeneous fleet of vehicles. We consider five scenarios with varying degrees of flexibility in demand and transportation to study the impact of backorders and split deliveries on vehicle utilization and total cost. Most research considering multiple items and large customer networks rely on metaheuristics, but we employ an exact branch-and-cut algorithm to solve four of our five scenarios. Coupled with our upper bounding heuristic, the branch-and-cut heuristic develops good quality solutions at the root node in very little time for smaller problem instances (average %RLB over Scenarios 1 through 4 is 96.1% *with* UB heuristic, versus 90.5% *without*).

It is interesting to note that most solution approaches, whether heuristic or exact, tend to tackle the production portion of the problem first. We do this when solving $[PD]$ before $[R]$ in our upper bounding heuristic, as do all authors listed in Table 1.1 *except* Solyalı and Süral (2017). Instead, their multi-phase heuristic begins by solving a traveling salesman problem over the entire network to generate a suitable order in which to visit customers. In many practical applications, routes are more commonly considered to be a tactical decision, revised only when needed (Kalcsics, 2015; Bender et al., 2016). It is thus logical to fix these customer routes first, and then determine the optimal production and distribution schedule subject to those routing constraints.

Our fifth scenario, solved by the decomposition heuristic, does just that. We cluster customers first thereby creating dedicated routes that remain unchanged over the planning horizon.

Solution quality is comparable to results from Scenarios 2 and 4, with average computational time running under 9 seconds. Scenario 5 is an attractive approach, when priority is given to replenishing customer demand on time, and to offering stable delivery schedules. Though costs are just slightly higher due to increased production and transportation costs, Scenario 5 achieves an average of 81% vehicle utilization with fewer backorders than Scenarios 2 and 4.

4.1 Future Work

4.1.1 Coordinated Capacitated Lot-Sizing

Possible avenues for further research include improving the hybridization of the evolutionary algorithm with Benders decomposition, specifically, the method for solving the master problem. We sequentially apply an exact approach (Benders decomposition) and a metaheuristic (evolutionary algorithm), though a more intertwined method, such as a *matheuristic* (Bollapragada et al., 2011; Camargo et al., 2014) may be more appropriate. As presented in Poojari and Beasley (2009) and discussed in Rahmaniani et al. (2017), the master problem can be solved using an EA rather than by optimization. The solutions developed may or may not be feasible, so the EA parameters would need to be adjusted to ensure that at least some feasible solutions can be passed to the subproblem. Modifying evolutionary operators, such as mutation and crossover, can also impact how solutions are developed. Furthermore, the master problem could be solved to some tolerance, ε, where ε decreases as more Benders cuts are added. This ensures that not much time is spent solving the master problem to optimality at early stages, when one is further from the best overall solution.

Regarding more exact methods, incorporating branch-and-bound methods, as Gendron et al. (2016) do for their two-level uncapacitated facility location problem, could improve lower bounds. However, to exploit their technique, instead of relaxing the demand constraints, the capacity constraints (2.14) would be relaxed to separate the formulation into n uncapacitated facility location problems. While this has been discussed by Jans and Degraeve (2004) and Süral et al. (2009) as yielding inferior LP bounds in the single-family CCLSP, no testing has been done in the multiple family case.

Aside from improving these solution approaches, there are pure lot-sizing and combined applications where natural problem extensions to the multi-family setting may prove interesting. From a lot-sizing perspective, those features could include multi-level lot-sizing, rolling horizons (Goren et al., 2010), setup crossover (Goren et al., 2012), time windows (Darvish et al., 2016), or setup carryover (Fiorotto et al., 2014). The production-routing problem is also a promising avenue where heuristic approaches for a multi-item setting have gained traction (Kang et al., 2017) but exact methods are rare (Adulyasak et al., 2015).

4.1.2 Production Routing

As discussed in Section 1.2, we do not use a strong reformulation of the CCLSP in this work, due to the findings of Adulyasak et al. (2014a). However, their computational experiments (and much of the previous research) examined the *single-item* PRP. Brahimi and Aouam (2016) do employ a facility location reformulation in their multi-item PRP, but under a non-coordinated replenishment scheme. Future work could compare the impact of applying this strong reformulation for the CCLSP when utilizing an exact solution approach.

How customers are clustered within a production-routing setting is also an area that deserves more attention. While Konur and Geunes (2016) examine a strategic setting where annual demand dictates the economic order quantity to be delivered each period, their formulation may not be amenable to situations when demand is not constant over the planning horizon. The work by Nananukul (2013) addresses this specifically. They develop route clusters based on various customer characteristics, such as demand or geographic proximity, although they solve a single-item problem with tabu search. Our Scenario 5 addresses clustering under a multi-item setting, using k-means++ to seed clusters based on demand over the horizon. There is an opportunity here to explore *exact* approaches for the tactical PRP with multiple items, where customer clusters are determined first, as these decisions will strongly influence the remaining production and distribution decisions.

Buzacott (2013) stresses that production research should aim to “(i) [understand] the issues that managers must address, (ii) [develop] models or theories that provide insight into these issues, and (iii) [apply] the result of the research to improve real-world production.” In our work, we address points (i) and (ii) by extending the CCLSP and PRP to consider multiple families, focusing on exact methods and decomposition heuristics to improve both upper and lower bounds. We are able to show that under a number of different parameter settings, we achieve good production and production-routing plans in a reasonable amount of time. Though we have not yet addressed the optimal operational settings for such problems in practice, we are primed to test our approach on real production data, in the hopes of integrating our lot-sizing and production-routing techniques into models larger in scope.

References

Aardal, K. and T. Larsson (1990). A Benders decomposition based heuristic for the hierarchical production planning problem. *European Journal of Operational Research 45*(1), 4–14.

Absi, N., C. Archetti, S. Dauzère-Pérès, and D. Feillet (2015). A two-phase iterative heuristic approach for the production routing problem. *Transportation Science 49*(4), 784–795.

Adulyasak, Y. (2017). *Instances and Results*. https://sites.google.com/site/ayossiri/publications, accessed March 13, 2017.

Adulyasak, Y., J.-F. Cordeau, and R. Jans (2014a). Formulations and branch-and-cut algorithms for multivehicle production and inventory routing problems. *INFORMS Journal on Computing 26*(1), 103–120.

Adulyasak, Y., J.-F. Cordeau, and R. Jans (2014b). Optimization-based adaptive large neighborhood search for the production routing problem. *Transportation Science 48*(1), 20–45.

Adulyasak, Y., J.-F. Cordeau, and R. Jans (2015). The production routing problem: A review of formulations and solution algorithms. *Computers & Operations Research 55*, 141–152.

Archetti, C., L. Bertazzi, G. Laporte, and M. G. Speranza (2007). A branch-and-cut algorithm for a vendor-managed inventory-routing problem. *Transportation Science 41*(3), 382–391.

Archetti, C., L. Bertazzi, G. Paletta, and M. G. Speranza (2011). Analysis of the maximum level policy in a production-distribution system. *Computers & Operations Research 38*(12), 1731–1746.

Armentano, V. A., A. Shiguemoto, and A. Løkketangen (2011). Tabu search with path relinking for an integrated production–distribution problem. *Computers & Operations Research 38*(8), 1199–1209.

Arthur, D. and S. Vassilvitskii (2007). k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pp. 1027–1035. Society for Industrial and Applied Mathematics.

Bahl, H. C. and S. Zionts (1987). Multi-item scheduling by Benders' decomposition. *Journal of the Operational Research Society 38*(12), 1141–1148.

Bard, J. F. and N. Nananukul (2009a). Heuristics for a multiperiod inventory routing problem with production decisions. *Computers & Industrial Engineering 57*(3), 713–723.

Bard, J. F. and N. Nananukul (2009b). The integrated production–inventory–distribution–routing problem. *Journal of Scheduling 12*(3), 257–280.

Bard, J. F. and N. Nananukul (2010). A branch-and-price algorithm for an integrated production and inventory routing problem. *Computers & Operations Research 37*(12), 2202–2217.

Barnhart, C., E. L. Johnson, G. L. Nemhauser, M. W. Savelsbergh, and P. H. Vance (1998). Branch-and-price: Column generation for solving huge integer programs. *Operations Research 46*(3), 316–329.

Bayley, T., H. Süral, and J. H. Bookbinder (2017). A hybrid Benders approach for coordinated capacitated lot-sizing of multiple product families with setup times. *International Journal of Production Research 56*(3), 1326–1344.

Bender, M., A. Meyer, J. Kalcsics, and S. Nickel (2016). The multi-period service territory design problem – an introduction, a model and a heuristic approach. *Transportation Research Part E: Logistics and Transportation Review 96*, 135–157.

Benders, J. F. (1962). Partitioning procedures for solving mixed-variables programming problems. *Numerische mathematik 4*(1), 238–252.

Boctor, F. F., G. Laporte, and J. Renaud (2004). Models and algorithms for the dynamic-demand joint replenishment problem. *International Journal of Production Research 42*(13), 2667–2678.

Bollapragada, R., F. Della Croce, and M. Ghirardi (2011). Discrete-time, economic lot scheduling problem on multiple, non-identical production lines. *European Journal of Operational Research 215*(1), 89–96.

Boudia, M., M. A. O. Louly, and C. Prins (2005). Combined optimization of production and distribution. In *CD-ROM proceedings of the International Conference on Industrial Engineering and Systems Management (IESM '05)*.

Boudia, M., M. A. O. Louly, and C. Prins (2007). A reactive GRASP and path relinking for a combined production–distribution problem. *Computers & Operations Research 34*(11), 3402–3419.

Boudia, M., M. A. O. Louly, and C. Prins (2008). Fast heuristics for a combined production planning and vehicle routing problem. *Production Planning and Control 19*(2), 85–96.

Boudia, M. and C. Prins (2009). A memetic algorithm with dynamic population management for an integrated production–distribution problem. *European Journal of Operational Research 195*(3), 703–715.

Braekers, K., K. Ramaekers, and I. Van Nieuwenhuyse (2016). The vehicle routing problem: State of the art classification and review. *Computers & Industrial Engineering 99*, 300–313.

Brahimi, N. and T. Aouam (2016). Multi-item production routing problem with backordering: a milp approach. *International Journal of Production Research 54*(4), 1076–1093.

Buschkühl, L., F. Sahling, S. Helber, and H. Tempelmeier (2010). Dynamic capacitated lot-sizing problems: a classification and review of solution approaches. *OR Spectrum 32*(2), 231–261.

Buzacott, J. A. (2013). Then and now–50 years of production research. *International Journal of Production Research 51*(23-24), 6756–6768.

Camargo, V. C., F. M. Toledo, and B. Almada-Lobo (2014). HOPS – Hamming-oriented partition search for production planning in the spinning industry. *European Journal of Operational Research 234*(1), 266–277.

Chandra, P. and M. L. Fisher (1994). Coordination of production and distribution planning. *European Journal of Operational Research 72*(3), 503–517.

Chitsaz, M., J.-F. Cordeau, and R. Jans (2017). A unified decomposition matheuristic for assembly, production and inventory routing. *GERAD Technical Report G-2017-03, HEC Montréal Canada*.

Darvish, M., H. Larrain, and L. C. Coelho (2016). A dynamic multi-plant lot-sizing and distribution problem. *International Journal of Production Research 54*(22), 6707–6717.

de Araujo, S., B. De Reyck, Z. Degraeve, I. Fragkos, and R. Jans (2015). Period decompositions for the capacity constrained lot size problem with setup times. *INFORMS Journal on Computing 27*(3), 431–488.

Denizel, M., F. T. Altekin, H. Süral, and H. Stadtler (2008). Equivalence of the LP relaxations of two strong formulations for the capacitated lot-sizing problem with setup times. *OR Spectrum 30*(4), 773–785.

Dixon, P. S. and E. A. Silver (1981). A heuristic solution procedure for the multi-item, single-level, limited capacity, lot-sizing problem. *Journal of Operations Management 2*(1), 23–39.

Erenguc, S. S. and H. M. Mercan (1990). A multifamily dynamic lot-sizing model with coordinated replenishments. *Naval Research Logistics 37*(4), 539–558.

Fiorotto, D. J., R. Jans, and S. A. de Araujo (November 2014). An analysis of formulations for the capacitated lot sizing problem with setup crossover. Working Paper: CIRRELT-2014-57, Montreal, QC.

Fisher, M. L. (2004). The Lagrangian relaxation method for solving integer programming problems. *Management Science 50*(12_supplement), 1861–1871.

Floyd, R. W. (1962). Algorithm 97: shortest path. *Communications of the ACM 5*(6), 345.

Fumero, F. and C. Vercellis (1999). Synchronized development of production, inventory, and distribution schedules. *Transportation Science 33*(3), 330–340.

Gao, L. L., N. Altay, and E. P. Robinson (2008). A comparative study of modeling and solution approaches for the coordinated lot-size problem with dynamic demand. *Mathematical and Computer Modelling 47*(11), 1254–1263.

Gendreau, M. and J.-Y. Potvin (2010). Tabu search. In M. Gendreau and J.-Y. Potvin (Eds.), *Handbook of Metaheuristics*, pp. 243–263. Springer.

Gendron, B., P.-V. Khuong, and F. Semet (2016). A Lagrangian-based branch-and-bound algorithm for the two-level uncapacitated facility location problem with single-assignment constraints. *Transportation Science 50*(4), 1286–1299.

Goren, H. G., S. Tunali, and R. Jans (2010). A review of applications of genetic algorithms in lot sizing. *Journal of Intelligent Manufacturing 21*(4), 575–590.

Goren, H. G., S. Tunali, and R. Jans (2012). A hybrid approach for the capacitated lot sizing problem with setup carryover. *International Journal of Production Research 50*(6), 1582–1597.

Hansen, P., N. Mladenović, and J. A. M. Pérez (2010). Variable neighbourhood search: methods and applications. *Annals of Operations Research 175*(1), 367–407.

Hindi, K., K. Fleszar, and C. Charalambous (2003). An effective heuristic for the CLSP with set-up times. *Journal of the Operational Research Society 54*(5), 490–498.

Jans, R. and Z. Degraeve (2004). Improved lower bounds for the capacitated lot sizing problem with setup times. *Operations Research Letters 32*(2), 185–195.

Jans, R. and Z. Degraeve (2007). Meta-heuristics for dynamic lot sizing: A review and comparison of solution approaches. *European Journal of Operational Research 177*(3), 1855–1875.

Joneja, D. (1990). The joint replenishment problem: new heuristics and worst case performance bounds. *Operations Research 38*(4), 711–723.

Kalcsics, J. (2015). Districting problems. In G. Laporte, S. Nickel, and F. Saldanha da Gama (Eds.), *Location Science*, pp. 595–622. Springer.

Kang, H.-Y., A. H. Lee, C.-W. Wu, and C.-H. Lee (2017). An efficient method for dynamic-demand joint replenishment problem with multiple suppliers and multiple vehicles. *International Journal of Production Research 55*(4), 1065–1084.

Kao, E. P. (1979). A multi-product dynamic lot-size model with individual and joint set-up costs. *Operations Research 27*(2), 279–289.

Karalli, S. M. and A. D. Flowers (2006). The multiple-family ELSP with safety stocks. *Operations Research 54*(3), 523–531.

Kirca, Ö. (1995). A primal-dual algorithm for the dynamic lotsizing problem with joint set-up costs. *Naval Research Logistics 42*(5), 791–806.

Koç, Ç., T. Bektaş, O. Jabali, and G. Laporte (2016). Thirty years of heterogeneous vehicle routing. *European Journal of Operational Research 249*(1), 1–21.

Konur, D. and J. Geunes (2016). Integrated districting, fleet composition, and inventory planning for a multi-retailer distribution system. *Annals of Operations Research*, 1–33.

Lei, L., S. Liu, A. Ruszczynski, and S. Park (2006). On the integrated production, inventory, and distribution routing problem. *IIE Transactions 38*(11), 955–970.

Li, Y., Y. Tao, and F. Wang (2012). An effective approach to multi-item capacitated dynamic lot-sizing problems. *International Journal of Production Research 50*(19), 5348–5362.

Maes, J., J. O. McClain, and L. N. Van Wassenhove (1991). Multilevel capacitated lotsizing complexity and LP-based heuristics. *European Journal of Operational Research 53*(2), 131–148.

Masmoudi, O., A. Yalaoui, Y. Ouazene, and H. Chehade (2017). Lot-sizing in a multi-stage flow line production system with energy consideration. *International Journal of Production Research 55*(6), 1640–1663.

Mercan, H. M. and S. S. Erenguc (1993). A multi-family dynamic lot sizing problem with coordinated replenishments: a heuristic procedure. *International Journal of Production Research 31*(1), 173–189.

Mohamed, Z., M. Youssef, and F. Huq (2004). An efficient model for multifamily lot-sizing and scheduling: application to a real life problem. *Production Planning & Control 15*(1), 90–101.

Nananukul, N. (2013). Clustering model and algorithm for production inventory and distribution problem. *Applied Mathematical Modelling 37*(24), 9846–9857.

Naoum-Sawaya, J., B. Ghaddar, E. Arandia, and B. Eck (2015). Simulation-optimization approaches for water pump scheduling and pipe replacement problems. *European Journal of Operational Research 246*(1), 293–306.

Narayanan, A. and P. Robinson (2010). Efficient and effective heuristics for the coordinated capacitated lot-size problem. *European Journal of Operational Research 203*(3), 583–592.

Nikolaev, A. G. and S. H. Jacobson (2010). Simulated annealing. In M. Gendreau and J.-Y. Potvin (Eds.), *Handbook of Metaheuristics*, pp. 1–39. Springer.

Pochet, Y. and L. A. Wolsey (2006). *Production planning by mixed integer programming*. Springer.

Poojari, C. A. and J. E. Beasley (2009). Improving Benders decomposition using a genetic algorithm. *European Journal of Operational Research 199*(1), 89–97.

Rahmaniani, R., T. G. Crainic, M. Gendreau, and W. Rei (2017). The Benders decomposition algorithm: A literature review. *European Journal of Operational Research 259*(3), 801–817.

Reeves, C. R. (2010). Genetic algorithms. In M. Gendreau and J.-Y. Potvin (Eds.), *Handbook of Metaheuristics*, pp. 109–139. Springer.

Robinson, P., A. Narayanan, and F. Sahin (2009). Coordinated deterministic dynamic demand lot-sizing problem: A review of models and algorithms. *Omega 37*(1), 3–15.

Ruokokoski, M., O. Solyalı, J.-F. Cordeau, R. Jans, and H. Süral (2010). Efficient formulations and a branch-and-cut algorithm for a production-routing problem. GERAD Technical Report G-20 10-66, HEC Montréal Canada.

Silver, E. A. (1979). Coordinated replenishments of items under time-varying demand: dynamic programming formulation. *Naval Research Logistics 26*(1), 141–151.

Silver, E. A. and H. Meal (1973). A heuristic selecting lot size requirements for the case of deterministic time varying demand rate and discrete opportunities for replenishment. *Production and Inventory Management Journal 146*, 64–74.

Solyalı, O. and H. Süral (2012). The one-warehouse multi-retailer problem: Reformulation, classification, and computational results. *Annals of Operations Research 196*(1), 517–541.

Solyalı, O. and H. Süral (2017). A multi-phase heuristic for the production routing problem. *Computers & Operations Research 87*, 114–124.

Suerie, C. and H. Stadtler (2003). The capacitated lot-sizing problem with linked lot sizes. *Management Science 49*(8), 1039–1054.

Süral, H., M. Denizel, and L. Van Wassenhove (2009). Lagrangean relaxation based heuristics for lot sizing with setup times. *European Journal of Operational Research 194*(1), 51–63.

Süral, H., N. E. Özdemirel, Ý. Önder, and M. S. Turan (2010). An evolutionary approach for the TSP and the TSP with backhauls. In Y. Tenne and C.-K. Goh (Eds.), *Computational Intelligence in Expensive Optimization Problems*, pp. 371–396. Springer.

Toledo, C. F. M., M. da Silva Arantes, M. Y. B. Hossomi, and B. Almada-Lobo (2016). Mathematical programming-based approaches for multi-facility glass container production planning. *Computers & Operations Research 74*, 92–107.

Trigeiro, W. W., L. J. Thomas, and J. O. McClain (1989). Capacitated lot sizing with setup times. *Management Science 35*(3), 353–366.

Wagner, H. M. and T. M. Whitin (1958). Dynamic version of the economic lot size model. *Management science 5*(1), 89–96.

Warshall, S. (1962). A theorem on boolean matrices. *Journal of the ACM 9*(1), 11–12.

Zangwill, W. I. (1966). A deterministic multi-period production scheduling model with backlogging. *Management Science 13*(1), 105–119.

APPENDICES

Appendix A

Creation of Data Sets for *MF-CCLSP*

Since no standard test sets exist for the multiple family CCLSP, we developed 60 problems based on the single-family instances of Trigeiro, Thomas, and McClain (1989), labelled TTM. We describe here how these problems were formed, beginning first with an overview of the TTM instances used (Table A.1).

Table A.1: Description of TTM Data Sets

TTM Data Sets	m	$\times$	n
G51 - G55.dat	12	$\times$	15
G56 - G60.dat	24	$\times$	15

Table A.2 describes how the TTM instances were modified. All demand, setup cost, holding cost, setup time and capacity parameters were carried through, though in our work we do not use setup costs.

Since TTM instances do not include family setup times, we create them as follows: 100 setup times were randomly generated over $U\ (1,5)$ and then multiplied by 100. We assigned five of the generated family setup times to each of the original 10 problem instances (G51-G60.dat). However, for those new instances that have fewer than five families of items, only the first two or three of the generated setup times were needed.

With these added family setup times, the original capacity values had to be updated to ensure feasibility. In each period, capacity was initially increased by 300 units, the mean of the uniform distribution used above for the family setup times. Individual problem testing led to slight mod-

Table A.2: Description of Multi-Family Problem Set Creation

Problem Set	m	$\times$	1	$\times$	n	Description
k1-k5.dat	3	$\times$	2	$\times$	10	First 10 time periods of k16-k20.dat
k6-k10.dat	6	$\times$	2	$\times$	10	First 10 time periods of k21-k25.dat
k16-k20.dat	3	$\times$	2	$\times$	15	First 6 items of G51-G55.dat, split into two families
k21-k25.dat	6	$\times$	2	$\times$	15	Same as G51-G55.dat
k46-k50.dat	3	$\times$	3	$\times$	10	First 10 time periods of k61-k65.dat
k51-k55.dat	6	$\times$	3	$\times$	10	First 10 time periods of k66-k70.dat
k61-k65.dat	3	$\times$	3	$\times$	15	First 9 items of G51-G55.dat
k66-k70.dat	6	$\times$	3	$\times$	15	First 18 items of G56-G60.dat
k91-k95.dat	3	$\times$	5	$\times$	10	First 10 time periods of k106-k110.dat
k96-k100.dat	6	$\times$	5	$\times$	10	First 10 time periods of k111-k115.dat
k106-k110.dat	3	$\times$	5	$\times$	15	First 15 items of G56-G60.dat
k111-k115.dat	6	$\times$	5	$\times$	15	Combined G51-G55.dat with last 18 items of G56-G60.dat

ifications, making the capacity constraint tighter or looser based on the perceived difficulty of solving the reformulated problem directly, with no heuristics.

Appendix B

PRP Scenario 2 Results using SECs (3.11)

Scenario 2 is solved using SECs in the form of constraints (3.11). When the branch-and-cut algorithm is warm-started with the results of the upper bounding heuristic, overall solution time improves drastically. While larger instances ($T = 9$, $n \geq 30$) still do not solve to optimality, duality gaps reported in Table B.1 are much lower. The solution at the root node is much better, but not as good as when SECs (3.22) are used.

Table B.1: Scenario 2 Summary Without and With UB Heuristic using SECs (3.11)

					Without UB Heuristic					With UB Heuristic				
n	K	L	T	V	%LB	Nodes	CPU	%RLB	RCPU	%LB	Nodes	CPU	%RLB	RCPU
10	1	2	3	3	100.0	383.7	83.3	78.9	3.3	99.9	796.3	7.8	98.6	1.7
10	1	2	6	3	99.9	1439.3	26.1	64.1	4.5	99.9	2014	31.6	97.7	3.8
10	1	2	9	3	100.0	5008.0	95.8	53.6	10.3	99.9	1800.7	50.7	97.2	7.0
10	2	2	3	4	100.0	2769.3	41.8	50.8	6.3	99.9	3883.7	36.4	99.0	3.4
10	2	2	6	4	100.0	7457.7	208.3	34.2	18.7	100.0	8799.0	176.5	97.6	13.2
10	2	2	9	4	90.2	23318.0	2040.6	43.1	36.9$^{(1)}$	100.0	4828.3	626.7	96.3	27.6
10	5	2	3	5	99.9	1299.3	144.4	53.3	29.9	100.0	357.7	62.8	98.5	22.1
10	5	2	6	5	99.9	4594.7	2097.5	25.6	117.1	99.9	2053.7	714.1	97.3	83.2
10	5	2	9	5	74.7	4280.0	3600.2	33.0	211.4$^{(3)}$	98.6	2777.3	3600.3	96.7	441.7$^{(3)}$
20	1	2	3	3	100.0	358.0	36.8	74.8	17.0	100.0	524.3	35.1	98.4	12.1
20	1	2	6	3	100.0	7758.0	563.8	30.9	37.9	99.7	23346.7	1349.8	97.9	37.6$^{(1)}$
20	1	2	9	3	99.9	8863.0	1280.0	17.1	113.5	99.9	6808.0	1296.8	96.6	95.9
20	2	2	3	4	100.0	831.0	119.7	32.5	41.9	99.9	581.3	183.4	98.7	114.1
20	2	2	6	4	73.6	8980.0	2343.3	14.6	141.1$^{(1)}$	99.7	6281.7	1603.4	97.8	157.6$^{(1)}$
20	2	2	9	4	66.7	3325.0	3600.5	9.0	348.0$^{(3)}$	97.3	2001.7	3605.4	95.3	365.7$^{(3)}$
20	5	2	3	5	100.0	1247.0	761.0	24.3	167.1	100.0	438.0	414.7	98.0	156.7
20	5	2	6	5	71.7	2758.0	3198.6	13.8	292.3$^{(2)}$	99.6	2902.3	3601.1	98.1	312.6$^{(3)}$
20	5	2	9	5	8.3	376.5	3600.4	8.3	1761.5$^{(2)}$	97.4	218.0	3609.1	97.1	1629.9$^{(3)}$
30	1	2	3	4	76.4	4817.0	1468.0	30.5	142.3$^{(1)}$	99.9	675.3	350.9	96.4	162.0
30	1	2	6	4	84.6	2866.7	3614.1	21.2	493.4$^{(3)}$	98.9	2554.3	3367.8	95.8	400.8$^{(2)}$
30	1	2	9	4	24.1	1066.7	3653.0	16.5	746.6$^{(3)}$	97.8	1370.7	3600.6	96.9	636.4$^{(3)}$
30	2	2	3	5	75.9	1366.0	3063.9	22.7	1051.9$^{(2)}$	99.7	3872.0	1997.1	98.1	150.7$^{(1)}$
30	2	2	6	5	43.5	935.0	2977.4	25.4	564.5$^{(2)}$	98.3	1456.3	3419.8	96.7	634.5$^{(2)}$
30	2	2	9	5	8.8	49.0	3600.3	8.8	2361.7$^{(1)}$	95.1	33.3	3601.3	31.9	752.4$^{(3)}$
30	5	2	3	6	99.8	342.0	2376.5	29.7	662.5$^{(1)}$	99.4	539.0	3071.1	66.0	262.1$^{(2)}$
30	5	2	6	6	–	–	–	–	–	99.3	458.3	3747.0	99.2	971.2$^{(3)}$
30	5	2	9	6	–	–	–	–	–	96.9	0.0	3600.5	0.0	0.0$^{(3)}$
40	1	2	3	4	100.0	1416.3	1733.2	44.7	662.1	99.5	4634.7	2442.8	98.1	222.9$^{(1)}$
40	1	2	6	4	72.4	980.7	3550.8	19.1	742.7$^{(2)}$	99.0	3071.3	3617.7	98.6	468.3$^{(3)}$
40	2	2	3	5	71.3	1000.7	2407.8	19.2	443.2$^{(1)}$	100.0	617.0	1453.5	98.6	439.4
40	2	2	6	5	99.8	1043.0	3600.2	50.0	634.6$^{(1)}$	99.1	581.7	3602.5	98.4	1132.6$^{(3)}$
40	5	2	3	6	19.1	150.3	3600.2	18.5	2092.2$^{(3)}$	99.0	195.3	3442.2	66.3	685.6$^{(2)}$
40	5	2	6	6	–	–	–	–	–	98.5	12.3	3600.6	32.8	918.2$^{(3)}$
50	1	2	3	4	66.5	2647.7	3600.3	39.0	258.7$^{(3)}$	99.5	2470.3	3187.8	98.1	304.0$^{(2)}$
50	2	2	3	5	100.0	474.3	2161.8	41.9	546.6	99.3	548.0	3080.5	99.0	920.6$^{(2)}$
50	5	2	3	6	–	–	–	–	–	99.6	44.7	3600.6	99.6	2078.5$^{(3)}$

(#) Indicates number of instances (out of 3) that did not solve to optimality
– Indicates that no incumbent solution was found at the root note within the 1-hour time limit