Multi-period Production Setup-sequencing and Lot-sizing through ATSP Subtour Elimination and Patching

Alistair R. Clark
University of the West of England
School of Mathematical Sciences
Bristol, BS16 1QY, England.
Email: `Alistair.Clark@uwe.ac.uk`

Reinaldo Morabito Neto and **Eli A. V. Toso**
Universidade Federal de São Carlos
Departamento de Engenharia de Produção
São Carlos SP, 13565-905, Brazil.
Email: `Morabito@power.ufscar.br`
Email: `elitoso@yahoo.com`

Abstract

This paper considers a production lot sizing and scheduling problem at an animal feed plant with sequence-dependent setup times that do not satisfy the triangular inequality. A multi-period model based on the Asymmetric Travelling Salesman Problem (ATSP) is formulated, initially for the situation at the feed plant where the setup state is zeroed between periods, and then further developed for the case where the setup state is preserved from one period to the next. An iterative solution procedure based on subtour elimination is developed, and then enhanced by the inclusion of a subtour patching procedure. Computational tests with plant data that include other models and methods show that the subtour elimination is practicably fast where the setup state is zeroed between periods, but needs the patching procedure when the setup state is preserved. In this latter case the elimination and patching of ATSP-type subtours is very fast indeed, given the company's particular setup times, but needs further testing for other setup time configurations.

Key Words: Lot sizing, Production scheduling, Sequence-dependent setup times, Asymmetric travelling salesman problem, Subtour elimination, Subtour patching.

Introduction

In many manufacturing systems, production lots correspond to specific orders and so have a predetermined size. However a product may instead feed into many distinct orders with different deadlines. In such a situation, it can make economic sense to relate the product's lot-sizes to its total demand aggregated from the different orders, particularly when there is a setup cost or time charged for each lot. Such setups are often sequence-dependent, that is, the size of the setup charge depends on the product processed immediately beforehand in the sequence of production lots.

A good example is provided by the subject of this paper, namely, Anifeed, an animal-feed company (whose real name has been altered to protect its identity). Anifeed's production schedule needs to specify the lot sizes and setup sequence of many different feed mixes. The capacity-time needed for a setup is sequence-dependent, so that scheduling production to efficiently use capacity and still meet demand results in a problem which can be computationally too complex to solve exactly. The temptation is to deal with lot sizing and sequencing independently of each other, but in the animal feed industry, as in similar processes, this creates difficulties in being able to flexibly meet changing market demand and order due dates within the available production capacity. Thus an integrated modelling approach is needed, but there still remains the challenge of how to solve such a model effectively.

Anifeed currently produces about 200 animal feed supplements which can be grouped into about 20 families. Products within the same family do not contaminate each other and have the same production time per batch. All animal feed supplements follow the same basic production route, and make use of the same key resources: silos, dosing machines, pre-mix machines, a single mixer, and post-mix packaging, as shown in Figure 1. The amount of time spent at these operations varies somewhat between product families.

Once a batch has been processed at a given stage, it can then move onto the next stage, thus freeing the previous stage for another batch. The exception to this is that, to avoid contamination, the first batch of a new product family cannot enter the dosing machine until the last batch of the previous family has left the mixer. The mixer is considered to be the bottleneck that determines productive capacity and flow. Technically, the mixer must only be at least half-full to ensure efficient mixing, but there are economic advantages to producing a full batch. Thus lot size is restricted to be a whole number of batches.

Product changes are frequent, typically about 30 to 40 per week, but inter-family changeovers are far fewer. Some product families can cause contamination of other families, and so the feed mix blending equipment must sometimes be cleaned, resulting in substantial setup time of about 100 minutes and consuming scarce production time. The amount of cleaning can be minimised by the effective sequencing of production batches.

Most of Anifeed's products follow a seasonal pattern of demand, with peaks in certain months. Since employee turnover is high, manpower levels can be adjusted to cope with this seasonality, thus determining basic (pre-overtime) production capacity.

The demand in a particular period often exceeds capacity, and so overtime is frequently worked to satisfy demand. The animal feed market is highly competitive, and so deliv-

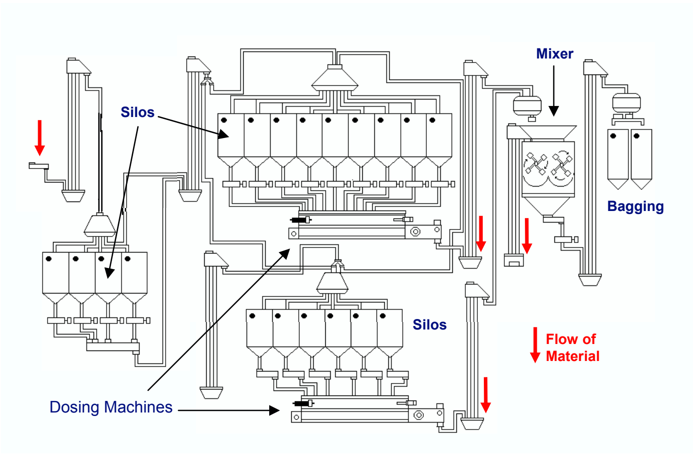

Figure 1: Production Process at Anifeed

ery delays to clients must be avoided if possible. A further possibility to avoid delivery delays is to produce some feeds ahead of demand when slack capacity is available. However, animal feed products are perishable and so cannot be produced too far in advance. Anifeed aims not to produce feeds more than about a month in advance of shipping to clients.

This paper develops and tests a modelling and solution approach for Anifeed's lot sizing and scheduling problem based on a formulation related to the Asymmetric Travelling Salesman Problem (ATSP). Following a brief review of previous research, the paper develops two ATSP-based models with non-triangular setup times, a feature of many animal feed plants. The first model represents the situation where it is possible to prepare setups between productive periods, which is the case at Anifeed. The second model considers the more difficult situation where the setups state is carried over between periods. Optimal solution methods are then developed, based on iterative subtour elimination and patching. After outlining an alternative model and solution approach for experimental comparison, computational tests are carried out on all models and methods using randomly perturbed data based a typical month at Anifeed. The paper concludes with a discussion and pointers for future research.

Review of previous research

Research into lot sizing and scheduling research has been carried out for many years, as shown by the surveys of Wolsey (1995), Drexl and Kimms (1997) and Karimia et al. (2003). However, lot-sizing and *sequencing* has been studied by fewer researchers. The Capacitated Lot-Sizing Problem (CLSP) in Drexl and Kimms (1997), for example, does not include sequencing decisions. Smith-Daniels and Smith-Daniels (1986) formulated models for the CLSP with sequence-dependent set-up times, but their solution methods were restricted to small problems. Arosio and Sianesi (1993) proposed a complex heuristic algorithm for simultaneous lot-sizing and sequencing on a single machine with sequence-dependent set-ups, but made no comparisons with optimal solutions in their computational tests. This paper does make such comparisons for a mid-sized real problem.

The General Lotsizing and Scheduling Problem (GLSP), developed by Fleischmann and Meyr (1997), minimises inventory and sequence-dependent setup costs on a single machine with finite capacity, allowing multiple setups in each single "large-bucket" time period. The GLSP was extended by Meyr (2000) who formulated the General Lot Sizing Problem - Setup Times (GLSP-ST) model for simultaneous lot-sizing and scheduling on a single production line with sequence-dependent setup times. He divided the planning periods into a predetermined number of "small-bucket" micro-periods which contain at most one setup. Meyr did not model the backlogging of demand (a consequence of insufficient capacity). The wide-ranging review on capacitated lot sizing by Karimia et al. (2003) notes that, surprisingly, there is very little literature on problems with backlogging, a feature of the model developed in this paper.

The Asymmetric Travelling Salesman Problem (ATSP) has been very extensively researched (Lawler et al.; 1985; Laporte; 1994; Junger et al.; 1995; Carpaneto et al.; 1995; Zhang; 1997; Glover et al.; 2001; Cirasella et al.; 2001; Johnson et al.; 2002; Buriol et al.; 2003) and can be adapted to model the problem of sequencing a set of lots with sequence dependent setups between them. For example, Salomon et al. (1997) transformed the Discrete Lotsizing and Scheduling Problem with sequence dependent set-up costs and set-up times (DLSPSD) into a TSP with Time Windows (TSPTW) and used dynamic programming to solve it. As will be shown below, the adaptation in this paper is not direct, since the production system is often already setup for a particular product (i.e., starting at a given city) and some products may not be produced in a given period if the demand is sufficiently small or the capacity tight.

A method that has been found to be successful in practice for optimally solving the ATSP is to quickly solve the corresponding Assignment Problem (AP) as a linear programme, identify the resulting subtours, and then resolve the AP, explicitly prohibiting these subtours. The method carries on iteratively in this manner until no subtours result. It can be used heuristically (and its convergence rate sometimes accelerated) by patching the subtours into a single tour at each iteration (Karp; 1979; Karp and Steele; 1985; Frieze and Dyer; 1990; Frieze et al.; 1995; Zhang; 1997), thus providing a feasible solution (and an upper bound). This paper adapts the subtour elimination method to lot sequencing first over single periods, and then over multiple periods with setup carryover between periods. An extension of the method then successfully uses the patching heuristic to greatly accelerate convergence to an optimal solution.

Modelling Approaches

A model to help Anifeed schedule production more efficiently is now formulated as a mixed integer linear programme (MIP) with continuous inventory & backorders variables, integer lot-size variables, and binary setup-sequence variables. The model minimises inventory, backorders and overtime, three criteria of major importance to the company, while keeping within available capacity and overtime limits.

The following indices are used:

i :	Product family, i = 1, ..., N
t :	Time period, t = 1, ..., T

where:

N =	the number of families
T =	the number of periods in the scheduling horizon

The input data required by the model are:

C_t	Available capacity time in each period t.
p_i	Time needed to produce one batch of each product family i.
lm_i	Minimum lot size of product family i (an integer number of batches).
h_i	Cost of holding one week's inventory of product family i.
co_t	Unit cost of overtime for week t.
st_{ji}	Setup time needed to changeover from product family j to family i
d_{it}	Forecast of demand for product family i at the end of week t.
I_{i0}	Inventory of product family i at the start of the planning horizon.
u_t	Upper limit on the number of overtime hours permitted in period t.

The decisions output by the model are:

I_{it}^+	Inventory of product family i at the end of period t.
I_{it}^-	Backlogs of product family i at the end of period t.
q_{it}	(Integer) production lot size of product family i in period t. Since demand occurs only at the end of the periods, each family need only be produced once (or not at all) in any period.
y_{jit}	= 1 if production is to be changed over from product family j to family i in period t, otherwise = 0.
O_t	Number of overtime hours needed in period t.

Modelling setup preparation between periods

In many companies, including Anifeed, the setup of the initial family produced in a period takes place in the elapsed time gap between planning periods (for example, at the weekend), without eating into productive capacity. This situation is modelled first. The carry-over of a setup configuration from one period to the next is modelled afterwards.

Let i_0 represent a "phantom" family from which there is zero setup time to any other family. If the setup state at the start of a period is i_0, then production of any other family can begin immediately since $st_{i_0,i} = 0$ for all families i. In reality the setup to the first family may not have time zero, but since it occurs between periods, it should not be included in capacity requirement calculations (such as in constraints (3) below). This is the only purpose of family i_0, so setups to it from any family are prohibited by setting $st_{i,i_0} = \infty \; \forall \; i$. In addition, it has zero production time and demand, i.e., $p_{i_0} = 0$ and $d_{i_0,t}0 \; \forall \; t$.

A mixed integer programming (MIP) formulation based on the Asymmetric Travelling Salesman Problem (ATSP) now follows.

Model AtspAnifeed: The objective function minimizes heavy backlog penalties, and the costs of inventory & overtime:

$$\text{Minimise} \quad \sum_i \sum_t h_i \left(I_{it}^+ \; + \; 1000 \; I_{it}^-\right) \\ + \sum_t co_t \; O_t \qquad (1)$$

Differently to Hax and Candea (1984) and Meyr (2000), expression (1) does not consider setup costs, the reason being that setups basically consume just labour and time, neither of which incurs an immediate direct cost. Setup times are not directly penalized in the objective function (1), but indirectly though their use of overtime. Thus the minimisation of overtime in the objective function will prevent superfluous setups via the capacity constraints (3) below.

Constraints (2) balance inventory, backlogs, production and demand over consecutive weeks:

$$I_{i,t-1}^+ \; - \; I_{i,t-1}^- \; + \; q_{it} \; - \; d_{it} \; = \; I_{it}^+ \; - \; I_{it}^- \\ \forall \; i, t \qquad (2)$$

The capacity constraints (3) take into account the setup times as well as actual production times, and the possibility of a limited amount of overtime:

$$\sum_i p_i \, q_{it} + \sum_j \sum_i st_{ji} \; y_{jit} \; \le \; C_t + O_t \quad \forall \; t \qquad (3)$$

Note that the first setup will be from the "phantom" family i_0 and so have setup time zero.

Constraints (4) ensure that production of a family can occur in a period only if the line is set up accordingly:

$$p_i \, q_{it} \; \le \; (C_t \; + \; u_t) \sum_j y_{jit} \qquad \forall \; t, \; i \neq i_0 \qquad (4)$$

Constraints (5) enforce a minimum lot size and are needed as setup times do not always satisfy the triangular inequality:

$$q_{it} \; \ge \; lm_i \sum_j y_{jit} \qquad \forall \; t, \; i \neq i_0 \qquad (5)$$

To better understand why constraints (5) are needed, consider a family i whose production contaminates that of family k unless a thorough cleaning occurs as part of the substantial setup time st_{ik}. In the animal feed industry, such cleaning can sometimes occur during the production of an intermediate family j whose setup time st_{ij} from i and setup time st_{jk} to j are jointly short enough that $st_{ik} > st_{ij} + st_{jk}$. Thus the triangular inequality $st_{ik} \le st_{ij} + st_{jk}$ does not hold in this case. Without constraints (5) to impose sufficient production of j to allow proper cleaning of i's contaminants, an optimal schedule could setup from i to k via zero production of j rather than directly.

Note that the disobeying of the triangular inequality implies that it could be optimal in certain circumstances for any intermediate "cleansing" family j to be produced in more than one lot within the same period. The model's assumption of at most one lot per family per period would not hold in such a situation. The development of an appropriate optimal model is not tackled in this paper, but flagged as a subject for future research.

Note also that the triangular inequality always holds for the zero-valued setup times from phantom family i_0, but that constraints (5) will, in effect, not apply to the first product produced in a period given such zero-valued setups.

Constraints (6) prohibit setups between the same family:

$$y_{iit} \;=\; 0 \qquad\qquad \forall\, i,\; t \tag{6}$$

Constraints (7) permit a setup to a family only if there is a setup from the family already setup at the start of each period:

$$\sum_j y_{i_0,j,t} \;\geq\; \sum_k y_{kit} \qquad \forall\, i \neq i_0,\; t \tag{7}$$

Constraints (8) permit a setup from a family (other than i_0) only if it has been setup to:

$$\sum_i y_{ijt} \;\geq\; \sum_k y_{jkt} \qquad \forall\, j \neq i_0,\; t \tag{8}$$

Constraints (9) prohibit more than one setup from a family:

$$\sum_j y_{ijt} \;\leq\; 1 \qquad\qquad \forall\, i,\; t \tag{9}$$

Constraints (10) prohibit family subtours in each period:

$$\sum_{i \to j\, \in S} y_{ijt} \;\leq\; |S| - 1 \qquad \forall \text{ subtours } S,\; t \tag{10}$$

as adapted from the ATSP subtour exclusion constraints in Orman and Williams (2004) and Carpaneto et al. (1995). By summing the reverse as well as the forward arcs of each subtour, constraints (10) can be replaced by:

$$\sum_{i \to j\, \in S} (y_{ijt} + y_{jit}) \;\leq\; |S| - 1 \;\; \forall \text{ subtours } S, t \tag{11}$$

which are at least as strong as (10) for the following reason: for a given subtour S, any combination of the variables $\{y_{ijt}\;,\; y_{jit} \mid i \to j \in S\}$ that disobeys constraint (11) will either (i) be S, (ii) be its reverse subtour, or (iii) have a family $i \in S$ for which $y_{ijt} = 1$ for two $j \in S$. All three of these possibilities are prohibited.

Constraints (12) impose limits on overtime working:

$$0 \;\leq\; O_t \;\leq\; u_t \qquad\qquad \forall\, t \tag{12}$$

Constraints (13) prohibit the inventory and backlogs variables to be negative:

$$I^+_{it},\; I^-_{it},\; \geq\; 0 \qquad\qquad \forall\, i, t \tag{13}$$

Constraints (14) ensure the setup changeover y_{ijs} variables are binary:

$$y_{ijt} = 0 \text{ or } 1 \qquad\qquad \forall\, i, j, t \tag{14}$$

Constraints (15) require the production to be a whole number of batches.

$$q_{it} \;\geq\; 0 \text{ and integer} \qquad\qquad \forall\, i, t \tag{15}$$

Given that the setup times of some cleansing families may break the triangular inequality, the above formulation is not guaranteed to produce an optimal solution.

Note the huge number of subtour prohibition constraints (11). However, the vast majority will not be binding at the model's optimal solution so, imposing the constraints selectively just for those subtours that occur, a series of Assignment-type problems can be solved, as shown later in this paper.

Given that $st_{i,i_0} \;=\; \infty \;\forall\; i$, constraints (16) below are redundant, but can be added to speed up the solution time. They ensure that there is no setup to the "phantom" family i_0 already setup at the start of each period:

$$y_{i,i_0,t} \;=\; 0 \qquad\qquad \forall\, i, t \tag{16}$$

Modelling setup carryovers between periods

Until now in this paper, the initial setup state at the start of each period t has been the phantom family i_0 rather than a variable. This accords with the context and practice of many companies (including Anifeed) where the initial setup can occur between planning periods without eating into capacity. For example, the Anifeed plant operates from Monday to Saturday only, enabling the first family produced in a period to be setup in advance on Sunday by maintenance personnel.

However, many companies restart production with the same setup state as at the end of the previous period. In this case, the carry-over of a setup state from the end of one period to the start of the next can [and should] be treated as a variable. This is modelled as follows.

Model AtspCarryover: Declare the following new variable:

z_{it} = 1 indicates that family i is the setup state at the start of period t, otherwise = 0.

The existing setup configuration z_{i1} is known and so is a parameter rather than a variable.

The objective function (1) and constraints (2), (3), (6), (9) and (11)-(15) remain unchanged from model AtspAnifeed. However, other constraints must be modified or added in order to model the carryover of a setup state from the end of one period to the start of the next, as follows.

Clearly, only a single family can be the setup configuration at the start of a period, enforced by a new constraint:

$$\sum_i z_{it} \;=\; 1 \qquad\qquad t = 2, ..., T \tag{17}$$

Constraints (4) are replaced by (18) below, still ensuring that production of a family can occur in a period only if the line is set up accordingly. Note that constraints (18) permit the production of the family already setup at the start of period t.

$$p_i\, q_{it} \;\leq\; (C_t + u_t) \left(z_{it} + \sum_j y_{jit} \right) \qquad \forall\, i, t \tag{18}$$

Constraints (5), which enforce a minimum lot size, need not apply to the family already setup at the start of a period, and so are replaced by (19) below:

$$q_{it} \;\geq\; lm_i \left(\sum_j y_{jit} - z_{it} \right) \qquad \forall\, i,\; t \qquad (19)$$

Constraints (7) are replaced by (20), permitting a setup to a family only if it is set up from, or is the family already setup at the start of the next period:

$$\sum_i y_{ijt} \leq \sum_k y_{jkt} + z_{j,t+1} \;\; \forall\, j,\; t = 1, ..., T{-}1 \quad (20)$$

Constraints (8) are replaced by (21), permitting a setup from a family only if it has been setup to or is already setup at the start of a period:

$$z_{jt} \;+\; \sum_i y_{ijt} \;\geq\; \sum_k y_{jkt} \qquad \forall\, j,\; t \qquad (21)$$

The redundant constraints (16) are replaced by the now necessary constraints (22) which prohibit a setup to the family already setup at the start of a period:

$$1 \;-\; z_{it} \;\geq\; \sum_j y_{jit} \qquad \forall\, i,\; t \qquad (22)$$

Additional constraints (23) and (24) are needed, both to prohibit a setup from the family already setup at the start of the next period:

$$1 \;-\; z_{i,t+1} \;\geq\; \sum_j y_{ijt} \quad \forall\, i,\; t = 1, ..., T{-}1 \quad (23)$$

and to ensure there must be a setup from the family already setup at the start of the period. The rare case of no setups at all in that period is an alternative possibility that is not modelled in constraint (23).

$$z_{it} \;\leq\; \sum_j y_{ijt} \qquad \forall\, i,\; t = 1, ..., T \qquad (24)$$

Solution Approaches

Subtour Elimination

The initial optimal solution to models AtspAnifeed or AtspCarryover will consist of zero or more subtours and a single sequence starting with the family already setup at the start of the period and ending with the last family setup in the period. In the following solution method, the subtours that arise in one period are prohibited in all periods in subsequent iterations.

Methods AtspAnifeed and AtspCarryover

```
Repeat {
    Solve Model AtspAnifeed or AtspCarryover prohibiting
    only those subtours encountered so far in any period.
    **Comment:** provides a *lower* bound to optimal solution.
    For $t = 1, \ldots, T$ do {
        Identify any subtours in period $t$ and prohibit them
        in all periods from now on;
    }
    Resolve;
} While subtours exist in any period;
Comment: the solution is now optimal.
```

Note that an eventual feasible solution with no subtours will not feature a circular tour in each period, but rather a single unbroken sequence of up to N families starting with the family already setup at the start of the period and ending with the last family setup in the period.

Patching as well as prohibiting the subtours after optimising all periods simultaneously

A fast procedure that can give good and often near-optimal results for the ATSP is to solve an assignment problem and then use a *patching heuristic* (Karp; 1979; Karp and Steele; 1985; Frieze and Dyer; 1990; Frieze et al.; 1995; Zhang; 1997) to gather (i.e., patch) the optimal assignment subtours into a single salesman tour.

The following extension of Method AtspCarryover patches the subtours in each period to obtain a single unbroken sequence, and thus a feasible solution, at each iteration.

Method AtspPatching

```
Repeat {
    Solve Model AtspCarryover with no subtour constraints
    using least patching upper bound as initial incumbent;
    Comment: provides a lower bound to optimal solution.
    For $t = 1, \ldots, T$ do {
        Identify any subtours in period $t$ and prohibit them
        in all periods from now on;
    }
    Resolve;
    For $t = 1, \ldots, T$ do {
        Patch any subtours in period $t$ into a single
        unbroken sequence;
    }
    Fix the $T$ single-tour sequences and resolve;
    Comment: The resulting feasible solution provides an
    upper bound to the optimal solution of the model.
    Unfix the single-tour sequences;
} While subtours exist in any period;
Comment: the solution is now optimal.
```

The patching operation gradually joins the single sequence and the subtours together so that the increase in total setup time is minimised, starting with the two subtours or sequence that have the most families, as in Karp (1979) and Karp and Steele (1985).

The GLSP model and Relax-&-Fix

The *relax-&-fix* method (Wolsey; 1998) solves a sequence of partially-relaxed MIPs, each one with a reduced set of integer variables whose number is small enough to quickly obtain optimal MIP solutions (Clark and Clark; 2000). As the series progresses, each set of integer variables is permanently fixed at their solution values. The procedure is broadly similar to a depth-first identification of an initial integer solution for a MIP model in a large branch-&-bound search. It is fast, but solutions are sometimes poor.

The computational tests below include two types of the *relax-&-fix* method, applying it to an alternative MIP formulation developed in Toso (2003) and Toso et al. (2006), in which the setup configuration at the start of each period is a variable rather than a parameter, similar to the *General Lot Sizing Problem - Setup Times* (GLSP-ST) proposed by Meyr (2000). The tests also try to solve the exact model optimally as a benchmark (Method Glsp).

The first type of *relax-&-fix* method relaxes the binary setup-state variables in periods 2 onwards, solves the model, permanently fixes the period 1 setup values, restores the integrality constraints to the setup variables in period 2 keeping them relaxed for periods 3 onwards, solves the model again, permanently fixes the period 2 setup values, and so on until period T (Method GlspRFSetup). The second type of *relax-&-fix* method relaxes the integrality of the lot-size variables, solves the model for the binary setup variables, permanently fixes them, restores the integrality constraints to the lot-size variables and solves the model again (Method GlspRFSize). Details can be found in Toso et al. (2006).

Computational Tests

The tests now compare the quality and run time of the following six models and methods:

1. ATSP Model [Anifeed's situation: the inter-period setup state is a *parameter*]
   (a) Method AtspAnifeed: Solve in all periods simultaneously then prohibit subtours.
2. ATSP Model [inter-period setup state is a *variable*]
   (a) Method AtspCarryover: Solve in all periods simultaneously then prohibit subtours.
   (b) Method AtspPatching: Patch as well as prohibit the subtours after solving in all periods simultaneously.
3. Toso (2003) Model [similar to GLSP-ST, inter-period setup state is a *variable*]
   (a) Method Glsp: Exact.
   (b) Method GlspRFSetup: *Relax-&-Fix* on lot setup-state.
   (c) Method GlspRFSize: *Relax-&-Fix* on lot size.

The test data was taken from Anifeed, comprising demand for 21 families over T=4 weeks. The sequence-dependent inter-family setup times were either 0 or 100 minutes. For each of the 20 runs, the demand pattern was randomly perturbed around the actual values taken from a rainy season month. Each MIP solved within the methods was allowed to run for a maximum of one hour of CPU time, with the exception that the GlspRFSetup method's 4 separate MIPs each had a limit of 15 minutes. The models and methods were implemented in the AMPL mathematical programming language (Fourer et al.; 2003). The MIPs were solved using Cplex 9.1 (Ilog; 2004). The tests were run on a 2.00 GHz 798MHz Pentium M processor with 1.5Gb of RAM.

Figure 2 shows box & whisker plots for the solution values produced by six methods. As expected, method AtspAnifeed has a lower value than the other five methods, since the first setup is carried out at the weekend between production periods (and so is a different problem than the other five).

The other 5 methods solve the same problem (where the setup state is carried over between production periods). Note from Figure 2's plots (which show medians, quartiles, and extremes) that method GlspRFSetup clearly performs worse

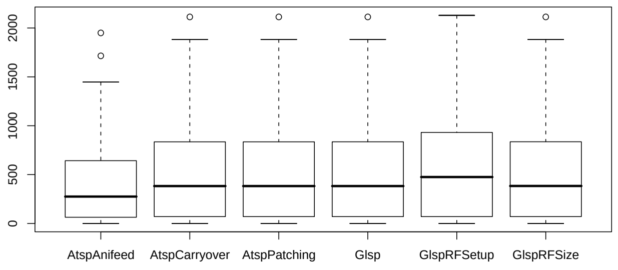

Figure 2: Solution Box & Whisker plots for each Method

than the other four. The solution values of these remaining four methods differ very little, but an Analysis of Variance, with the run as a random factor, showed highly significant differences between the four (p-value < 0.001), with the AtspCarryover and AtspPatching methods being equally best. All 20 runs produced identical values for these two methods with mean 597.14, while GlspRFSize and Glsp had means 600.47 and 605.88 respectively.

Figure 3 shows box & whisker plots for the solution CPU times of six methods. Note immediately the fast times of the AtspAnifeed and AtspPatching methods, contrasting with the slowness of the other four methods.

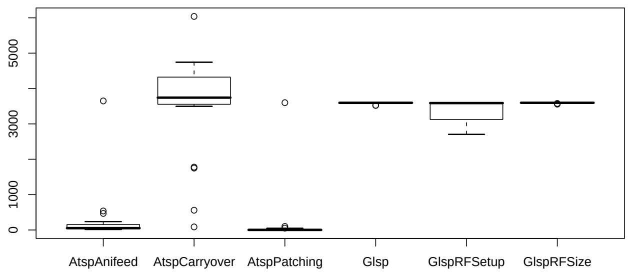

Figure 3: CPU time Box & Whisker plots for each Method

The only MIP of method Glsp and the first MIP of the GlspRFSize method never converged within their CPU time limit of 1 hour. The second MIP of the GlspRFSize method usually converged very quickly, generally within a few seconds. The GlspRFSetup method's 4 separate MIPs very rarely converged converged within each's CPU time limit of 15 minutes. Thus the GLSP-based methods all took about 1 hour, occasionally less.

The AtspCarryover method had a very variable CPU time performance. Each iteration's MIP was also limited to 1 hour of CPU time. Although this limit was reached infrequently, it was sufficiently often to result in a median total solution time of just over an hour. A median 66.35 iterations and 189 subtours were required for convergence. The final MIP always converged to optimality within the time limit, thus guaranteeing optimality of the no-subtours solution.

Remember that the AtspCarryover method is optimising linked sequences over T=4 periods simultaneously, while the AtspAnifeed model is optimising 4 delinked sequences of 1 periods each. Thus it is not surprising that the AtspAnifeed method converged far more quickly than the AtspCarryover with mean time of 297.5 seconds (i.e, under 5 minutes) and median 52.6 seconds (i.e., under a minute). These are operationally acceptable times for Anifeed's schedulers.

The most encouraging result of all is the very fast convergence afforded by the use of the patching algorithm at each iteration of method AtspPatching. In fact, the algorithm always gave a patched solution with the same value as the unpatched subtour solution. In other words, the first iteration's upper and lower solution bounds were identical, thus proving the optimality of the patched solution. This resulted in a mean solution time of 193.5 seconds, and a very low median of just 2.1 seconds, essentially the solution time of the starting assignment-type MIP. However, the convergence after just one iteration is quite possibly due to the existence of many alternative optimal solutions, resulting from the 0 or 100-minute setup times.

Future computational tests will randomly perturb the 100-minute setup times values to lie in [80, 120], and thus test the methods under circumstances of varying setup values.

Conclusions

The experimental results showed that the GLSP-based methods were able to produce near-optimal solutions within an hour's CPU time, but almost always without proven convergence. The ATSP-based methods were more successful. When there was inter-period cleansing without setup carryover, the ATSP subtour elimination method was able in 19 of the 20 runs to converge to a provably optimal solution, without patching. However, the adaptation to include setup carryover resulted in a larger MIP that took much longer to solve, generally not converging within 1 hour. The way around this was the patching of subtours to produce a feasible solution and upper bound that enabled the setup carryover model to converge extremely rapidly, almost always within 1 subtour elimination iteration. This rapid convergence, however, was possibly due to the fact that all non-zero setup times have the same value of 100 minutes, thus supplying not only multiple optimal solutions, but also subtour solutions with the same (optimal) solution value.

The next step in the research is to extend the computational tests to randomly perturb the 100-minute setup times values to lie in the interval [80,120], and thus test the methods under circumstances of varying setup values. The relative performance of the methods could be quite different to the test results above.

Future research includes further testing with different data, and the development of models that recognise that the disobeying of the triangular inequality implies that it could be optimal in certain circumstances for any intermediate cleansing family to be produced in more than one lot within the same period.

References

Arosio, M. and Sianesi, A. (1993). A heuristic algorithm for master production scheduling generation with finite capacity and sequence dependent setups, *International Journal of Production Research* **31**: 531–553.

Buriol, L., Franca, P. M. and Moscato, P. (2003). A new memetic algorithm for the asymmetric traveling salesman problem, Technical Report, Faculty of Electrical Engineering, Universidade Estadual de Campinas, Brazil.

Carpaneto, G., Dell'Amico, M. and Toth, P. (1995). Exact solution of large scale asymmetric travelling salesman problems, *ACM Transactions on Mathematical Software* **21**(4): 394–409.

Cirasella, J., Johnson, D. S., McGeoch, L. A. and Zhang, W. (2001). The asymmetric traveling salesman problem: Algorithms, instance generators, and tests, *Springer Lecture Notes in Computer Science* **2153**: 32–59.

Clark, A. R. and Clark, S. J. (2000). Rolling-horizon lot-sizing when setup times are sequence-dependent, *International Journal of Production Research* **38**(10): 2287–2308.

Drexl, A. and Kimms, A. (1997). Lot sizing and scheduling - survey and extensions, *European Journal of Operational Research* **99**: 221–235.

Fleischmann, B. and Meyr, H. (1997). The general lotsizing and scheduling problem, *OR Spektrum* **19**(1): 11–21.

Fourer, R., Gay, D. M. and Kernighan, B. W. (2003). *AMPL - A Modeling Language for Mathematical Programming*, second edn, Duxbury Press / Brooks-Cole Publishing Company, USA. http://www.ampl.com/.

Frieze, A. M. and Dyer, M. E. (1990). On a patching algorithm for the random asymmetric travelling saleman problem, *Mathematical Programming* **46**: 361–378.

Frieze, A. M., Karp, R. M. and Reed, B. (1995). When is the assignment bound tight for the asymmetric traveling-salesman problem?, *SIAM Journal on Computing* **24**(3): 484–493.

Glover, F., Gutin, G., Yeo, A. and Zverovich, A. (2001). Construction heuristics for the asymmetric tsp, *European Journal of Operational Research* **129**: 555–568.

Hax, A. and Candea, D. (eds) (1984). *Production and Inventory Management*, Prentice-Hall, Englewood Cliffs, New Jersey.

Ilog (2004). *CPLEX 9.1 User's Manual*, ILOG S.A. See website www.cplex.com.

Johnson, D. S., Gutin, G., McGeoch, L. A., Yeo, A., Zhang, W. and Zverovich, A. (2002). Experimental analysis of heuristics for the asymmetric travelling salesman problem - algorithms, instance generators and tests, *in* G. Gutin and A. P. Punnen (eds), *The Traveling Salesman Problem and its Variations*, Kluwer.

Junger, M., Reinelt, G. and Rinaldi, G. (1995). The traveling salesman problem, *in* M. O. Ball, T. L. Magnanti, C. L. Monma and G. L. Nemhauser (eds), *Network Models*, Vol. 7 of *Handbooks in Operations Research and Management Science*, North Holland, chapter 4, pp. 225–330.

Karimia, B., Fatemi Ghomia, S. M. T. and Wilson, J. M. (2003). The capacitated lot sizing problem: a review of models and algorithms, *Omega* **31**: 365–378.

Karp, R. M. (1979). A patching algorithm for the non-symmetric traveling-salesman problem, *SIAM Journal on Computing* **8**(4): 561–573.

Karp, R. M. and Steele, J. M. (1985). Probabilistic analysis of heuristics, *in* E. L. Lawler, J. K. Lenstra, A. H. G. Rinnoy Kan and D. B. Shmoys (eds), *The Traveling Salesman Problem - A Guided Tour of Combinatorial Optimization*, Wiley, Chichester.

Laporte, G. (1994). The travelling salesman problem: An overview of exact and approximate algorithms., *Europena Journal of Operational Research* **59**: 231–247.

Lawler, E. L., Lenstra, J. K., Rinnoy Kan, A. H. G. and Shmoys, D. B. (1985). *The Traveling Salesman Problem - A Guided Tour of Combinatorial Optimization*, Wiley, Chichester.

Meyr, H. (2000). Simultaneous lot-sizing and scheduling by combining local search with dual optimization, *European Journal of Operational Research* **120**: 311–326.

Orman, A. J. and Williams, H. P. (2004). A survey of different integer programming formulations of the travelling salesman problem, *Working paper LSEOR 04.67*, London School of Economics, Dept of Operational Research, LSE, Houghton Street, London, WC2 2AE, UK.

Salomon, M., Solomon, M., Van Wassenhove, L. N. and Dumas, Y. (1997). Solving the discrete lot sizing and scheduling problem with sequence dependent set-up costs and set-up times using the travelling salesman problem with time windows, *European Journal of Operacional Research* **100**: 494–513.

Smith-Daniels, V. L. and Smith-Daniels, D. E. (1986). A mixed integer programming model for lot sizing and sequencing packaging lines in the process industries, *IIE Transactions* **18**(3): 278–285.

Toso, E. A. V. (2003). *Otimização do problema integrado de dimensionamento de lotes e programação da produção: Estudo de caso na indústria de rações*, Dissertação de mestrado, Universidade Federal de São Carlos, Departamento de Engenharia de Produção, Rodovia Washington Luís (SP-310), Km 235, São Carlos SP, 13565-905, Brazil.

Toso, E. A. V., Morabito, R. and Clark, A. R. (2006). *Lot-Sizing and Sequencing Optimisation at an Animal-Feed Plant*, submitted for publication.

Wolsey, L. A. (1995). Progress with single-item lot-sizing, *European Journal of Operational Research* **86**: 395–401.

Wolsey, L. A. (1998). *Integer Programming*, Wiley, New York.

Zhang, W. (1997). A note on the complexity of the asymmetric travelling salesman problem, *Operations Research Letters* **20**: 31–38.