The Production Transportation Scheduling System

by

Peter L. Jackson
THE PRODUCTION TRANSPORTATION SCHEDULING SYSTEM

PETER L. JACKSON
ASSOCIATE PROFESSOR
AUGUST, 1998
# TABLE OF CONTENTS

I. ABSTRACT ........................................................................................................... 4

A. WHO SHOULD READ THIS MANUAL? ......................................................... 4

B. THE ROLE OF PTS IN ENTERPRISE MANAGEMENT .............................. 5

C. OVERVIEW OF THIS MANUAL ................................................................... 8

II. SYSTEM FUNCTIONALITY ........................................................................... 10

A. THE PTS FUNCTIONAL PROCESS .............................................................. 10

B. BUILDING THE PTS DATABASE ............................................................... 12

C. DEVELOPING THE SCHEDULE ................................................................. 15

D. EXPORTING THE SCHEDULE .................................................................. 18

III. SYSTEM OPTIMIZATION ......................................................................... 20

A. FLAG FORMATION OPTIMIZATION .......................................................... 20

B. STOP SEQUENCE OPTIMIZATION ........................................................... 25

C. PRODUCTION SMOOTHING OPTIMIZATION .......................................... 29

   A Mathematical Formulation of the Production Smoothing Problem ... 30

IV. SYSTEM COMPONENTS ............................................................................. 32

A. PTS DIRECTORY STRUCTURE ................................................................. 36

B. TRANSCAD MAPPING MACROS FILE ORGANIZATION .......................... 42

C. PTS USER INTERFACE VIEWS ................................................................. 44

D. PTS VISUAL BASIC PROJECT STRUCTURE ........................................... 51

V. APPENDIX: FIGURES IN LANDSCAPE VIEW ............................................ 53
**LIST OF FIGURES**

<table>
<thead>
<tr>
<th>Figure</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>I.1</td>
<td>Enterprise Management View</td>
<td>5</td>
</tr>
<tr>
<td>I.2</td>
<td>Enterprise Management - Sales, Dispatch, Capacity Management and Production Control</td>
<td>7</td>
</tr>
<tr>
<td>I.3</td>
<td>The Production Transportation Scheduling System Role in Enterprise Management</td>
<td>8</td>
</tr>
<tr>
<td>II.1</td>
<td>The PTS Functional Process</td>
<td>10</td>
</tr>
<tr>
<td>II.2</td>
<td>Build PTS Database Process (A2)</td>
<td>13</td>
</tr>
<tr>
<td>II.3</td>
<td>Export Revised Order File (A5)</td>
<td>18</td>
</tr>
<tr>
<td>III.1</td>
<td>Flag Formation Process</td>
<td>20</td>
</tr>
<tr>
<td>III.2</td>
<td>Form Single Flag Process (F1)</td>
<td>22</td>
</tr>
<tr>
<td>III.3</td>
<td>Improve Set of Flags Process (F3)</td>
<td>23</td>
</tr>
<tr>
<td>III.4</td>
<td>ARC Swapping (Stop Sequence Optimization)</td>
<td>26</td>
</tr>
<tr>
<td>III.5</td>
<td>Sequence Flag (F12)</td>
<td>27</td>
</tr>
<tr>
<td>III.6</td>
<td>Production Smoothing Optimization</td>
<td>29</td>
</tr>
<tr>
<td>IV.1</td>
<td>PTS System Components</td>
<td>33</td>
</tr>
<tr>
<td>IV.2</td>
<td>PTS Directory Structure</td>
<td>37</td>
</tr>
<tr>
<td>IV.3</td>
<td>TRANSCAD Mapping Macros File Organization</td>
<td>42</td>
</tr>
<tr>
<td>IV.4</td>
<td>PTS User Interface Views</td>
<td>44</td>
</tr>
<tr>
<td>IV.5</td>
<td>PTS User Interface Visual Basic Project Structure</td>
<td>52</td>
</tr>
</tbody>
</table>
I. ABSTRACT

The Production Transportation Scheduling System (PTS) is a software system that integrates functions in production and transportation scheduling and provides optimization support for those functions. PTS has been developed as a custom system for a company in the building products industry by Dr. Peter L. Jackson, a professor in the School of Operations Research and Industrial Engineering, Cornell University.

A. Who Should Read This Manual?

This manual makes reference to individuals with three levels of software expertise or "need-to-know" requirements. A PTS Developer is someone who must understand both the theory and the source code of the system in order to make fundamental changes to the software. A PTS Administrator is someone who is responsible for maintaining backups and the many links between PTS and the other software systems with which PTS must coexist. The PTS Administrator may also be called upon to extend PTS capabilities by creating queries, external macros and reports that can be called from within the PTS user interface program. The PTS Administrator needs to know the hooks into PTS but does not require a detailed knowledge of the source code. A PTS User is someone who must use PTS to perform the production transportation scheduling function that PTS is designed to support. This individual requires minimal knowledge of the structure and inner workings of the system.

This manual is intended primarily for the PTS Developer and the PTS Administrator but the PTS User may find material of interest here as well. The PTS Administrator's Manual documents procedures of interest only to an administrator. The PTS User's Manual is devoted to explaining the functionality of the software from a user's perspective.
B. The Role of PTS in Enterprise Management

The sponsoring company is adopting an Enterprise Data Management approach to its business systems. PTS has been designed to fit as a module in such a system. Figure I.1 depicts the scope that a typical Enterprise Data Management view would take in integrating various business functions.\(^1\) PTS provides support in the particular areas of dispatching and capacity management (the shaded regions in this figure).

![Diagram of Enterprise Management View]

**Figure I.1 Enterprise Management View**

Figure I.2 provides an expanded view of the business functions “Sales/Dispatch” and “Capacity Management/Production Control.” In this view, there are several

operational activities that can have significant impact on the profitability of the enterprise. The highlighted activities in “Sales/Dispatch” are Flag Formation, Carrier Choice, Trip Itinerary, and Shipping Schedule. The highlighted activities in “Capacity Management/Production Control” are Production Scheduling, Workcenter Scheduling, and Routing. Each of these activities requires a skilled scheduler to assess the economic tradeoffs in customer service, truck utilization, and workforce utilization and make decisions that balance these tradeoffs.

In many cases, the scheduling decisions of Figure I.2 cannot be made in isolation from one another. For example, in one plant, flag formation cannot be considered separately from carrier choice because the operational constraints of using a fleet truck (owned by the building products company) are substantially different from using a contract carrier vehicle. The maximum number of drops using a contract carrier is very small whereas a fleet truck has greater freedom to make multiple drops. Consequently, the flags (trips) must be formed with a particular carrier type in mind and the scheduler must look at each drop and decide whether to include it on a fleet truck trip or on some form of contract or common carrier.

Similarly, the trip itinerary must sometimes be considered together with the flag formation activity because, otherwise, some trips may turn out to be infeasibly long.

This same plant has another example in which scheduling decisions are linked. At that plant, due to the make-to-order strategy employed, customer orders are assigned to flags and flags are assigned to production shifts. No customer orders are satisfied out of inventory. Consequently, the shipping schedule and the production schedule are essentially identical for that plant. For the most part, production scheduling is performed by the transportation dispatch scheduler.
**Figure I.2 Enterprise Management Sales, Dispatch, Capacity Management and Production Control**

Figure I.3 illustrates the conceptual organization of these critical scheduling activities into separate software systems. The first system, now implemented as PTS, the Production Transportation Scheduling System, covers the decisions of assigning orders to flags, selecting a carrier, sequencing the stops on a flag, and assigning flags to days (that is, creating a production schedule). The second system, not yet implemented, would handle more detailed production control decisions such as assigning flags to shifts and sequencing production operations. The second system, when eventually implemented, could be a customized extension of PTS or it could be a commercial package that is adapted to the company environment.

PTS is unique in that it provides optimization support for all of the shaded functions in Figure I.3. Commercial packages, such as PCMiler, typically provide optimization for only a subset of these functions. Furthermore, three of the functions are tightly integrated: Order to Flags, Stop Sequence, and Carrier Choice. That is, the
optimization support specifically looks at the complex interactions inherent in these three functions and searches for solutions to optimize the tradeoffs. The fourth function, Flags to Days, is a separate optimization that attempts to load available production capacity with flags that create the least delay in customer service. It is this fourth function, Flags to Days, that is ignored by most commercial transportation planning packages.

Figure I.3 The Production Transportation Scheduling System Role in Enterprise Management

C. Overview of this Manual

This manual is organized as follows. In the next section, called “System Functionality,” we review the basic functional process that PTS is designed to support and we describe the process of connecting PTS with the Enterprise Data Management system. We also highlight the features of PTS which support decision making. The following section is entitled “System Optimization,” and provides an overview of the structure of the optimization approaches used in PTS. The final section is called “System
Components” and describes the organization of the software from a developer’s perspective. Appendices provide programmer level details in the form of lists that map specific functions to particular files in the project.
II. **SYSTEM FUNCTIONALITY**

In this section, we review the basic functional process that PTS supports (section A “The PTS Functional Process”), we describe the process of connecting PTS with the Enterprise Data Management system (sections B “Building the PTS Database” and D “Exporting the Schedule”) and we highlight the features of PTS which support decision making (section C “Building the Schedule”).

A. **The PTS Functional Process**

The PTS Functional Process can be viewed as a six-step process as illustrated in Figure II.1. This process would be repeated on at least a daily basis.

![Diagram of the PTS Functional Process]

**Figure II.1 The PTS Functional Process**

- **A1. Download Current Orders.** The PTS scheduling process takes place on a copy of the current order file and not on the actual order file itself. This copy must be downloaded from the mainframe database system at the beginning of
the scheduling process. This portion of the process is not part of PTS and is maintained by the company IT department.

- **A2. Build PTS Database.** The PTS database is rebuilt whenever a new order file is available (the user determines the timing of this build by running a macro from the PTS program window). Only a few tables are maintained locally by the PTS user. These include tables not currently available on the mainframe system such as allowable flag names, carrier names, and typical locations from which to pool customer orders.

- **A3. Form Loads and Sequence Stops.** This function is performed by the PTS user (the transportation scheduler) using the PTS program. The features of PTS which support this function are described in Section C.

- **A4. Assign Flags to Production Days.** This function is performed by the PTS user (the transportation scheduler) using the PTS program. The features of PTS which support this function are described in Section C.

- **A5. Export Revised Order File.** When the PTS user has completed a set of flags and wishes to send this schedule to the mainframe, he or she launches a macro from the PTS program to export the schedule. This macro outputs a file of orders with flag names, stop numbers, production dates, shift assignments, carrier names, and pooling indicators.

- **A6. Upload Revised Orders.** The revised order file must be uploaded and integrated into the mainframe database. This portion of the process is not part of PTS and is maintained by the company IT department.
B. Building the PTS Database

The process of building the PTS Database is fully automated and implemented as a macro within a Microsoft Access database. The PTS user launches this macro from within the PTS program. This macro is found in the database file PTSBuild.mdb.

At the point in time at which the company adopts a new mainframe database system for Enterprise Data Management, this macro would have to be modified to adapt to a new format order file that would be downloaded from the mainframe. In particular, note that the current order file that is downloaded is not Year 2000 compliant. Consequently, this macro is not Year 2000 compliant in its current form. Observe that it is likely that this conversion can be accomplished without any changes to the PTS program.

The Build PTS Database macro can be viewed as a seven step process as illustrated in Figure II.2.
Figure II.2 Build PTS Database Process (A2)

- A21. Reset Data Files. Microsoft Access database files are unreliable because they grow in size with changes to the database and they frequently become corrupted. For this reason, the PTS database is deleted and completely reconstructed each time the Build PTS Database macro is run. Step A21 deletes the temporary database files (PTSData.mdb, PTSLink.mdb, and PTSCopy.mdb) and replaces them with three blank databases with the same names.

There are two permanent databases in PTS: PTSBuild.mdb (which stores the table formats and macros needed to build PTSData) and PTSMain.mdb (which stores the tables containing user-supplied data such as carrier names). These files are unlikely to become corrupted because changes to them are minor and infrequent. However, copies of these databases in their original state are stored
in the Backup directory. The user or administrator should make periodic
backups of the PTSMain database to preserve the user-supplied data.
Note that this step destroys any work by the PTS user performed in the PTSData
database. In particular, any loads that were created or flags that were
scheduled are lost in this step unless they were previously exported to the
mainframe (A5, A6).

- **A22. Reset PTSLink.** The PTSLink database is blank as a result of step A21.
  This step copies templates of the tables for the PTSLink database from
  PTSBuild into PTSLink. The tables are initially empty. These templates,
  which are stored in PTSbuild, would have to be modified if the format of the
download files changes.

- **A23. Import Download Files.** The order and line detail file and the notes file
  that are downloaded from the mainframe are in text format. This step imports
  them into tables in PTSLink.

- **A24. Reset PTSData.** The PTSData database is blank as a result of step A21.
  This step copies templates of the tables for the PTSData database from
  PTSBuild into PTSData. The tables are initially empty.

- **A25. Export PTSDataQueries.** The PTS program makes use of a large
  number of pre-formatted queries. These queries are stored in PTSBuild for
  safekeeping and copied to the PTSData database in this step. These queries are
  an essential part of PTS and should not be modified without a thorough
  knowledge of how the PTS program makes use of them.

- **A26. Make PTSData Tables.** In this step, the database PTSBuild makes links
to PTSLink, PTSData, and PTSMain and executes action queries that populate
all the tables in PTSData.
The queries that populate TableOrderHead and TableLineDetail would have to be modified when the format of the download files change. In particular, these queries are not Year 2000 compliant because the download files currently use two-digit year fields.

There is one peculiarity to the structure of PTSData. TransCAD is unable to read Microsoft Access databases. For this reason, we make a copy of TableOrderHead (the list of orders) and export it as a dBase IV format file (maporder.dbf) which TransCAD can read. PTSData is attached to this file so both the PTS program and TransCAD can make changes to the file. This adds some complexity to the PTS program since it must coordinate selections of orders in the MapOrder table and in TableOrderHead.

- **A27. Detach PTSBuild Tables.** In step A26, the PTSBuild database made links to tables in PTSData, PTSLink, and PTSMain. In this step, those links are deleted, returning PTSBuild to its original configuration. If for some reason this step fails, the administrator can manually delete the links or simply restore the PTSBuild.mdb file from the backup directory.

### C. Developing the Schedule

The process of developing a schedule (steps A3 and A4) is described in greater detail in the PTS User’s Manual. In this section, we will highlight some of the features of PTS that facilitate this process.

- **A3. Form Loads and Sequence Stops.** This function is performed by the PTS user (the transportation scheduler) using the PTS program. The PTS program provides many user support services:
  - line detail reports synchronized with order detail reports,
  - customizable queries to select orders for view,
  - sorting functions,
• highlighting functions and queries,
• subselection extraction functions,
• pooling functions (assigning orders to pooling locations),
• geographical (map) views of order location by zipcode,
• mapping functions (zoom, pan, find zipcode),
• order selection by zipcode functions (select rectangle or lasso), and
• subselection extraction and merge by zipcode functions.

In addition to these viewing and grouping functions, the PTS program provides two optimization functions in support of the A3 process:

• **Stop Sequence Optimization** (Optimization 1): this optimization takes a selection of orders identified by the user and sequences them according to zipcode to minimize the length of the trip from the depot. The user can add constraints to force certain stops to be visited in a particular sequence. The result of this operation is the creation of an unscheduled flag (or load): a group of orders with a common identifier, sequenced by zipcode for drops.

• **Flag Formation Optimization** (Optimization 2): this optimization takes a selection of orders identified by the user and organizes it into separate unscheduled flags (loads) to minimize the transportation cost of delivering all of the orders. In addition to creating loads, this algorithm recommends a carrier choice for the flag (not a specific carrier: the choices are limited to contract carrier, fleet long truck, or fleet short truck). Included in the transportation cost are penalties for exceeding the capacity of the chosen carrier, the maximum number of drops for the chosen carrier, and the maximum mileage for the chosen carrier. These capacities and penalties are user-specified. This
optimization also performs stop sequence optimization on each of the unscheduled flags that it creates.

- **A4. Assign Flags to Production Days.** This function is performed by the PTS user (the transportation scheduler) using the PTS program. The flags which have been scheduled are displayed in a tree structured list, organized by production day and shift. The unscheduled flags are displayed in a sortable list. The user drags unscheduled flags and drops them into the tree structured list on the day and shift desired. A dialog box prompts the user to supply a flag name, a particular carrier name, and a pick-up date from drop down lists. Scheduled flags can be unscheduled by dragging them and dropping them on the unscheduled list. In addition to these graphical tools, PTS supports this process in two ways:

1. PTS displays totals and subtotals of production volumes, measured in linear feet, of various selections of scheduled and unscheduled flags. These totals are analyzed by scheduling type and displayed in both graphical and tabular formats, and

2. PTS provides a **Production Smoothing Optimization** (Optimization 3) to recommend production dates for each unscheduled flag. This optimization takes the list of unscheduled flags, prioritizes them by the oldest order on each load, and recommends a specific production day for each load. This optimization is designed to minimize customer waiting time subject to daily production capacities specified for each scheduling type. The list of unscheduled flags can then be sorted by these recommended production dates to simplify the A4 process of dragging and dropping flags.
D. Exporting the Schedule

The process of exporting the schedule is fully automated and implemented as a macro within a Microsoft Access database. The PTS user launches this macro from within the PTS program. This macro is found in the database file PTSBuild.mdb.

At the point in time at which the company adopts a new mainframe database system for Enterprise Data Management, this macro would have to be modified to adapt to a new format file that would be uploaded from the mainframe. Observe that it is likely that this conversion can be accomplished without any changes to the PTS program.

The export process is a simple three-step process as illustrated in Figure II.3.

![Diagram](image)

**Figure II.3 Export Revised Order File (A5)**

- **A51. Copy Table Format to PTSLink.** The database PTSLink was initially blank as a result of step A21. This step copies a template of the export table from PTSBuild to PTSLink.

- **A52. Transfer Revised Order Data to PTSLink.** This step populates the export table in PTSLink using the revised order data found in TableOrderHead in PTSData.
• **A53. Export Text File for Upload.** This step exports the export table in PTSLink as a text file. At this point, it becomes the responsibility of the the A6 process to upload this file and update the mainframe database.
III. **SYSTEM OPTIMIZATION**

This section provides an overview of the structure of the optimization approaches used in PTS. As indicated in the previous section, there are three optimizations (stop sequence, flag formation, and production smoothing) provided within PTS. Since stop sequence optimization is performed as a subroutine within flag formation optimization, we describe flag formation first.

A. **Flag Formation Optimization**

Flag Formation Optimization can be seen as a four step process in which the third step is iterated. Figure III.1 illustrates.

![Diagram of Flag Formation Process]

**Figure III.1 Flag Formation Process**

- **F1. Form Single Flag.** The first step in the process is to group orders by zipcode and then sequence these zipcode stops into a single flag.

- **F2. Break Flag into Feasible Flags.** The flag created in step F1 will, in many cases, be infeasible: it will exceed the maximum number of stops allowed for a carrier, or it will exceed the maximum mileage, or the total shipping units.
will exceed the capacity of any trailer. This step breaks the flag into smaller flags that are feasible, if that is possible. In doing this, it minimizes the total transportation cost of the resulting flags subject to the constraint that the sequence of stops on any of the smaller flags is simply a subsequence of the stops on the original single flag. This constraint prevents this step from finding the truly optimal set of feasible flags. It results in a partial optimization.

This step uses an operations research technique known as dynamic programming to perform the (partial) optimization and it is extremely fast. On each of the flags that are created in this step, we optimize the stop sequence using the same sequencing algorithm as in F12.

- **F3. Improve Set of Flags.** The flags created in step F2 are likely to be feasible but they are not likely to be optimal. Step F3 is an iterative step that groups and regroups the orders into different flags seeking to find the solution with the lowest transportation cost. It does this by combining nearby flags and then finding the best way to break them into smaller feasible flags. After this algorithm has considered every single flag created in F2, it terminates. This approach gives much better solutions than step F2 alone, but we cannot guarantee that it will find truly optimal solutions. An experienced user may be able to spot improvements to the resulting solution. For this reason, PTS has extensive tools for editing solutions.

- **F4. Save Improved Flags.** In this step, the order file is updated with the information that indicates what flag each selected order is on, and what stop on the flag delivers the order. Each flag is identified by a unique number (flag ID or flag identifier) and each order on the flag bears that number. These flag identifiers are not the same as the flag codes assigned by the user when
dragging an unscheduled flag and dropping it on the schedule (step A4). PTS does not assign such flag codes during the flag formation process. Instead, it identifies a flag using an order number chosen randomly from the orders that are assigned to the flag. Since an order can be on only one flag, this randomly chosen order number will uniquely identify the flag. So, the flag ID matches the order number for one of the orders on the flag. Every order in the flag is given the same flag ID.

The process of forming orders into a single flag (F1) is a two step process as illustrated in Figure III.2.

![Diagram of flag formation process]

---

**Figure III.2 Form Single Flag Process (F1)**

- **F11. Group Orders By Zipcode.** The first step is to take all the orders selected by the user using the various selection processes mentioned when describing A3, and group them by zipcode. In the process of grouping them by zipcode, this step calculates a summary total of the number of shipping units in this selection in this zipcode. This step also matches the zipcode with a zipcode in the ZIPSUSED database. The ZIPSUSED database stores the geographical location of each zipcode (latitude, longitude, and nearest highway intersection). Each group of orders with summary totals and zipcode
location is called a node or a stop. With this information, PTS can compute
the distance between stops and it can compute the total shipping units across
multiple stops. The list of nodes with summary data is used at multiple points
in the process (F2, F3).

- **F12. Sequence Flag.** The second step is to take the list of nodes and sequence
  them into a single trip starting from the depot. (The zipcode location of the
depot is part of the PTSMAIN Economic Parameters table.) The algorithms
used to sequence stops on a flag are listed in the next section.

The process of improving a set of flags (F3) is a six-step iterative process as
illustrated in Figure III.3.
- **F31. Sort Flags by Azimuth.** The improve-flags process maintains a list of current flags sorted in order of the azimuth of the flag centroid relative to the depot (the azimuth of a point is the angle from true North made by a line connecting the depot to the point; we choose the centroid of the nodes of a flag to represent the location of the flag). Thus, the flag with centroid that is Northeast of the depot but nearly due North of the depot (azimuth close to zero) is the first flag in the list. The last flag on the list is the flag with centroid that is Northwest of the depot but nearly due North (i.e. azimuth close to 360°).

- **F32. Pick Next Flag.** This step keeps track of the current azimuth being processed and progressively increases the current azimuth from 0° to 360°. The next flag to process is the flag whose centroid has the smallest azimuth greater than the current azimuth. Processing flags in order of their azimuth is called ‘sweeping’ because you are sweeping around the compass as you go. Note that the list of flags can be dynamic (flags can be added or deleted) but the sweeping process will not get trapped in a loop because the current azimuth increases by a positive amount on each iteration and there are only a finite number of possible flags that could be added. The iterative process terminates when the current azimuth exceeds 360°. (We have experimented with sweeping more than once around the compass but the marginal benefits in solution quality do not appear to justify the additional processing time requirements.)

- **F33. Pick a Nearby Flag.** Steps F33-F36 are repeated for each of several nearby flags. A flag is ‘nearby’ if its azimuth is larger but close to the azimuth of the flag being considered. This step picks the nearby flags in sequence (by azimuth) and performs steps F34-F36 for each one. The number of nearby
flags to treat has been determined through experimentation and is not subject to change.

- **F34. Combine the Two Flags.** This step takes the flag being considered (from F32) and a nearby flag (from F33) and combines the pair into a single flag. The stops on this single flag are sequenced to minimize the length of the trip (using the stop sequence algorithm of step F12).

- **F35. Break Combined Flag into Feasible Flags.** This step optimizes the transportation cost of serving all the stops in the combined flag using multiple flags subject to the constraint that the sequence of stops on any of the smaller flags is simply a subsequence of the stops on the original combined flag. This step is identical to step F2 and uses the same fast dynamic programming algorithm to solve the optimization.

- **F36. Replace Original Two Flags if Improvement.** Combining the two flags in step F34 may not lead to an improvement in transportation cost. This step adds up the transportation cost of the new flags that resulted from step F35 and compares that with the sum of the transportation cost of the original two flags that were combined in step F34. If the new flags are better than the old flags, then the old flags are removed from the list of current flags and the new flags are added.

**B. Stop Sequence Optimization**

The functional step F12 (Sequence Flag) is a critical optimization step in PTS. It is used not only when optimizing a single flag chosen by the user but it is also used as a subroutine in the flag formation optimization (F2 and F35 use F12). PTS may be required to solve hundreds of these stop sequencing problems in a single user session, so the algorithms chosen to solve it must be fast and must yield good solutions. This sequencing problem is called the Travelling Salesman Problem. It is a well-studied problem in the
literature. We have chosen to implement the algorithm known as “Farthest Insertion followed by Two-Opt.”

The principle behind Two-Opt is simple. A sequence of stops that results in arcs that cross themselves, as in the example of Figure III.4, can typically be improved by replacing the crossing arcs with two non-crossing arcs (the dashed lines in the figure). Observe that if A-B is replaced by A-C, then we have no choice but to replace C-D with B-D. Consequently, it is enough to look at an arc such as A-B and consider all arcs that are not in the tour but are connected to either A or B (i.e. all possible replacement arcs) and are shorter than A-B. If the tour can be improved by swapping A-B with any of these replacement arcs, then we would make the swap.

![Diagram of arc swapping](image)

**Figure III.4 Arc Swapping (Stop Sequence Optimization)**

A tour is said to be “two-optimal” if no such swaps can be found to improve the tour. The algorithm to search for pairs of arcs to swap is called the “Two-Opt” method of solving the Travelling Salesman Problem. Figure III.5 illustrates the logic.
Figure III.5 Sequence Flag (F12)

- **F121. Sequence Nodes Using Farthest Insertion.** Generate an initial tour of the stops by sequencing the nodes using the "Farthest Insertion" technique. This is a simple procedure: given a partial tour, find the stop that is the furthest distance from any stop that is in the partial tour. That stop is the next to add to the tour. Find the best place to insert it (either before or after the stop that it is closest to in the partial tour) and continue until all stops have been added to the tour. The resulting tour typically is good but it will exhibit many crossing arcs which can be improved by steps F122-F124.

- **F122. Consider Each Arc in Tour.** Every time we are presented with a new or improved tour, we must run through steps F122-F124. The first step is to loop over every arc in the tour. For each one of these arcs, use steps F123-F124 to search for a replacement arc for this that will improve the tour. If no improvement can be found, for any of the arcs in the tour, then the tour is said to be "Two-Optimal". This is the tour that is passed to step F125.
- **F123. Consider Each Possible Shorter Replacement Arc.** In Figure III.4 we would consider every arc connected to A that is not in the tour and every arc connected to B that is not in the tour. Reject any arcs that are not shorter than A-B. Each one of these arcs is a potential replacement arc that could improve the tour.

- **F124. Swap Arcs if Tour Improves.** Given an arc and a replacement arc (a replacement arc shares one stop in common with the arc), find the other two arcs that must be swapped and swap the two arcs in the tour with the two arcs out of the tour (as in Figure III.4). If the resulting tour is shorter than keep the swap and start step F122 again with an improved tour. Otherwise, return to steps F122 and F123 to continue the search for arcs.

- **F125. Choose Best Tour Direction.** Every flag includes the depot as its first stop. The direction of the tour will determine which customer is served first. In this step, we pick the closest of the two customers that are connected to the depot in the tour and make the direction of that customer the direction of the tour.
C. Production Smoothing Optimization

Figure III.5 illustrates the general process of production smoothing conducted by the PTS.

**Figure III.6 Production Smoothing Optimization**

- **P1. Extract Data for Optimization.** The production smoothing optimization requires its input data to be presented in tabular form. This step represents the process of preparing these tables.

- **P2. Solve Optimization Problem.** This function solves the optimization problem that is described below ("A Mathematical Formulation of the Production Smoothing Problem").

- **P3. Apply Optimized Schedule Dates to Selected Flags.** The solution to the optimization problem is received in tabular form. This step represents the process of transferring that data back into the PTS database and updating the selected flags with the schedule dates suggested by the optimization.
A Mathematical Formulation of the Production Smoothing Problem

Index Sets

S: the set of all schedule types;
D: the set of all production days;
F: the set of all selected flags;

Parameters

M: a large number;

\( p_{fd} \): delay penalty for scheduling flag \( f \) on day \( d \); (we assume that the penalties increasing in days: \( p_{fd} > p_{fd'} \) if \( d > d' \))

a\( s \): production (in linear feet) of schedule type \( s \) required by flag \( f \);

l\( sd \): minimum desired production of schedule type \( s \) on day \( d \);

u\( sd \): maximum desired production of schedule type \( s \) on day \( d \);

Making the connection with Figure III.6, the data (\( \{ l_{sd} \}, \{ u_{sd} \} \)) correspond to “Production Targets (Min/Max) by Schedule Type by Day,” the data (\( \{ p_{fd} \} \)) correspond to “Delay Penalty by Flag by Day,” and the data (\( \{ a_{s} \} \)) correspond to “Production Load by Schedule Type by Flag.”

Variables

\( X_{fd} = 1 \) if flag \( f \) is assigned to day \( d \); 0 otherwise;

O\( sd \): excess production (in linear feet) of schedule type \( s \) on day \( d \);

U\( sd \): under-production (in linear feet) of schedule type \( s \) on day \( d \);

Production Smoothing Optimization Problem

Choose (\( \{ X_{fd} \}, \{ O_{sd} \}, \{ U_{sd} \} \)) to minimize \( Z \):

\[
Z = \sum_{f \in F} \sum_{d \in D} p_{fd} X_{fd} + \sum_{s \in S} \sum_{d \in D} M(O_{sd} + U_{sd})
\]

subject to:
The Production Transportation Scheduling System

\[ l_{sd} \leq \sum_{f \in F} a_{pf}X_{fd} - O_{sd} + U_{sd} \leq u_{sd} \text{ for all } s \in S, d \in D; \]

\[ X_{fd} \in \{0,1\} \text{ for all } f \in F, d \in D; \]

\[ O_{sd} \geq 0 \text{ for all } s \in S, d \in D; \text{ and } \]

\[ U_{sd} \geq 0 \text{ for all } s \in S, d \in D. \]

The optimization problem is to minimize a sum of penalties: the penalties of assigning flags to days (earlier days have lower penalties than later days) and the penalties of exceeding or undershooting the daily production targets. The constraints are such that in an optimal solution, we will have:

\[ O_{sd} = \max \left\{ 0, \sum_{f \in F} a_{pf}X_{fd} - u_{sd} \right\} \]

and

\[ U_{sd} = \max \left\{ 0, l_{sd} - \sum_{f \in F} a_{pf}X_{fd} \right\}, \]

which have the desired interpretation of overproduction and underproduction, respectively. The restriction of the assignment variables to \{0,1\} makes the problem difficult: it is now a 0-1 Programming problem. We elect to solve this problem using the optimization package XA by Sunset Software. By design, the problem is always feasible but the number of combinations to consider is so great that XA may terminate with a fractional solution to some of the variables. We rely on post-processing (P3) to round off these fractions.
IV. SYSTEM COMPONENTS

This section describes the organization of the software from a developer's perspective. The functional description of PTS in Section III was not concerned with implementation details. It attempted to present the functions of PTS in a uniform manner. The implementation of these PTS functions, however, is not at all uniform. They are implemented in a variety of languages: SQL queries, Microsoft Access macros, Visual Basic forms and modules, and TransCAD macros. The implementations, in all cases, make use of commercial software packages to complete their functionality.

The choice of implementation language for each function was based on whatever software package was best suited for the task. Accessibility for maintenance was also a consideration. For example, all of the Microsoft Access macros could be replaced with Visual Basic code. The functionality of these macros would then be hidden within compiled code. For ease of maintenance, we chose to implement these functions in external macros which are easily understood and modified. The advantage is that changes to the external database system are not likely to force a recompilation of the Visual Basic code: the changes can likely be accommodated with changes to the macros and queries that are external to the compiled code.

The most complex functions are implemented in the TransCAD macro language. This language is superior to Visual Basic for implementing mathematical algorithms with advanced data structures. However, TransCAD requires considerable programming sophistication ("a+b" means different operations depending on the data structures used to store "a" and "b": the "+") could mean either addition or concatenation). In general, our order of preference for implementing functions was SQL, Access macros, Visual Basic modules, and TransCAD macros.
Figure IV.1 gives an overview of the various system components that make up PTS. The fact that it takes so many components to make up the system is a reflection of the complexity of implementing a graphical decision support system with optimization.

Figure IV.1 PTS System Components

- **PTS User Interface (Visual Basic).** The front-end to PTS is a Visual Basic program with multiple windows providing different views of the database and menus, buttons, and dialogs to execute the various PTS functions. Some of the functionality of PTS resides in the Visual Basic modules that make up this User Interface program. Other functionality lies outside this program in various external macros and queries that can be called by the PTS User Interface. The current version of PTS is compiled using Visual Basic 5.0. The PTS Developer requires a commercial license for Visual Basic 5.0 to compile the PTS User Interface. No license is needed to run the compiled code.

- **3rd Party Components (True DBGrid, Pinnacle Graph, Anibutton).** One of the strengths of Visual Basic is the wide variety of 3rd party software
components that can be included in a project with programming ease. The PTS User Interface makes use of several of these components: a database-aware grid (for tabular views of SQL query results), a graph (for chart views of SQL query results) and an animated button (to display multistate actions). The PTS Developer requires a commercial license for each of these packages in order to compile the Visual Basic User Interface. No license is needed to run the compiled code.

- **TransCAD Server.** TransCAD is a powerful geographical information and programming system that provides a variety of mapping services. TransCAD is used by PTS to display maps within a map window of the User Interface. It is also used to execute macros which implement the flag formation and stop sequence optimization functions. PTS also makes use of certain geographical files that are supplied with the TransCAD package. A TransCAD license is needed to run the server. That is, TransCAD must be installed on the user’s machine under separate license from the Caliper Corporation.

- **PTS Mapping Macros (TransCAD UI Database).** The PTS Mapping macros implement the flag formation and stop sequence optimization functions. These macros also make calls to TransCAD functions to draw maps and to interact with geographical databases (databases with latitude and longitude and highway information). These macros are compiled into a TransCAD UI database which is accessed by the TransCAD server. The PTS Developer requires a TransCAD license to compile these macros. The PTS User requires a TransCAD license to run the server and execute the macros. These macros are called from within the PTS User Interface program in response to various menu and button choices. Thus, TransCAD is an essential commercial package with a separate license required for each user installation.
- **Microsoft Jet Database Engine.** Visual Basic interacts with relational databases by means of the Microsoft Jet Database Engine. This permits a high degree of synergy between the procedural language of Visual Basic and the non-procedural language of SQL. The Microsoft Jet Database Engine is included with the Visual Basic programming language so no separate license is required to compile code that uses this engine. No license is needed to run compiled code that use this engine.

- **Microsoft Access.** Much of PTS functionality is implemented in SQL queries and macros and stored in Microsoft Access databases. The PTS Developer requires a licensed copy of Microsoft Access in order to modify any of the queries or macros stored in databases. The PTS User requires a licensed copy of Microsoft Access to execute the macros. PTS currently requires version 2.0 of Microsoft Access.

- **Microsoft Access SQL queries.** SQL is now the preferred way to interact with datasets stored in external files. Nearly all PTS input/output is handled through SQL queries. Most of these queries are stored in Microsoft Access databases (PTSData, primarily). Others are created dynamically within the PTS User Interface program and executed on the fly. The PTS Developer requires a licensed copy of Microsoft Access in order to modify any of the queries stored in databases.

- **Microsoft Access Database Macros.** As discussed above, macros are unnecessary when the same functionality can be accomplished in Visual Basic code. However, to maximize the separation of database maintenance from user interface code, many PTS functions are implemented as Access macros stored in Microsoft Access databases. In particular, steps A2 and A5 in the PTS functional process are implemented completely using Access macros.
- **LPRun Optimization.** LPRun is a standalone program that solves optimization problems. The PTS User Interface program calls LPRun to solve the Production Smoothing Optimization (Optimization 3). Once called, LPRun reads the optimization problem description from a database file called PTSOpt, it reads the data for the problem from the database file PTSData, and it writes the solution back into the database file PTSData. LPRun was developed by Professor Peter Jackson of Cornell University.

- **XA Optimization Library.** LPRun solves the optimization problem by calling routines from the XA Optimization Library, a commercial package by Sunset Software. The PTS Developer and User both require a license to use XA. That is, XA must be installed under separate license from Sunset Software on any machine running PTS.

### A. PTS Directory Structure

The purpose of this section is to describe the directory structure used by the PTS software. The commercial software packages that require separate licenses (XA DLL, TransCAD, Microsoft Access 2.0, and Microsoft Visual Basic 5.0) are supplied with their own installation programs. These setup programs create their own directory structures on the hard drives where they are installed. Refer to the documentation for these packages to learn their implementation details. In this section, we describe the directory structure created by the custom setup program provided to install PTS.

Figure IV.2 is a graphical depiction of the file directory structure used by PTS.
PTS. This is the top level directory. It contains the executable file for the user interface to PTS (PTS.exe), an initialization file (PTS.ini) and the following subdirectories.
• **AS400Links.** The text files that are downloaded from the mainframe or uploaded to the mainframe should be stored in this directory.

• **Data.** The main PTS databases are located in this directory with backup in DataSource. The main PTS databases are as follows:
  - **PTSBuild.mdb.** This database holds all the table templates, queries and macros necessary to rebuild the PTSData.mdb database from the download files. It does not contain any data that requires updating. The one exception is that it does contain a table called TableDataPaths which lists the locations of several directories. However, this table is automatically updated by PTS.exe using data found in PTSMain.mdb. Therefore, it should never change from the time of its first installation. A backup of this file is stored in the directory Datasource.
  - **PTSMain.mdb.** This database holds several developer tables that describe the structure of tables for display in the PTS user interface. It also contains several tables of data maintained by the PTS user, including allowable flag names, carrier names, economic parameters, the depot location, and pooling zipcodes. These data will change over time so the PTS User or Administrator should make a backup copy of this file periodically. The original version of this file is stored in the directory Datasource.
  - **PTSOpt.mdb.** This database holds tables which describe the production smoothing optimization problem in a form readable by LPRUN. Refer to the documentation of LPSQL (optimization modeling software by Professor Peter Jackson) for insights into the function and structure of this file. The data in PTSOpt.mdb will not
change over time: it is a static database. A backup of this file can be found in the Datasource directory.

- PTSData.mdb. PTSData is the database that the PTS User interacts with. This database holds the current order file and stores the results of the PTS User’s scheduling activities until these results are uploaded to the mainframe. This database is constructed from the order files that are downloaded from the mainframe. This database is completely reconstructed whenever the Rebuild Database macro is executed. Consequently, it is not necessary to keep a backup of this file.

- PTSLink.mdb. PTSLink is a temporary database that is used during the construction of PTSData. It is used to import the downloaded text files and format them into Access tables. It is also used during the upload process. This database is completely reconstructed whenever the Rebuild Database macro is executed. Consequently, it is not necessary to keep a backup of this file.

- PTSCopy.mdb. This is temporary database into which the PTS user can export data from the various windows. If the PTS user requires custom reports or printouts not provided in the existing system, the PTS Administrator could design these reports and copy them into this database together with macros to control the printing. PTS makes no use of this file except as a destination to export data and call user specified macros. This mechanism provides an easy way to extend the capabilities of PTS. This database is created anew whenever the Rebuild Database macro is executed. If the PTS Administrator designs macros or reports to be stored in this database, these macros and
reports must reside in a different permanent database under the control of the PTS Administrator.

- **DataSource.** This directory is used to hold backups of critical files. In particular, it holds a backup for PTSBuild.mdb, PTSMain.mdb, and PTSOpt.mdb. Note that there are no backup or restore functions built into PTS. The PTS User and PTS Administrator are responsible to make backups periodically of critical files.

- **GeoData.** PTS makes use of several geographical files that are supplied with the TransCAD package on the CD-ROM entitled “Geographical Files.” These files include city and state locations, and 5 digit zipcode locations. Rather than require the PTS User to always use this CD-ROM, we recommend that those files be copied to this directory. This copy step is not included as part of the setup installation program. The PTS Administrator must perform this step manually. A DOS batch file is provided in this directory that will perform the copy function. Some modification to the batch file may be required depending on the drive labels in use. There is no need to backup the contents of this directory since the CD-ROM holds the original to all of these files.

- **Icons.** This directory is a copy of the Icons directory supplied with Visual Basic. The PTS Administrator is free to use any of these icons to customize macros as they appear within the PTS User Interface program. No backup is necessary.

- **LPSQL.** The optimization program LPRUN.exe and its initialization file (LPEDIT.ini) are stored in this directory. There are no data in this directory that require backup.
• **MapData.** This directory contains geographical files that were created specifically for PTS: the highway structure of the United States and the list of zipcodes actually used by the company’s customers. This list of zipcodes used grows over time as new customers are acquired. Consequently, the PTS Administrator should arrange to make periodic backups of the files in this directory.

• **TCIface.** This directory contains the compiled UI database of TransCAD macros developed specifically for PTS. This is a static database. It can be reinstalled using the PTS setup program or it can be recompiled using the TransCAD resource compiler from source files in TCSource.

• **TCSource.** This directory contains the source code for the TransCAD Mapping Macros developed specifically for PTS. It also contains a DOS batch file for calling the TransCAD resouce compiler to reconstruct the UI database in TCIface. These files should be modified only by a PTS Developer. The originals of these files are maintained by Professor Peter Jackson.

• **Temp.** Temporary data files are stored in this directory. These files are recreated each time the Rebuild Database macro is executed. No backup is necessary.

• **Templats.** Templates for temporary Dbase IV-type data files are stored in this directory. These template files are static. They can be re-copied to this directory by re-running the PTS installation program.

• **VBSource and Bitmaps.** The Visual Basic project and source files for the PTS User Interface program are stored in these directories. These files are static. They can be re-copied to this directory by re-running the PTS installation program. These files should be modified only by a PTS Developer. The originals of these files are maintained by Professor Peter Jackson.
B. TransCAD Mapping Macros File Organization

As explained above, the PTS Mapping macros implement the flag formation and stop sequence optimization functions. These macros also make calls to TransCAD functions to draw maps and to interact with geographical databases (databases with latitude and longitude and highway information). These macros are compiled by the TransCAD Resource Compiler into a TransCAD UI database which is accessed by the TransCAD server. The source files (called “resource files”) are located in the TCSource directory. The compiled UI database files are located in the TCIface directory. Figure IV.3 illustrates.

![Diagram showing file organization]

**Figure IV.3 TransCAD Mapping Macros File Organization**

- **TCIface.lst.** This is not a macro file. This is a list of resource files supplied to the TransCAD resource compiler.
- **Intrface.rsc.** This is the top level macro file. TransCAD requires a macro called “INTERFACE” as the entry point to a UI database. This file contains that macro.
- **Global.rsc.** This file declares global variables and contains macros to read an initialization file containing directory locations of the geographical files. The
macro in this file calls the “build” macro to reconstruct the TransCAD workspace for PTS.

- **Build.rsc.** The build macro rebuilds the TransCAD workspace: it loads the map, the states, cities, and zipcodes. It also reads the customer orders, maps their zipcode locations and displays them on the map.

- **MapUtil.rsc.** This file contains a variety of useful macros for manipulating the map displayed in the PTS map window. It includes macros for zooming, panning, selecting orders, and displaying flags.

- **FormFlag.rsc.** This file contains the macros needed to implement the Flag Formation Optimization (Optimization 2). These macros call macros in SeqFlag.rsc.

- **SeqFlag.rsc.** This file contains macros to implement the Sequence Stops Optimization (Optimization 1) when the user has imposed constraints on the sequence. These macros call the TSP macros in Tsp.rsc.

- **Tsp.rsc.** This file contains macros to solve the Travelling Salesman Problem.

- **TCIface.dbd-TCIface.6.** This is the compiled UI database for TransCAD.
C. PTS User Interface Views

The main views available to the PTS User from the PTS User Interface are illustrated in Figure IV.4.

![Diagram of PTS User Interface Views]

**Figure IV.4 PTS User Interface Views**

- **Developer Tables, Administrator Tables, User Tables.** PTS has a generic table editor that is accessed by the menu item “View Tables.” The table editor window consists of a drop down list of available tables to select, a scrollable list of the key fields in the selected table from which the user can select a record, and a property list of editable fields for the selected record. Fields with value lists are editable via drop-down lists. It is worth noting that this window is data-driven so the PTS Administrator can add additional tables (to the PTSMain database) without any coding changes to the PTS User Interface program.
The list of tables available is short for the PTS User. It consists of the list of flags names available, the list of carrier names, the economic parameters to control flag formation, and the production smoothing target defaults. The list of tables available to the PTS Administrator includes all of the PTS User tables as well as a list of queries available from the menu, and a list of macros available in the Macro window. The PTS Administrator also has access to a table of file path names that specifies to PTS the location of critical files. The tables available to the PTS Developer include all of the preceding as well as the tables needed to define tables (elements, fields, table names, and folders). It is this latter set of tables that permits the Table window to be data-driven.

- **Macros.** This window is an iconic display of all the external macros that can be called from the PTS User Interface program. The user double-clicks on an icon (or selects from a list in an alternate view) in order to launch the macro. This screen is completely customizable by the PTS Administrator. The details of the commands to launch the macros and the icons to display are contained in the Administrator table “Macros” in the PTSMain database. Any external program can be launched using a macro. The Administrator can instruct PTS to display a “Wait” dialog box and go into Wait mode until a particular file is created, signalling the completion of the external process. The Macros Table can be edited using the generic table editor window described above.

We have limited our use of macros in PTS to launching Microsoft Access and executing macros within an Access database.

- **Query Design.** The query design window is a form designed to create basic SQL queries. It is capable of populating the SELECT clause with any number of fields, the WHERE clause with conditions, the FROM clause with up to two tables, the HAVING clause with one pair of matching fields, and the
ORDER BY clause with fields and sorting directions. This form is intended for use by the PTS Administrator to design special views of the customer order table (see Order Detail Window) or to provide custom reports (see Reports Window). The queries designed here can be stored automatically in the PTSMain database (in the Queries table). These queries are then available to the PTS User as menu items.

The PTS Administrator can bypass this query design editor and use the more powerful editor available in Microsoft Access. The resulting SQL query can then be copied and pasted into PTS.

If the query includes a question mark ("?") in the value field of a WHERE condition, then, when the query is executed from the Query menu, the user will first be prompted with a dialog box to fill in the missing value. Up to six question marks can be included in the same query. An example of this is the query that allows the user to look at orders from multiple zones. This feature is not implemented for custom reports.

- **Order Detail.** The Order Detail window is specialized to display only records from the Order table (called TableOrderHead). The field headings can be customized by the PTS Administrator (accessing the Fields table) and the arrangement and column width of the fields can be customized by the PTS User. Note that a number of fields have been designated by the PTS keyword. The PTS program and many built-in queries rely on the existence of these fields. The other fields are not essential (unless the PTS Administrator has constructed special queries and custom reports that use those fields). When it is time to transition to a new Enterprise database, it will be sufficient if only the PTS keyword fields are available in the new download.
The selection of orders displayed in this window changes frequently from various user actions. For example, clicking on a flag node in the Scheduled Flag outline list will cause all of its orders to be displayed. Clicking on a row of the Unscheduled Flag list will cause all of its orders to be displayed. Selecting one of the menu items from menu Query will launch a query to populate the Order Detail window. Finally, orders selected on the map can be copied to Order Detail window. Note that the orders themselves are permanently stored in the Order table: the user need not fear that orders have been lost when they disappear from Order Detail window. By repeating the query, the selection can be brought back into view.

The PTS User can sort the orders in this window by clicking on selections of column headings. The Order Detail window has highlighting tools that permit the user to extract convenient subsets of orders from the selected orders (either manually or through pre-programmed queries). Orders highlighted in the Order Detail window can be copied to the Map Window. The user can also launch the Flag Formation processes from this window.

- **Map.** The Map Window displays a map of the United States and Canada and shows the location of all orders in the Order table. Orders in the same zipcode are grouped together and displayed as a single point (the label displayed for the point is for only one of the orders at that point). Currently, the resolution of the order data is at the 5-digit zipcode level. The Map window supports many mapping functions such as zooming, panning, and searching for zipcodes. It also provides extensive capabilities to select orders, to extract subsets of orders, to merge sets of orders, to select existing flags of orders, and to subtract flags of orders. The Flag Formation processes are normally launched from this window.
All of the functionality of the Map window comes from the TransCAD Mapping macros and the TransCAD server. The TransCAD server must be running in the background for this window to work. Communication between TransCAD and the PTS User Interface is accomplished using DDE (Dynamic Data Exchange). This introduces some time delays and synchronization problems between the two programs that require some patience from the user. The faster the processor on the machine on which PTS is installed, the less of a problem this appears to be. We considered implementing the entire user interface within TransCAD but it was not adequate to the task.

Note that the orders selected in the Map window do not need to be the same as the orders selected in the Order Detail window. The Order Detail window has copy functions to copy to and from the Map and so bring them into alignment. This feature also enables the user to create interesting selections. One window (either the Map with its geographical selection tools or the Order Detail window with its query tools) can be used to select specific sets of orders and these sets can be copied and appended to the orders selected in the other window.

The power of the selection capabilities (and their extension through customized queries) in combination with the Flag Formation optimization processes should not be underestimated. It is one of the key strengths of PTS.

- **Summary Totals Report.** The next four reports appear on tabs within the Reports Window. The Summary Totals report takes the current selection of orders in the Order Detail window and performs a Transform query to display summary totals by schedule type. The PTS User can select different variables to summarize in this way and different groupings of subtotals. The results are
also displayed in a chart. This report can be connected live to Order Detail report so that the updates are performed automatically.

- **Line Detail Report.** This report displays the line detail for every order displayed in the Order Detail report. As with the previous window, this report can be connected live to the Order Detail report. Furthermore, as the user highlights a row in the Order Detail report, the first matching line entry for that order is automatically scrolled into view in the Line Detail report. This report is customizable in terms of the column headings, the order of the columns, and the column widths. The PTS User has drag and drop control over the latter two attributes.

- **Trial Stops Report.** Whenever a row is selected in the Unscheduled Flag list, the orders for that flag are displayed in the Order Detail window and in this report (if the live refresh option is selected). In this window, however, the orders are sorted by stop and the cumulative mileage and shipping units delivered are listed as well so the PTS user can examine the true feasibility of the load.

Note that the stop sequence and mileage data are the result of one of the Flag Formation optimization processes. That is, you cannot display stop sequences in this report until after you have formed a flag.

- **Custom Report.** This report displays the result of any query that the PTS Administrator adds to the Custom menu. A few sample queries are included in the base installation of PTS to illustrate the possibilities. This report is not connected live with any other window. The PTS User must select a menu item from the Custom menu to refresh this report.

- **Scheduled Flags.** The next two lists appear on the front tab of the Schedule window. The Scheduled Flags list is implemented as tree-structured outline
tool with flags belonging to Shift folders and Shifts belonging to Day folders. The sequence of flags within a Shift folders indicates its production sequence. The PTS User can rearrange a production schedule by simply dragging and dropping flags. The PTS User can open new folders to represent new production days or shifts. The PTS User can also drag a row from the Unscheduled Flag list and drop it onto the day and shift where it should be produced. After this action, a dialog box prompts the user to supply specific schedule information such as a flag name, a carrier name, and a pull date).

Note that this list is under the complete control of the PTS User. The Production Smoothing Optimization does not operate on this list. Instead, it updates unscheduled flags with recommended schedule dates in the Unscheduled Flags list. Only flags in the Scheduled Flags outline are exported back to the mainframe (A5 and A6).

- **Unscheduled Flags (Loads).** Initially, this list displays all unscheduled orders. Each order is treated as an unscheduled flag. Then, as the PTS User forms loads using the Flag Formation processes, these orders are combined and displayed as a single row in this list together with summary mileage and shipping units information as well as estimated cost (according to the Economic Parameters table). Each flag is costed as a contract carrier, as a fleet long truck, and as a fleet short truck. The lowest of the three costs is also displayed. This constitutes a PTS recommendation for carrier type. This is the cost that the Flag Formation optimization attempts to minimize.

The PTS User can drag a row to the Scheduled Flags outline view to schedule the flag or the user can leave it on the unscheduled list. Selecting any row or collection of rows in this list causes the matching orders to be displayed in the Order Detail window and the summary total information to be displayed in the
Summary Totals report. In this way, the PTS User can look at collections of flags and predict the impact of these flags on the production schedule by schedule type. The PTS User can sort this list in different ways by simply clicking on the column headings.

From this window, the PTS User can select a collection of flags and then launch the Production Smoothing Optimization to recommend to what production days to assign the different flags.

- **Working Totals of Unscheduled Flags.** This window is a dialog box that can be called up at any time to give the PTS User feedback on how the scheduling process is progressing. It displays statistics on the number of orders unscheduled and the transportation cost totals associated with the orders that have been formed into flags. It compares these statistics with the statistics from the last time the dialog box was displayed, thereby giving a progress report.

- **Production Smoothing Targets.** This window is a dialog box that permits the PTS User to modify the targets for the Production Smoothing Optimization and to launch the macro the runs this optimization. The user can also use this window to edit the defaults for these targets. This dialog is activated by clicking the Production Smoothing button on the Schedule window toolbar.

D. PTS Visual Basic Project Structure

The PTS User Interface program, PTS.exe, is a Visual Basic program compiled using Visual Basic 5.0. The source code for this program is found in the directory PTS\VBSource. The Visual Basic project file is “pts2.vbp.” The structure of the project together with a description of the purpose of each file is shown in Figure IV.5. The functions and procedures in each file are listed in the appendices.
Figure IV.5 PTS User Interface Visual Basic Project Structure

This completes our description of the Production Transportation Scheduling System from the perspective of a Developer. Additional details can be found in the Administrator’s Manual and the User’s Manual.
V. Appendix: Figures in Landscape View

Marketing Information System

Purchasing

Accounts Payable

Production Program Planning

Requirements Planning

Sales/Order Processing

Accounts Receivable

Material Flow Control

Sales/Dispatch

Personnel Information System

Capacity Management/Production Control

Payroll Accounting

Financial Accounting

Cost Accounting

Office Automation

CAD

Material Flow Control

A.W. Scheer, 1989

Figure 1.1 Enterprise Management View
Figure I.2 Enterprise Management Sales, Dispatch, Capacity Management and Production Control
Figure I.3 The Production Transportation Scheduling System Role in Enterprise Management
Figure II.1 The PTS Functional Process
Figure II.2 Build PTS Database Process (A2)
The Production Transportation Scheduling System

Figure II.3: Export Revised Order File (A5)

1. Copy Table Format to PTSLink
2. PTS Database
3. Order Header Table from PTSSData
4. Revised Order Data to PTSLink
5. Export Text File for Upload
6. Revised Order File

Table Format Stored in PTSSBuild

PTSLink Table Structure
Figure III.1 Flag Formation Process
Figure III.2 Form Single Flag Process (F1)
Figure III.3 Improve Set of Flags Process (F3)
Figure III.4 Arc Swapping (Stop Sequence Optimization)
Figure III.5 Sequence Flag (F12)
Figure III.6 Production Smoothing Optimization
Figure IV.1 PTS System Components
Figure IV.2 PTS Directory Structure
Figure IV.4 TransCAD Mapping Macros File Organization
Figure IV.4 PTS User Interface Views
Figure IV.5 PTS User Interface Visual Basic Project Structure