Decision Making in Manufacturing and Services
Vol. 2 • 2008 • No. 1–2 • pp. 63–78

Developing and Deploying Electronics Assembly Line Optimization Tools: A Motorola Case Study

Thomas M. Tirpak*

Abstract. The assignment of workloads to production equipment is one category of planning decision for an electronics assembly factory. In practice, line balancing requires not only selecting machines with sufficient placement accuracy and feeder capacity, but also addressing a host of other operational objectives and constraints. Motorola Labs led a multi-year effort to apply mathematical programming to balance a variety of production mix and volume scenarios. By representing the optimization problem as a specially structured, mixed linear-integer program, we were able to incorporate a high degree of reality in the model, simultaneously optimizing fixed setups, handling custom parts, maximizing machine uptime, and mitigating secondary bottlenecks. This paper presents the story of how we developed and deployed a software solution that significantly improved assembly cycle times, setup changeovers, and overall factory productivity, saving the company tens of millions of dollars.

Keywords: Electronics Assembly, Line Balancing, Linear-Integer Programming, Industrial Case Study

Mathematics Subject Classification: Primary: 90C90, Secondary: 90C11, 90C47

Journal of Economics Literature Classification: Primary: L63, Secondary: C61

Revised: 18 June 2008

1. INTRODUCTION

Line balancing for surface mount technology (SMT) lines involves distributing the assembly workload for a collection of board designs across a set of machines, so as to optimize the utilization of available production capacity. This task is one of several interrelated decisions for Design-to-Manufacturing (DTM) Information Management as defined in Tirpak (2000). After balancing the line, one can perform setup optimization, placement sequence optimization, and machine program generation. The decision hierarchy for SMT assembly processes is discussed in (McGinnis, Ammons, Carlyle, Cranmer, DePuy, Ellis, Tovey and Xu, 1992), (Rothhaupt, 1995) and (Van de Vall, 1998). SMT assembly optimization itself fits within the larger scope of planning and scheduling for electronics manufacturing, as addressed in (Sawik, Schaller, Tirpak, 2002) and (Kaczmarczyk, Sawik, Schaller, Tirpak, 2004).

* Motorola Home & Networks Mobility Business, 1301 E. Algonquin Rd., Room 4300A, Schaumburg IL 60196, U.S.A. E-mail: T.Tirpak@Motorola.com

Automated SMT assembly lines include process stations that perform the following steps. First, solder paste is applied to a printed wiring board, which is the substrate for the electronic circuit design being manufactured. Next, glue is dispensed at the locations on the board where heavy parts are to be placed. (This step is only necessary for double-sided boards.) Next, electronic components are placed at the desired locations on the board. Several configurations of robotic assembly equipment are available for placing small parts, e.g., resistors, capacitors, and inductors, large parts, e.g., integrated circuits, and odd-shaped parts, e.g., connectors, as described in Tirpak, et al. (2002). SMT lines are typically configured to make the part placement equipment the bottleneck, because of the cost of acquiring and operating this equipment. After all the SMT parts have been placed on the board, it travels through a conveyor oven in which the solder is reflowed, establishing the electrical and mechanical contact between the components and board. SMT lines may also include stations for manual assembly tasks and for inspecting boards.

SMT placement machines are capable of assembling a wide variety of board designs. Typically, a bar code on the board is read as it enters the placement machine, automatically selecting one of the available placement programs. A limiting factor to the flexibility of these placement machines is, however, their part feeder capacity. Depending on the type of equipment, the size of the parts, and the number of available feeder slots in the feeder carriages/banks, only a certain number of different parts can be loaded on a machine at any given time. Feeder setup changes may be made to swap-out "standard parts" and insert "custom parts" for a given board or group of boards, per the production schedule; however, such feeder changes may result in lost production capacity. For further details about the SMT assembly process, the reader is referred to Tirpak (2000).

Tools are available from SMT equipment vendors and third-party Computer Integrated Manufacturing (CIM) vendors to assign the part placement workload for a board design to the available equipment, and to subsequently generate machine program files in the necessary formats. However, at the start of this Motorola Labs project, it was widely known among factory engineers that better tools were needed. Cycle time studies had identified station-to-station imbalances of up to 60% for certain board designs in high-mix factories. Motorola's Community of Practice for electronics manufacturing identified equipment optimization as their first-priority tactic for minimizing cycle times and capital expenditures. SMT placement machines cost about $500,000, and the total equipment cost for one assembly line typically exceeded $5,000,000.

This paper tells the story of how a Motorola Labs team, working together with Manufacturing Operations and Computer Integrated Manufacturing (CIM) Teams around the company, developed a state-of-the art toolkit for balancing SMT lines. Sections 2.0 through 9.0 describe stages in the evolution of SMTBAL, from research software that rarely ran as expected, to being an integral part of Motorola's DTM system in multiple factories, with diverse equipment and production scenarios. Section 10.0 discusses some lessons learned and areas for further R&D work.

2. SCENARIO I: "HELP! WE'RE ALWAYS CHANGING SETUPS"

In 1988, the Corporate Manufacturing Research Center (now part of Motorola Labs) began its R&D program in the area of SMT manufacturing technology, which was still fairly new at the time. One of the first successes was the application of Traveling Salesperson Problem (TSP) solution algorithms to optimize placement sequences for revolving-turret type machines, known as "chip shooters". Rothaupt (1995), Sawik (2000), and Tirpak, Mohapatra, Nelson, and Rajbhandari (2002), describe in detail the available equipment types and optimization approaches.

We had recently completed placement sequence optimization for the Fuji CP-III machine in the Advanced Reflow Factory in Schaumburg, Illinois, when the manager noted that cycle times had improved, but the factory still had a large bottleneck, namely setup changeovers, which took 1–3 minutes per part. This factory prototyped and manufactured a large number of low-volume communications products, and had more than 350 board designs on its active list. The available part feeder capacity on the machines was less than 25% of what would be required for all active parts; therefore, a single fixed setup was clearly out of the question.

Our first approach was to develop a heuristic that incrementally constructed a standard setup starting with the most-used parts for the highest-volume board designs. A key decision variable was the number of feeder slots to reserve for custom setups, namely, those parts that did not fit within the standard (common) setup. Tirpak (1993) presents a simulation approach to evaluate scenarios, and select the one with the best overall production time (assembly plus setup changeover) for the anticipated board mix. Many candidate scenarios did not reserve enough feeder space for additional custom setup parts for low-volume board designs, and thus required breaking into the standard setup. Although our heuristic generated feasible solutions, i.e., with enough feeder capacity for every board design, we felt the incremental method of adding parts to the standard setup generated suboptimal solutions, i.e., with excess setup changeover time.

In January 1992, we discussed this problem with the Manufacturing Research Center at Georgia Institute of Technology. They proposed extending a mixed linear-integer programming model for balancing placements across multiple machines, to include custom parts that are added to the standard setup on a product-by-product basis, as discussed in (DePuy, 1995). The approach seemed promising for our factory scenario; however, the resulting mathematical program had more than ten times the number of constraints and variables than the line balancing models previously solved with the MINTO mixed-integer optimizer. Our initial job was to write a translator from the machine program file formats to the lists of part ID numbers used by the Georgia Tech software. Likewise, we needed to interpret the MINTO solution and create lists of actual parts in the optimized feeder setups, i.e., the Motorola part ID numbers of the parts to be loaded on each of the machines in the line. After several months of work with initial factory scenarios – including several cases where MINTO did not return a solution – we concluded that the method had merit; however, it would take a lot of work to incorporate other factory constraints, while maintaining a high level of confidence that we could generate a solution within a reasonable time.

At this point, Motorola Labs began work on SMTBAL Version 1.0, a test bed for large-scale, mixed linear-integer programming applications for SMT line balancing. The initial goals were to (1) include a high degree of realism in the optimization model, (2) tune the model to obtain good solutions in one day or less, (3) use the model in as many factories as possible, and (4) develop a tool that can be used by people who know little or nothing about mathematical programming. After benchmarking three commercially available Mixed Linear-Integer Program solvers, we selected the XA Callable C Libraries from Sunset Software Technologies. To ensure modularity and maintainability, should we wish to switch to a different solver in the future, we adopted the Mathematical Programming System (MPS) format for representing optimization problems.

Appendix A describes the basic problem structure. For each production lot, i.e., production order consisting of multiple boards with the same board design, the assembly cycle time is given by the cycle time on the bottleneck station, which includes the time for placement and load/unload operations. By moving parts to the feeder setup(s) on the alternate station(s), a portion of the placement workload can be off-loaded from the station with the longest cycle time, subject to various constraints.

3. SCENARIO II: "WE JUST GOT TWENTY MORE PRODUCTS FOR THIS LINE"

After hearing about results for the Advanced Reflow Factory, engineers in another factory in the same business unit requested help with line balancing. The Jedi Factory in Plantation, Florida, produced an assortment of about 30 handheld radios. Individual board designs had 200-400 parts. The line configuration included two high-speed chip shooters and one large-part placement machine for the first side assembly. Boards were flipped, and continued through the second side with two high-speed chip shooters and two large-part machines. The line had been in operation for several years, during which time the parts for many new board designs had been added to the feeder setups. Thus, production cycle times were not balanced for most boards. We loaded the placement programs for each board into SMTBAL, and generated an optimized setup. The simulation report indicated a significant improvement over current cycle times and a good workload balance, in which the predicted cycle time were within 10-20% of each other (across the Jedi140, 144, and 148 machines) for most products. However, factory engineers raised concerns about the accuracy of our placement time estimates, which were based on machine manufacturers' specifications. Figure 1 presents a sample report. To increase the level of confidence in the optimization results, we decided to load the current feeder setups into SMTBAL and generate simulation reports, i.e., without optimizing the setups. We then used a linear regression model to adapt per-part placement time estimates for six categories of parts, based on the types of parts and the measured cycle time for each product. We generated simulation reports and candidate setups, which passed the sanity test and subsequently received the go ahead to install them for the line.

```
No  Assembly/Lot Name    S LotSize Lots RunTime  SetTime  One Lot  Total
=== ==================== = ======= ==== ======== ======== ======== ========
1   NUE7240B.148         T 200     30   380.00   0.00     380.00   11400.00
2   NUF6394B.148         T 100     112  124.07   0.00     124.07   13895.47
3   NUF6395C.148         T 100     18   124.07   0.00     124.07    2233.20
RunTime: Placement, replenish, and load/unload time for one lot. (minutes)
SetTime: Flex/custom setups for one lot. (minutes)

Estimated total placement time (sec.) per board:
No  Assembly/Lot Name    S LotSize jedi140    jedi144    jedi148
=== ==================== = ======= ========== ========== ==========
1   NUE7240B.148         T 200     72.56      90.16      88.00
2   NUF6394B.148         T 100     39.44      56.88      48.00
3   NUF6395C.148         T 100     39.44      56.88      48.00
```

Fig. 1. Simulated assembly reports from SMTBAL provided vital information for tuning the optimization model parameters and justifying the effort to change to a new feeder setup.

A team of engineers and operators made the setup change during the third shift, so as to minimize the impact on production. I received a call at 10:00 a.m. that day. The tired voice said, "The line is up and running, and looks balanced for the product we're running now, but guess what... we just found out we need to build twenty new products on this line! We can't expect the team to go through this early morning exercise again. What can we do?" We started thinking about ways to incrementally adjust the existing setup, but knew that tweaking a setup for products one at a time could result in a significantly imbalanced line, as it had done in the past. What we needed was a way to rebalance the line for the new (expanded) set of products, while minimizing the effort to change from the existing setup. We accomplished this by including setup change time as a secondary factor in the objective function, as explained in Appendix B. The tweaked setup report indicated that the line could be rebalanced by moving about fifteen parts, and adding some unique parts for the twenty new products. Overall, we achieved a 34% improvement in production time for the Jedi Line across the new product mix.

4. SCENARIO III: "WE NEED TO BUILD TWO PRODUCT FAMILIES ON THE SAME LINE."

The next application was to balance the Jedi Line in Dublin, Ireland, which consisted of a pair of Sanyo TCM 1000 chip shooters in double-pass mode, where boards were cycled through the same machines to assemble the top and bottom sides of nineteen different kits. Each kit belonged to one of two product families, i.e., Jedi or Renaissance, which shared many common parts. The TCM 1000s had six banks of feeders, as shown in Figure 2. We decided to use the middle banks on both machines for high-usage parts common to both the Jedi and Renaissance families. The left and right banks would be used for additional parts required by individual product families.

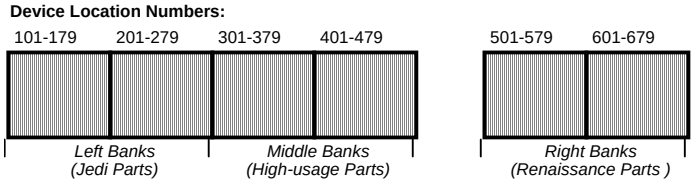

Fig. 2. The feeder banks of the Sanyo TCM 1000 can be used in split-mode, i.e., with the middle two banks and either the right or the left two.

We divided the line balancing problem into two phases. First, it was necessary to construct a list of the parts used most frequently by Jedi and Renaissance kits that can fit in the middle banks (320 feeder slots), while ensuring an even workload distribution across the two machines. The build plan for this phase included all top and bottom side kits, with a total of 363 unique parts. The feeder capacity in the SMTBAL software was set to 160 for each machine, but an unlimited number of custom/flex parts were allowed. The additional custom/flex parts, however, were subject to a setup time penalty.

The second phase considered the Jedi and Renaissance kits separately. The part-to-machine assignments for the high-use (middle bank) parts identified in the first phase were considered fixed. Namely, if a given part is assigned to the first machine's middle bank, it remains on the first machine. The SMTBAL software appended the remaining lower-usage parts required by the kit family, e.g., Jedi, to the feeder setups on each machine. In this case, the feeder capacity was set to 320, which is the number of slots in four banks, as shown in Figure 2. Feeder setups were constructed and implemented, resulting in a 17% net reduction in assembly cycle times across the Jedi and Renaissance product families.

5. SCENARIO IV: "WILL THIS MODEL EVER GIVE US AN ANSWER?"

By this time, news of SMTBAL had spread to other Motorola business units. Our next application was in Mansfield, Massachusetts, assembling ten families of networking products. The required number of feeders to hold all 292 parts exceeded the available feeder capacity by about 20%. Setup changeover time was a major concern, since factory schedules were generated for each two-day time horizon, and any mix of current products was possible. Thus, we decided to optimize setups with both standard and custom parts, using the model in Appendix A, with the addition of a set of variables to specify whether parts are included in custom setups for each product lot. Nevertheless, there were concerns about the computational complexity of this optimization problem. Enabling custom parts increased the number and size of the constraints and, more importantly, resulted in nearly 10 X the number of integer variables.

Although we had solved similar-sized optimization problems for other factory scenarios, this particular problem ran for several days without a solution. The enthusiasm

of the factory engineers quickly turned to pragmatism, asking "Will this model ever give us an answer? We're going to miss the window of opportunity to change the setup." We tested different branching strategies with the XA Solver, which improved the quality of the working solution but did not achieve convergence. After examining the problem formulation, we hypothesized that it was not the number of variables or constraints that was causing the problem, but rather the structure of the problem. The line consisted of two identical chip-shooters, followed by a large-part placement machine. Thus, the solver had no preference between the first two machines. Examining the branching tree of the integer programming algorithm indeed showed that some parts were assigned to the first machine and then to the second machine, and then changed to the first machine.

We decided to break this symmetry by providing an artificial preference for assigning certain parts to certain machines. Specifically, we adjusted the cycle time by a random small amount, e.g., 0.2 sec. became 0.0205 on machine A and 0.1998 on machine B. Overall, the calculated workload still reflected the true workload balance. We were happy to see the solver finally converge to a balanced setup, which was subsequently installed on the line. Several weeks of tracking via the Computer Integrated Manufacturing (CIM) System indicated that the actual throughput had increased by 8.5% including both production and setup time.

6. SCENARIO V: "THE REAL BOTTLENECK IS SOMEWHERE ELSE"

At this point, there was significant interest in SMTBAL, especially high-volume factories around the world. Work began on integrating the software into the standard program generation and download processes. One line balancing scenario for the factory in Libertyville, Illinois, resulted in a near-perfect match of the cycle times between two machines. The production engineer was proud of the result but worried that the line was still not balanced, because one machine required feeder replenishments four times more often than the other machine. We subsequently included an option in SMTBAL to provide part feeder capacity data, calculate replenishment frequencies, and include replenishment time in the overall production time minimization.

In a Swindon, U.K., factory, statistics for a "balanced" line indicated that operator assists (stopping the production flow) occurred almost fifteen times more frequently on one machine than another. The line was assembling boards for cellular network infrastructure products, some of which included large integrated circuits surrounded by small capacitors and resistors. For some designs, one part was used more than eighty times; therefore, the feeder reels for such parts would be depleted rapidly. We addressed this scenario by not only including reel replenishment time in the production time calculation, but also enabling parts to appear multiple times on the same machine and/or multiple machines.

A common problem was that, as additional products were added to a line's setup, feeder banks gradually ran out of space on the machine(s) with the popular parts for a given product family. Thus, it became more and more difficult to balance the line, and at some point the line became imbalanced. By reserving feeder capacity on all

machines, new parts required by new products or engineering changes notices (ECNs) can be accommodated more efficiently. We implemented this option in SMTBAL by enhancing the objective function to include a term reflecting the fraction of the available spaces in each feeder carriage that are filled.

One of the lessons of Factory Physics (Hopp and Spearman, 2000) is that it is easier to manage lines with distinct bottlenecks, as they provide a natural buffer for variability in the process flow. We frequently encountered this principle in practice in SMT assembly factories. Whereas the theoretical cycle time (including placement, setup, and replenishment times) for a given machine was lower than that of the bottleneck machine in the line, it limited the production rate of the entire line due to unplanned downtime, e.g., mis-picked parts, errors with the vision system, and jammed feeder reels. Furthermore, the work in queue between machines could cause blocking and/or starving in the line. In Figure 3, the maximum cycle time for a line with machines A-B-C is the same in Cases 1 and 2; however, the actual throughput is likely to be higher in Case 2. Appendix C explains how we adapted the optimization problem formulation to reduce the impact of secondary bottlenecks.

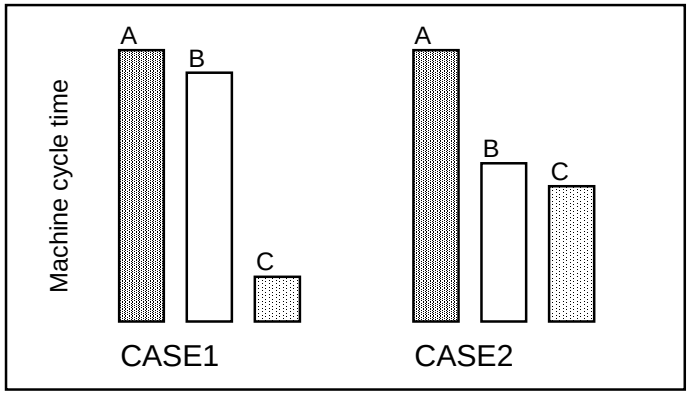

Fig. 3. Avoiding secondary bottlenecks and arranging workloads in decreasing order can mitigate the effects of unplanned downtime.

7. SCENARIO VI: "SOME PARTS REQUIRE SPECIAL CARE"

Line balancing applications in 1995–1997 involved a wide array of production scenarios, including machines and lines with more flexibility in the types of parts they could assemble. Although this flexibility created opportunities for better balancing, it also forced us to explicitly address assembly precedence constraints in our formulation of the line balancing problem. For some operations, a strict precedence must be kept, e.g., the placement of parts before the metal shield that is to cover those parts. As explained in Appendix D, precedence constraints for these operations may be expressed with respect to a particular station in the line or with respect to the assembly of a particular part.

When running SMTBAL for a high-volume line in Boynton Beach, Florida, we encountered preferences and constraints that reflected not only the feasibility and cycle time of placements, but also the ease of assembly and likelihood of a high-quality result. One example is that large parts (and heavy parts) should be placed as late as possible to minimize the opportunities for vibrating them off of their proper position on the printed wiring board during subsequent placement operations. Working with factory engineers, we developed a syntax for representing these constraints, e.g., "chip_12345, BEFORE, shield_1". Typically, fewer than five parts in the board design required special precedence constraints. However, it was extremely challenging to incorporate such constraints, as needed for individual parts, into the MPS formulation of the optimization problem. We decided to reorganize the software routines that generated the ROWS, COLUMNS, RHS (Right Hand Side), and BOUNDS sections of the MPS file, and structure them as independent code modules that could be toggled on or off, depending on the model options for the given scenario. Within each section of the MPS file, the data were generated, sorted accordingly, and finally written to an MPS file that was input to the XA Solver.

8. SCENARIO VII: "WHAT SETUP STRATEGY SHOULD WE USE?"

Within the first four years, we had used SMTBAL for more than a dozen factories, adding new functionality to handle a wide variety of production mix and volume scenarios, as well as equipment configurations. SMTBAL 2.0 was very flexible and well documented. However, the task of configuring it for a new production line typically required working with one of three experts in the company. Thus, we decided to develop an expert system to consolidate the knowledge gained through trial-and-error and factory applications to-date, and assist with the selection of a good setup strategy. As shown in the left column of Table 1, a key metric was the ratio of the number of feeder slots required to hold all the parts for all the board designs, divided by the number of available feeder slots.

For the Blue Line in Mansfield, MA, with fifty-four boards, 352 unique parts, and a total of 320 feeder slots available on two machines, the Feeder Ratio was approximately 1.7. Thus, we developed the following strategy. Board designs were assigned to one of four family groups. Each board design utilized a fixed setup (thirty-nine parts occupying fifty-three slots), plus one of four flex setups. For individual, low-runner board designs, which were not prioritized by the line balancing model, a couple parts in the flex setup were identified as candidates for tweaking for better balance. Benefits included: 15-20% faster cycle times, consolidation of thirty-eight individual setups into a single fixed setup and four flex setups, reduction in the number of feeders moved for setup changes, and reduction of blocking and starving of the machines due to parts replenishment.

Table 1. SMTBAL defaults to specific setup strategies, based on the expected utilization of feeder capacity.

Feeder Ratio	**Recommended Setup Strategy**
< 0.45	Construct a single fixed setup with all parts. Use only half of the available feeder banks. Reduce downtime by using the other half for off-line feeder replenishment.
< 0.95	Construct a single fixed setup with all parts. Use all of the available feeder banks.
< 1.25	Construct a fixed setup occupying part of the feeder banks, and append custom parts for individual products, as necessary.
< 2.00	Construct a fixed setup occupying a small portion of the feeder banks. Group the products into families, based on part similarity. Construct individual flex setups for each family.
≥ 2.00	Group the products into families, based on part similarity. Construct individual setups for each family.

9. SCENARIO VIII: "WE NEED TO MINIMIZE ENGINEERING TIME TOO"

The first applications of SMTBAL typically resulted in both celebration and trepidation. The former was the natural response to seeing a line running faster and smoother. The latter reflected the learning-curve to use SMTBAL for the first time and the desire to minimize the effort to maintain a balanced line. In many of the high-volume factories that successfully piloted SMTBAL, the factory's IT Group decided to integrate the software into their standard Design to Manufacturing (DTM) process flow, which began with Computer Aided Design (CAD) files and generated optimized machine setups and programs. For a given line and product family, consisting of a given set of board designs, many line balancing options, e.g., the feeder setup strategy, were hard coded, and most of the DTM process was automated. We also developed file import-export capabilities for common DTM tools, such as Cimbridge / Unicam, which maintained part geometry libraries and other process information.

A complementary approach, and one targeted at high-mix factories, was to develop an intuitive Graphical User Interface (GUI) and intelligent default settings for the full range of SMTBAL functionality. Although SMTBAL was a breakthrough for multicriteria optimization of high-mix setups, previously considered impossible or prohibitively expensive, i.e., 1–3 weeks of manual effort, we still needed to provide a tool that factory engineers could use at their desktop computers to generate such setups in less than a day. In 1997, Motorola Labs began work on Factory and Line Optimization (FLO) Tools, Windows based software that interfaced with the SMTBAL problem formulation and interpretation methods and the XA Solver via

Dynamically Linked Libraries (DLLs). The process was organized into a series of five screens: Setup, Feeder, Strategy, Results, and Machine. FLO Tools also included modules for: Board Grouping, Board-to-Line Assignment, and a Rule-Based Expert System for balancing lines with revolver-head placement machines (involving not only workload balancing but also optimization of nozzles for each revolver-head). Input data and incremental solutions, e.g., board-to-line assignments, were shared between the four modules in FLO Tools. A Web interface was also developed for FLO Tools. In 2002, the Motorola Software Group added Manufacturing Balance^{TM}to its suite of commercially available process planning and control tools.

Using the line balancing software and placement optimization tools integrated into its DTM process flow, the Wireless Data Group factory in Schaumburg, Illinois, eliminated the time to create custom setups for each new board design, achieved a 50% reduction in setup changeover time, and increased throughput by 13%. Because this factory prototyped new board designs, the improvement in both engineering time and setup time was particularly important. Management considered these to be critical factors in terms of time-to-market for new products. In addition to the benefits accrued from being first-to-market, the following impact was estimated over the typical product lifecycle:

$$(\mathrm{HS}_e + \mathrm{HS}_s)/\mathrm{WHD} * \mathrm{PCPP} = \text{Time-To-Market-Reduction}\,(\text{days})$$

$$\text{Time-To-Market-Reduction} * \mathrm{NP} * \mathrm{UNITS} * \mathrm{ASP} = \text{INCREASED REVENUE}$$

Where:

HS_e:	Average number of engineering hours saved per prototype build,
HS_s:	Average number of setup hours saved per prototype build,
WHD:	Work hours per day,
PCPP:	Average number of prototype cycles per board design,
NP:	Number of board designs supported,
UNITS:	Average number of units expected to be produced per day,
ASP:	Average selling price of the products.

10. CONCLUDING REMARKS

The line balancing project discussed in this paper successfully applied Mathematical Programming to improve throughput, quality, and costs, for more than twenty SMT assembly factories. The linear-integer programming formulation proved to be a sufficiently flexible and powerful platform for addressing a variety of goals and constraints, representing actual operating conditions. Compared with rule-based expert systems, as in (Csaszar, Nelson, Rajbhandari, Tirpak, 2000) our approach has the advantage of globally optimizing feeder setups with respect to multiple simultaneous objectives. From the beginning, we knew that we were pushing the computational limits of existing linear-integer program solvers. Nevertheless, with some trial-and-error, we succeeded in formulating and solving a mathematical program yielding a usable solution for a variety of high/low mix/volume production environments.

During the initial development and deployment of SMTBAL software, factory engineers participated in the design and testing of this toolset. We relied heavily on their inputs regarding factory operating constraints. These led to multiple enhancements, such as those discussed in Appendices A-D, and multiple incremental releases of the software. The simulation report in SMTBAL enabled us to perform "what if" studies for a variety of scenarios, and together with factory management select the best course of action. The software was supported initially by Motorola Labs, and then later by an internal technology transfer organization within Motorola Corporate.

We learned that good results with an initial pilot typically led to requests for testing the software in other factories, especially in Motorola's high-volume factories. SMTBAL was considered to be an expert tool; however, after the first major cycle time savings were measured, factories typically simplified things by automating SMTBAL in their standard DTM information management process. Though one of our goals was to hide the complexity of mathematical programming, we frequently received questions about how the balancing method works. Thus, we presented small-scale examples with the MS Excel Solver in a two-day SMT Manufacturing Optimization taught through Motorola University. This led to several other applications in factory layout optimization, production flow routing, supplier selection, and design optimization.

Across a wide range of factory scenarios, we achieved cycle time improvements of 8–34%, resulting in typical cost savings on the order of several hundred thousand dollars per line per year. Net savings to Motorola were in the tens of millions of dollars. Significant improvements in DTM cycle time (time-to-production-release) and engineering time were also observed. The feeder replenishment model and secondary bottleneck balancing in SMTBAL increased the actual production up time of the machines and thereby reduced variability in production lines. For example, by preparing backup reels for the ten most frequently used parts, engineers were able to increase the throughput of SuperCell Line 3 in Arlington Heights, Illinois, by approximately 30%.

The design and manufacture of electronic products involves many interrelated decisions during the life cycle of multiple product families. Speed and efficiency are crucial in both fast-paced consumer markets, where fashion trends quickly shape demand, and in established markets, where it is necessary to support a mix of legacy board designs. Decision-support systems, such as SMTBAL, play a crucial role in achieving the high levels of operational efficiency needed to compete within the manufacturing scope of a supply chain.

Acknowledgments

Thank you to Haomin Li, Weimin Xiao, and Mark Adams, for helping to develop SMTBAL/ FLO Tools. Thanks also to the factory engineers who tested and deployed the software, especially Ed Winter, Kevin Kent, Barry Groman, Ana Rodriguez, Brian Duffy, Joe Cawley, Steve Sabetta, Joe Belmonte, Leo Larivee, Dave Bettini, Andreas Brand, Andreas Schaller, Mike Rudnicki, Alex Lach, and Pete Hassler. Thanks to Georgia Tech, especially Gail DePuy, for the initial pilot. Thanks to Jim Byer (Sunset

Software Technology) for assistance with MPS and tuning solver parameters. Thanks also to Motorola Labs and factory managers, especially Tom Babin, Siegfried Pongratz, Roger Callanan, Faradeh Gozleveli, and Eric Gasmann. Thanks also to the DMMS reviewers and editorial board.

Appendixes

A. THE BASIC MODEL

The three main sets of entities in the optimization formulation are *Parts*, *Lots*, and *Stations*. For the scenarios described in this paper, the number of *Parts* was typically 200–300. For some scenarios it was more than 1,000. The number of *Lots* was 1–350, depending on whether the line was a dedicated production line, a low-mix line, or a high-mix line. The number of *Stations* was 1–5 for the actual lines; however, some experimental line configurations with 16–32 stations were also studied using this model.

There are several input parameters used to compute coefficients and bounds in the optimization model:

- $Panel\_Quant_j$=The quantity of panels of printed wiring boards assembled for lot j.
- $Feeder\_Slots_{ik}$= Number of feeder slots required for part i on station$k$.
- $Can\_Place_{ik} = 1$, if part i can be assembled by station k. 0, otherwise.
- $Part\_Quant_{ij}$ = Number of part i that are used on each panel of boards in lot j.
- $Part\_Time_{ik}$ = Time required for station k to place one part i.
- $Total\_Feeders_k$ = Total number of feeder slots available at station k.
- $Load\_Time_k$=Time required for load/unload operations for each panel at station k.

Among the following variables, the first two reflect actual decisions, i.e., the assignment of a part to the feeder setup on a particular placement station. The remaining three assist in formulating the objective function and constraints.

- $X_{ik} = 1$, if part i is located in the feeder setup at station k. 0, otherwise. (Binary variable)
- Y_{ijk} = Number of part i placed by station k for lot j.
- $Time_{jk}$ = Production time (part placement and board load/unload) for one panel in lot j on station k.
- $CTime_j$ = Total cycle time for lot j, i.e., the maximum production time, i.e., $Time_{jk}$, multiplied by the quantity of panels in lot j.
- $Feed_k$ = Number of feeder slot locations used at station k.

For the scenarios described in this paper, the total number of variables was typically 2,000-4,000. For some scenarios, it was nearly 16,000. The number of binary variables was typically 300-800. For some scenarios, it was 8,000. The number of constraints was typically 2,000-4,000. For some scenarios, it was nearly 18,000.

The objective function is to minimize the sum of the assembly times for each production lot:

$$\sum_{j \in Lots} CTime_j.$$

Let *Parts(k)* be the set of all parts that can be assembled by station k. Constraint (A.1) states that the total cycle time to complete each product lot is given by the cycle time of the longest station, multiplied by the quantity of panels in the lot. The assembly time for each panel consists of the time for board transport operations plus the sum of all part placement operations (A.2). The total number of feeder locations at a station is the sum of the feeder slots required for each part assigned to the station (A.3). The number of feeder slots utilized at a station may not exceed the total number available at that station (A.4). Each part on a panel must be assembled (A.5). A part cannot be assembled on a machine that does not contain that part in its feeder setup (A.6):

$$CTime_j \geq Panel\_Quant_j \, Time_{jk} \tag{A.1}$$

$$Time_{jk} = \sum_{i \in Parts(k)} Part\_Quant_{ij} \, Part\_Time_{ik} Y_{ijk} + Load\_Time_k \tag{A.2}$$

$$Feed_k = \sum_{i \in Parts(k)} Feeder\_Slots_{ik} X_{ik} \tag{A.3}$$

$$Feed_k \leq Total\_Feeders_k \tag{A.4}$$

$$\sum_{i \in Parts(k)} Y_{ijk} = Part\_Quant_{ij} \tag{A.5}$$

$$Y_{ijk} \leq Part\_Quant_{ij} X_{ik} \tag{A.6}$$

B. MINIMIZING CHANGES TO AN EXISTING SETUP

Let the variable $on\_station_{jk} \in \{0, 1\}$ be a flag indicating if part k is on station j in an existing setup. Let $delta\_plus_{jk} \in \{0, 1\}$ indicate if part k is in the existing feeder setup on station j but not in the new feeder setup, as given in X_{jk}. Let $delta\_minus_{jk} \in \{0, 1\}$ indicate if part k is not in the existing feeder setup on station j but in the new feeder setup, as given in X_{jk}. n is the number of allowed part changes from the existing feeder setup to the new feeder setup. Then, the additional constraints for limiting changes to the existing setup are:

$$delta\_plus_{jk} - delta\_minus_{jk} = on\_station_{jk} - X_{jk} \tag{B.1}$$

$$\sum_{\substack{j \in Lots \\ k \in Stations}} \left(delta\_plus_{jk} + delta\_minus_{jk} \right) \leq n \tag{B.2}$$

The objective function for the optimization model includes the summation, as shown on the left side of the (B.2) inequality, multiplied by a weighting factor characterizing the time to move a single feeder from one machine to another machine.

C. BALANCING SECONDARY BOTTLENECKS

Secondary bottlenecks can be balanced by adding a term to the objective function that reflects the assembly time differences (BT) between the stations in the line, where:

$$BT = BT\_weight \sum_{\substack{j \in Lots \\ r,s \in Stations}} BT\_diff_{jrs} \tag{C.1}$$

In which, BT_weight is the weight for the assembly time difference, as one of the multiple terms in the overall objective function. BT_diff_{jrs} is the assembly time difference between station r and station s for lot j, which is defined as:

$$BT\_diff_{jrs} \geq Time_{jr} - Time_{js} \tag{C.2}$$

$$BT\_diff_{jrs} \geq Time_{js} - Time_{jr} \tag{C.3}$$

D. ASSEMBLY PRECEDENCE CONSTRAINTS

The set of constraints enforcing part A to be placed at or before the station N can be expressed as:

$$X_{jA} = 0 \quad \text{for all} \quad j > S_N \tag{D.1}$$

Given that $MaxStation$ is the total number of stations in the line, the set of constraints enforcing part A to be placed at a station before the station where part B is placed can be expressed as:

$$MaxStation\, X_{jB} + \sum_{\substack{r \in Station \\ r \geq j}} X_{jA} \leq MaxStation \tag{D.2}$$

Let PT_weight be the time penalty for placing a part one station earlier than desired. Let *MaxStation* be the number of elements in set *Stations*. The effect of placing part A earlier than absolutely necessary can be characterized as:

$$PT_A = \sum_{j \in Lots} (MaxStation - j) X_{jA} \tag{D.3}$$

Thus, the total time penalty for all parts that are not placed as late as possible is:

$$PT = PT\_weight \sum_{i \in Parts(k)} PT_i \tag{D.4}$$

Equation (D.4) can be added to the overall objective function for the optimization problem, thereby characterizing the goal of placing certain parts as late as possible.

REFERENCES

Csaszar P.C., Nelson P.C., Rajbhandari R.R., Tirpak T.M., 2000: Optimization of Automated High Speed Modular Placement Machines Using Knowledge-Based Systems. *IEEE Trans. on Systems, Man, and Cybernetics Part C: Applications and Reviews*, Vol. 30, No. 4, pp. 408–417.

DePuy G.W., 1995: *Component Allocation to Balance Workload in Printed Circuit Card Assembly Systems.* Ph.D. Thesis. Georgia Institute of Technology, Atlanta, GA.

Hopp W.A., Spearman M.L., 2000: *Factory Physics Second Edition.* McGraw-Hill/Irwin, Boston MA.

Kaczmarczyk W., Sawik T., Schaller A., Tirpak T.M., 2004: Optimal Versus Heuristic Scheduling of Surface Mount Technology Lines. *International Journal of Production Research*, vol. 42 (10), pp. 2083–2110.

McGinnis L.F., Ammons J.C., Carlyle M., Cranmer L., DePuy G.W., Ellis K.P., Tovey A., Xu H., 1992: Automated Process Planning for printed Circuit Card Assembly. *IIE Transactions*, Vol. 24, No. 4, pp. 18–30.

Rothhaupt A., 1995: *Modulares Planungssystem zur Optimierung der Elektronikfertigung.* Carl Hanser Publishers, Munich.

Sawik T., 1999: *Production Planning and Scheduling in Flexible Assembly Systems.* Springer-Verlag, Berlin, Germany, pp. 113–116.

Sawik T., Schaller A., Tirpak T.M., 2002: Scheduling of Printed Wiring Board Assembly in Surface Mount Technology Lines. *Journal of Electronics Manufacturing, special issue on Production Planning and Scheduling in Electronics Manufacturing*, vol. 11 (1), pp. 1–17.

Tirpak T.M., 1993: *Simulation Software for Surface Mount Assembly.* Proc. of the 1993 Winter Simulation Conference. Los Angeles, CA, pp. 796–803.

Tirpak T.M., 2000: Design-to-Manufacturing Information Management for Electronics Assembly. *International Journal of Flexible Manufacturing Systems*, Vol. 12, Issue2/3, pp. 189–205.

Tirpak T.M., Mohapatra P.K., Nelson P.C., Rajbhandari R.R., 2002: A Generic Classification and Object-Oriented Simulation Toolkit for SMT Assembly Equipment. *IEEE Trans. on Systems, Man, and Cybernetics – Part A: Systems and Humans*, Vol. 32, No. 1, pp. 104–122.

Van de Vall L., 1998: Optimizing an SMT Line. *Surface Mount Technology Magazine*, pp. 48–52.