Fuzzy Family Setup Assignment and Machine Balancing

Fuzzy logic is applied to the world of printed circuit assembly manufacturing to aid in balancing machine loads to improve production rates.

by Jan Krucky

In disciplines such as engineering, chemistry, or physics, precise, logical mathematical models based on empirical data are used to make predictions about behavior. However, some aspects of the real world are too imprecise or "fuzzy" to lend themselves to modeling with exact mathematical models.

The tool we have for representing the inexact aspects of the real world is called fuzzy logic. With fuzzy logic, we can model the imprecise modes of reasoning that play a role in the human ability to make decisions when the environment is uncertain and imprecise. This ability depends on our aptitude at inferring an approximate answer to a question from a store of knowledge that is inexact, incomplete, and sometimes not completely reliable. For example, how do you know when you are "sufficiently close" to but not too far away from a curb when parallel parking a car?

In recent years fuzzy logic has been used in many applications ranging from simple household appliances to sophisticated applications such as subway systems. This article describes an experiment in which fuzzy logic concepts are applied in a printed circuit assembly manufacturing environment. Some background material on fuzzy logic is also provided to help understand the concepts applied here.

The Manufacturing Environment

In printed circuit assembly environments, manufacturers using surface mount technology are concerned with machine setup and placement times. In low-product-mix production environments manufacturers are primarily concerned with placement time and to a lesser degree setup time. In medium-to-high-product-mix production environments manufacturers are mainly concerned with setup time.

One solution to the setup problem is to arrange the printed circuit assemblies into groups or families so that the assembly machines can use the same setup for different products. In other words, reduce or eliminate setups between different assembly runs. The solution to minimizing placement time is to balance the component placement across the placement machines.

HP's Colorado Computer Manufacturing Operation (CCMO) is a medium-to-high-product-mix printed circuit assembly manufacturing entity. The heuristic, fuzzy-logic-based algorithms described in this paper help determine how to minimize setup time by clustering printed circuit assemblies into families of products that share the same setup and by balancing a product's placement time between multiple high-speed placement process steps.

The Placement Machines

The heart of our surface mount technology manufacturing lines in terms of automated placement consists of two Fuji CP-III high-speed pick-and-place machines arranged in series and one Fuji IP-II general-purpose pick-and-place machine.

A Fuji CP-III placement machine supports two feeder banks each having 70 slots available for component feeders to be mounted on (see Fig. 1). The components are picked from their feeders and placed on the printed circuit board, creating a printed circuit assembly. A component feeder might take one or two slots. The tape-and-reel type feeder, which is the one we use at CCMO, is characterized by its width for slot allocation purposes. The standard feeder tape widths are 8 mm, 12 mm, 16 mm, 24 mm, and 32 mm. The 8-mm feeder tapes consume one slot each while the 12-mm to 32-mm feeder tapes consume two slots. Additional feeder-to-feeder spacing constraints might increase the number of slots the feeders actually require. A component's presentation, package type, and style determine the tape-and-reel width and therefore the feeder size.

Split-Bank

A feature of the Fuji CP-III called *split-bank* addresses the problem of high setup costs by allowing one bank to be

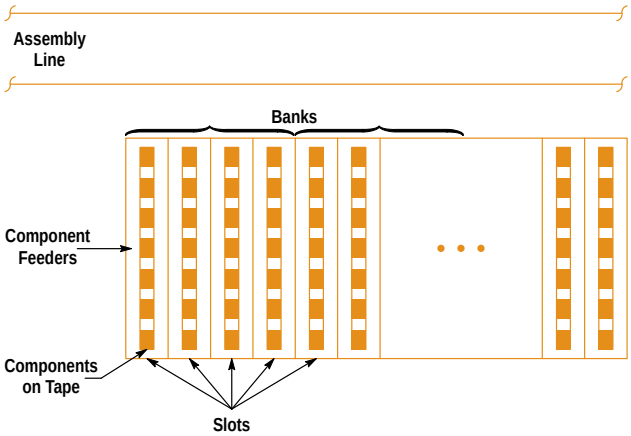


Fig. 1. Simplified representation of a tape-and-reel type placement machine.

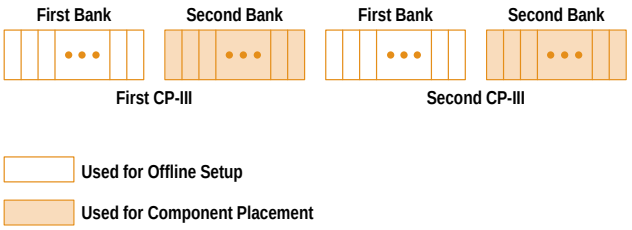


Fig. 2. The split bank feature of the Fuji CP-III assembly machine. The first feeder banks of each machine are used for offline setup.

used for component placement while the other bank is being set up offline. Fig. 2 illustrates this split-bank feature. In this configuration the first feeder bank on each machine is used to perform the offline setup, and the second bank is used for component placement.

Setup Time versus Placement Time

Our printed circuit assembly products vary quite a bit in their setup-slot requirements. They range from eight slots on the low end to 260 slots on the high end. For an average product requiring 45 slots it takes 45 online minutes to set up the feeders for placement. The average placement time is 2.5 minutes per board. In a low-to-medium-volume printed circuit assembly shop such as CCMO, the average lot size is 20 products. Therefore, for an average run of 20 products, 47% of the time is spent on setup (leaving the placement machine idle) and 53% of the time is spent placing the components. This constitutes an unacceptable machine use and hence a low output from the manufacturing shop. It's not just the online setup time, but also the frequency of having to do these setups that affects productivity and quality. A fixed setup for an entire run would seem to be a solution to this problem. However, in a medium-to-high-product-mix, low-volume environment such as ours, fixed setups cannot be used. HP CCMO currently manufactures 120 products with 1300 unique components placeable by the Fuji CP-III equipment.

Another quick solution to this online setup time problem would be to place a certain percentage of CP-III placeable components at a different process step. For example, we could use the Fuji IP-II for this purpose. This is not a feasible solution because the Fuji IP-II has a placement speed that is four times slower than the CP-III. We use the Fuji IP-II primarily for placing large components.

Alternate Setup Methodology

The setup time requirements mentioned above suggest that we needed an alternative setup methodology to minimize online setup time. The approaches we had available included expansion on the split-bank option described above, clustering the printed circuit assembly products into families with identical setup while still allowing the split-bank setup feature to be fully used, and balancing the two series CP-III placement loads as much as possible. Balancing implies that the components would be distributed between the two placement machines so that both machines are kept reasonably busy most of the time.

Since most of our printed circuit assembly products are double-sided, meaning that components are placed on the top and bottom sides of the printed circuit board, independent balancing for each side of the printed circuit assembly was considered. However, family clustering as viewed by the layout process dictated that a double-sided printed circuit assembly should be treated as a sum of the requirements for both sides of the board. Thus, no machine setup change would be required when switching from side A to side B of the same product.

Why Families

While clustering products into families is a viable and an attractive solution, other possible solutions such as partially fixed setups augmented by families or scheduling optimization to minimize the setup changes in the build sequence, are also worth consideration.

One can imagine that none of the solutions mentioned above will provide the optimal answer to every online setup-time issue, but their reasonable combination might. The following reasons guided us into choosing family clustering as an initial step towards minimizing online setup costs.

- Intuitive (as opposed to algorithmic) family clustering on a small scale has been in place at our manufacturing facility for some time.
- It appears that families give reasonable flexibility in terms of the build schedule affecting the entire downstream process.
- Families can take advantage of the CP-III's split-bank feature.
- By altering the time window of a particular family's assembly duration (i.e., by shift, day, week, month, and so on), one can directly control a family's performance and effectiveness.

We chose a heuristic approach to minimizing setup time because an exhaustive search is O(n!), where n is the number of products. Our facility currently manufactures 120 products and expects to add 40 new ones in the near future, which would make an exhaustive search unrealistic.

Primary Family

As explained above, we wanted to use the family clustering approach to take advantage of the CP-III split-bank setup feature. One can quickly suggest a toggle scenario in which each feeder bank would alternate between the states of being set up offline and being used for placement. However, it would be difficult to synchronize the labor-intensive activities of perpetual offline setups in a practical implementation. Also, this option would require a sufficient volume in the toggled families to allow the completion of offline setups at the idle placement banks. Given these reasons, a strict toggle approach would probably not have worked to improve the overall setup time in our environment. Instead of toggling, we selected an approach in which certain feeder banks are permanently dedicated to a family, and the remaining banks toggle between offline setup and placement. This approach led to the primary family concept.

A primary family is one that will not be toggled and is therefore always present on the machine. Since the primary family is permanently set up it logically follows that each nonprimary family includes primary family components. The more primary family slots used by a bank containing a nonprimary family the better. The summation of two series CP-III's banks provides four setup banks available on a line. We elected to dedicate the first bank of each CP-III to the primary family, leaving us with two banks for nonprimary families (see Fig. 3).

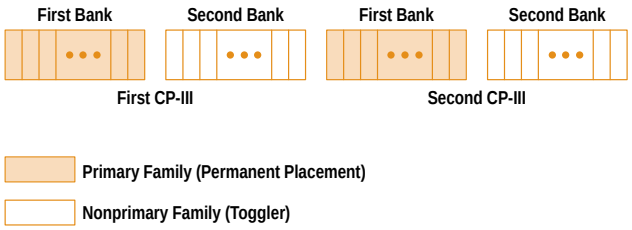


Fig. 3. The setup for primary and nonprimary families.

Fig. 4 shows how the primary family concept can be used to schedule a group of products to be built. Let's assume that we have primary family A and two nonprimary families AB and AC. (Products in families AB and AC contain components that are also part of products in family A.) First, any of family AB's products can be built. Although primary family A's products could be built using the same setup, it is highly undesirable since it would waste the presence of family AB's setup. So, after all the demand from family AB's products have been satisfied, we can switch to products in primary family A while we set up offline for family AC. On the practical side, it is unnecessary to set up the entire nonprimary family unless all the family's products are actually going to be built.

This example shows that the primary family concept is useful only if it is incorporated into the build schedule and the primary family must have sufficient product volume to allow the behavior depicted in Fig. 4.

Balancing

Family clustering results in a shared setup by a group of printed circuit assembly products which among other things might share the same components. This creates the problem of ensuring that the assembly of all products is adequately balanced on the two series CP-III placement machines. If the load is not balanced, an undesirable starvation in the process pipeline might occur. Balancing is accomplished by properly assigning the family's components between the two series CP-III machines. An intuitive guess suggests that the success of family clustering might make the balancing efforts proportionally harder. Although the online setup time reduction is the primary goal, it cannot justify a grossly imbalanced workload between two serial CP-IIIs. Therefore, machine balancing is the key that truly enables the family approach.

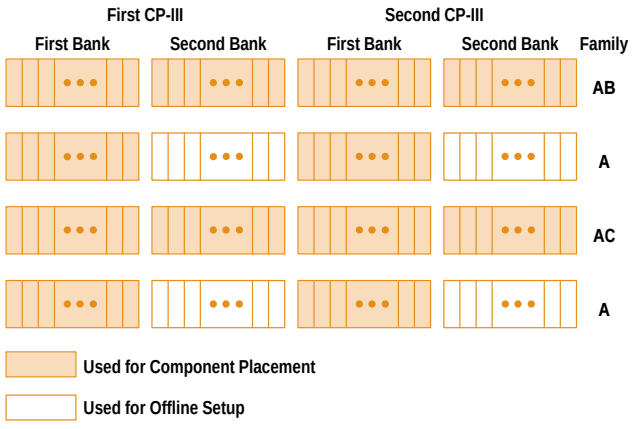


Fig. 4. An illustration of how the primary family concept can be used to schedule a group of products for assembly.

Other Methods

In addition to applying fuzzy set theory to solve our family assignment and balancing problem we also used two other approaches: the greedy board heuristic and an extension to the greedy board heuristic.

Greedy Board Approach. A research group at HP's strategic planning and modeling group in conjunction with Stanford University suggested the greedy board heuristic approach to minimize high setup cost for semiautomated manufacturing operations. HP's Networked Computer Manufacturing Operation (NCMO) implemented the greedy board approach at their site.[1]

In the greedy-board heuristic a family is defined by the repetitive addition of products, one at a time, until slot availability is exhausted. The selection criterion is a function of the product's expected volume and its additional slot requirements. The greedy ratio is:

$$G_i = s_i/v_i$$

where s_i is the number of additional slots a product p_i adds to the family, and v_i is the product's volume. Since the objective is to minimize the number of slots added while maximizing the family's volume, the product with the smallest greedy ratio wins and is added to the family. New slots are obtained and the selection process, via the greedy ratio, is repeated until either there are no more slots available or no more products are to be added. See the greedy board example on page 54.

The greedy board implementation at NCMO performs balancing by assigning components to the machines by a simple alternation until constraints are met. The components are initially sorted by their volume use. This approach balances the family overall, but it carries no guarantees for the products that draw from the family.

Extension to Greedy Board Heuristic. The greedy board heuristic tends to prefer smaller, high-volume printed circuit assemblies in its selection procedure. At CCMO we extended the original greedy ratio to:

$$G_i = \frac{s_i}{v_i \times c_i}$$

where c_i is an average number of slots product p_i shares with products not yet selected. The CCMO extension slightly curbs the volume greediness at the expense of including a simple measure of commonality. However, the results showed the CCMO extension to the greedy board heuristic performed slightly better than the original algorithm. The results from the two greedy-board approaches and the fuzzy approach are given later in this paper.

Despite the relatively good results achieved by our extension to the greedy board heuristic approach, we were still looking for an alternative approach. This led us to explore using fuzzy set theory to find a solution to our placement machine setup problem.

The Greedy Board Family Assignment Heuristic

As mentioned in the main article, a family in our manufacturing environment is a group of products (boards) that can be built with a single setup on the component placement machines. The greedy board heuristic is one way of assigning products to families for printed circuit assembly. The only data required for the greedy board algorithm is the list of components and the expected volume for each board. Each family is created by the repetitive addition of products, one at a time, until slot availability is exhausted.

In the following example assume there are eight component slots available per family and that the following boards must be assigned to families.

Board	Expected Volume (v_i)	Components
Alpha	1400	A, F, K, M
Tango	132	C, K
Delta	2668	H, D, R, F, K
Echo	1100	R, J, S, K
Beta	668	G, F, T, L
Lambda	1332	H, D, F, K
Gamma	900	A, J, E, K

The board with the lowest greedy ratio is the first one added to the current family being created.

Board	New Parts (s_i)	Expected Volume (v_i)	Greedy Ratio ($G_i = s_i/v_i$)
Alpha	4	1400	0.0029
Tango	2	132	0.0152
Delta*	5	2668	0.0019
Echo	4	1100	0.0036
Beta	4	668	0.0060
Lambda	4	1332	0.0030
Gamma	4	900	0.0040

Delta is the board with the lowest greedy ratio so it becomes the first member of the family. It has the highest product volume added per component slot used. Delta adds five components, leaving three slots to fill this family.

With the components H, D, R, F, and K already in the family, for the next board the ratios are computed as follows:

Board	New Parts (s_i)	Expected Volume (v_i)	Greedy Ratio ($G_i = s_i/v_i$)
Alpha	2	1400	0.0014
Tango	1	132	0.0076
Echo	2	1100	0.0018
Beta	3	668	0.0045
Lambda	0	1332	0.0000
Gamma	3	900	0.0033

The Lambda board is the one with the lowest greedy ratio because its components are a complete subset of the components already in the family. Since adding Lambda to the family does not require the addition of any components to the family, the greedy ratios given above still apply for the selection of the next board. The Alpha board has the next lowest ratio and it adds two new components (A and M) to the family. This brings the total number of components in the family to seven—one slot left.

After adding the Alpha board to the family, the new part-to-volume ratios for the remaining unassigned boards become:

Board	New Components (s_i)	Expected Volume (v_i)	Greedy Ratio ($G_i = s_i/v_i$)
Tango	1	132	0.0076
Echo	2	1100	0.0018
Beta	3	668	0.0045
Gamma	3	900	0.0033

Now Echo has the lowest ratio. However, the Echo board has two components, and since we already have seven components, adding the Echo components to the family would exceed our limit of eight components per family. Therefore, Tango is the only board that will fit even though it has the lowest theoretical contribution. Adding the Tango board fills up the family allotment. Finally, the components in the family include H, C, D, A, R, F, K, and M.

The next family is defined by following the above procedure for the remaining boards: Echo, Beta, and Gamma.

Fuzzy Set Theory

The following sections provide a brief overview of some of the basic concepts of fuzzy set theory applicable to the topics discussed in this paper. For more about fuzzy set theory see reference 2.

Fuzzy Sets

Unlike the classical yes and no, or *crisp* (nonfuzzy) sets, fuzzy sets allow more varying or partial degrees of membership for their individual elements (see Fig. 5). Conceptually only a few natural phenomena could be assigned a crisp membership value of either yes or no without any doubt. On the other hand, most of the real-world's objects, events, linguistic expressions, or any abstract qualities we experience in our everyday life tend to be more suited for a fuzzier set membership. Fuzzy sets allow their elements to belong to multiple sets regardless of the relationship among the sets.

In spite of their tendency to seem imprecise, fuzzy sets are unambiguously defined along with their associated operations and properties. The fuzzy sets used in the fuzzy family assignment and machine balancing heuristic exist in universes of discourse that are finite and countable.

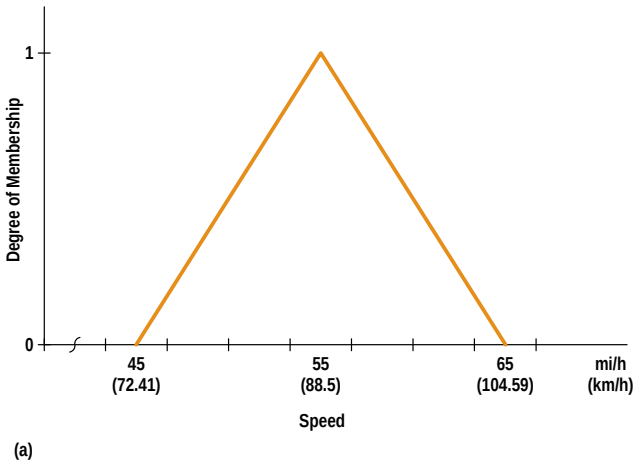
Definition. To begin our discussion of fuzzy sets we define the universe of discourse $X = \{x_1, x_2 \ldots x_i\}$ and let $\mu_A(x_i)$ denote the degree of membership for fuzzy set A on universe X for element x_i. The degree of membership function for fuzzy set A is $\mu_A(x) \in [0,1]$, where 0 represents the weakest membership in a set and 1 represents the strongest membership in a set.

$$\text{Fuzzy set A} = \frac{\mu_A(x_1)}{x_1} + \ldots + \frac{\mu_A(x_i)}{x_i}$$

where the horizontal bar is not a quotient but a delimiter.

Examples. The following examples show different types of fuzzy sets.

- Number as a fuzzy set:
  - Universe U = {0, 1, 2, 3, 4, 5}
  - Fuzzy set A = 0.2/0 + 0.7/1 + 0.8/2 + 0.2/3 + 0.1/4 + 0.0/5
  - Fuzzy set A might be described linguistically as "just about 2" because 0.8/2 has the highest degree of membership in fuzzy set A.
- Defining people in terms of their preference for certain alcoholic beverages:
  - Universe Y = {beer, wine, spirits} = $\{y_1, \ldots y_3\}$



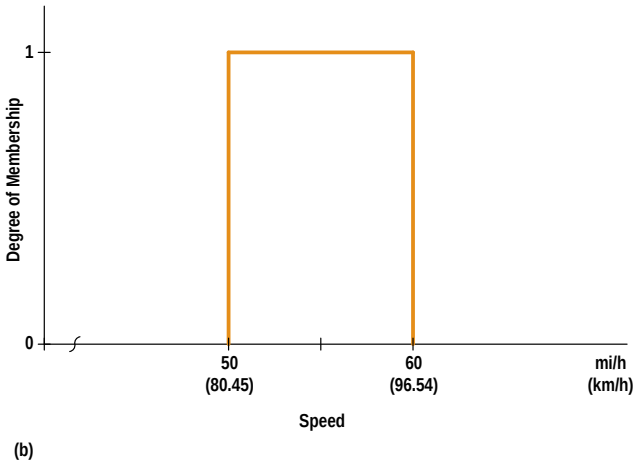


Fig. 5. Crispy and fuzzy representation of the notion of average driving speed. (a) In the fuzzy representation the membership class average driving speed varies from zero for <45 mi/h or >65 mi/h to 100% at 55 mi/h. (b) In a crisp representation membership in the average driving speed set is 100% in the range from 50 to 60 mi/h only.

- Fuzzy set B = $\frac{\mu_B(y_i)}{y_i}$ = 0.1/beer + 0.3/wine + 0.0/spirits might describe somebody who doesn't drink much but prefers wine and dislikes spirits
- Fuzzy set C = 0.8/beer + 0.2/wine + 0.1/spirits might describe a beer lover
- Fuzzy set D = 0.0/beer + 0.0/wine + 0.0/spirits might describe a person who doesn't indulge in alcoholic beverages
- Fuzzy set E = 0.8/beer + 0.7/wine + 0.8/spirits might describe a heavy drinker.

- Defining a person in terms of their cultural heritage:
  - Universe Z = {Zirconia, Opalinia, Topazia} and $\mu_F(z_i)$ represents a degree of cultural heritage from the three provinces in some imaginary gem-producing country.
  - Fuzzy set F = 0.3/Zirconian + 0.5/Opalinian + 0.1/Topazian might describe someone who was born in Western Opalina, attended a university in Zirconia, and married a Topazian living in Diamond City, Zirconia.
- Lunch hour:
  - Universe W = Day (continuous time of 24 hours)
  - The fuzzy set L (1100 to 1300) might represent the term "lunch hour" as shown in Fig. 6.

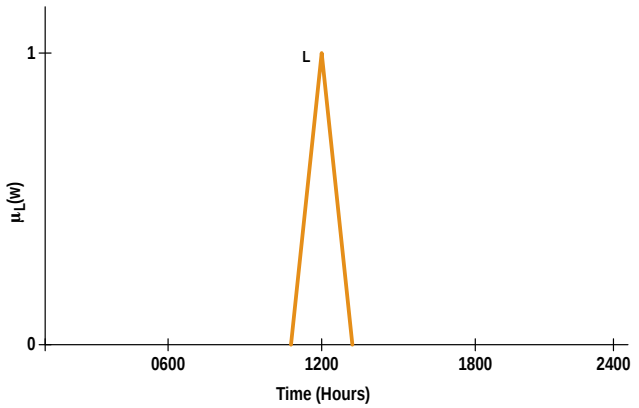


Fig. 6. A fuzzy set representation of the term lunch hour.

Operations. Operations such as union, intersection, and complement are defined in terms of their membership functions. For fuzzy sets A and B on Universe X we have the following calculations:

- Union: $\mu_{A\cup B}(x) = \mu_A(x) \vee \mu_B(x)$
  or $\forall x_i : \mu_{A\cup B}(x_i) = \mathrm{Max}(\mu_A(x_i), \mu_B(x_i))$
  (see Fig. 7a).

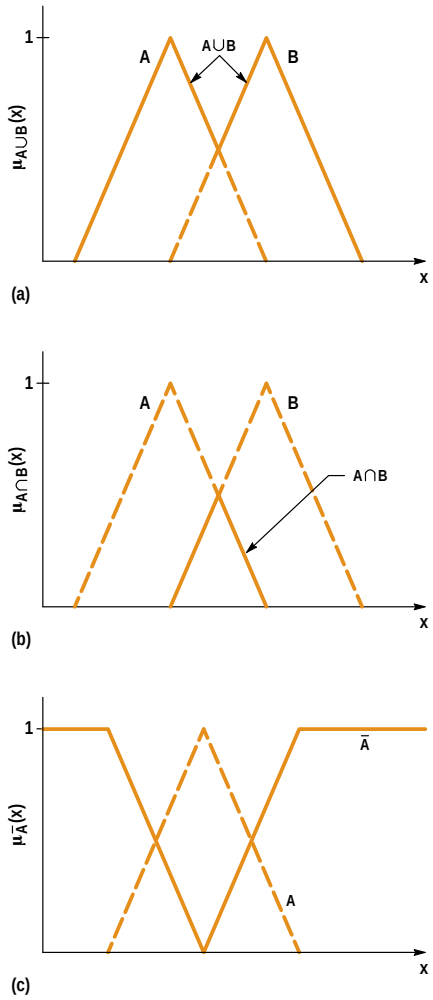


Fig. 7. Fuzzy set operations. (a) Union. (b) Intersection. (c) Complement.

- Intersection: $\mu_{A\cap B}(x) = \mu_A(x) \wedge \mu_B(x)$
  or $\forall x_i : \mu_{A\cap B}(x_i) = \text{Min}(\mu_A(x_i), \mu_B(x_i))$
  (see Fig. 7b).
- Complement: $\mu_{\bar{A}}(x) = 1 - \mu_A(x)$
  or $\forall x_i : \mu_{\bar{A}}(x_i) = 1 - \mu_A(x_i)$
  (see Fig. 7c).

All of the operations defined above hold for fuzzy or classical set theory. However, the two formulas known as excluded middle laws do not hold for fuzzy sets, that is:

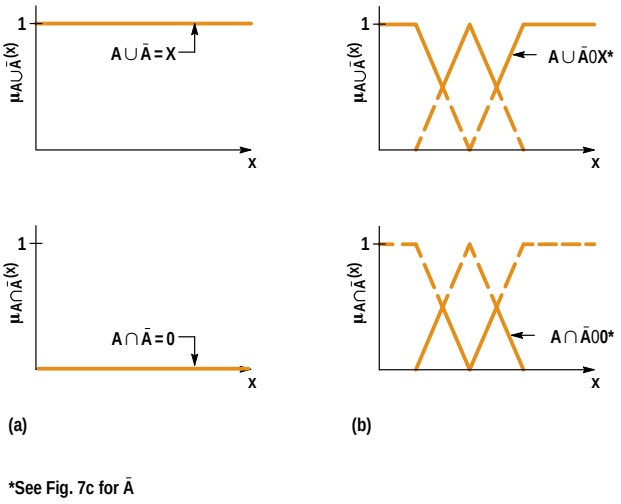
$$A \cup \bar{A} = X$$
$$A \cap \bar{A} = 0$$

for classical set theory, but

$$A \cup \bar{A} \neq X$$
$$A \cap \bar{A} \neq 0$$

for fuzzy set theory.

These laws, which take advantage of the either-or only membership for a classical set's elements, cannot hold for fuzzy sets because of their varying degree of set membership. Fig. 8 provides a graphical comparison between these two formulas for classical and fuzzy set operations.



*See Fig. 7c for Ā

Fig. 8. A comparison of the excluded middle laws for (a) classical sets and (b) fuzzy sets.

Fuzzification and Defuzzification

Fuzzification and defuzzification are operations that translate back and forth between fuzzy and crisp representations of information, measures, or events. Since most of our environment is more naturally represented in a fuzzy form rather than a crisp form, the need for a fuzzification step could be perceived as being a rare event. On the other hand, a defuzzification procedure is needed more often, as in the case in which a fuzzy set has to be expressed as a single crisp number. There are several defuzzification methods. One of the most commonly used and computationally trivial is the Max method. The Max method simply chooses an element with the largest membership value to be the single crisp representation of the fuzzy set. For example, for the fuzzy set C given above the Max defuzzification method would yield 0.8/beer (i.e., fuzzy set C describes a beer lover).

Fuzzy Relations

The concept of relations between fuzzy sets is fairly analogous to the idea of mapping in classical set theory in which the elements or subsets of one universe of discourse are mapped to elements or sets in another universe of discourse. For example, if A is a fuzzy set on universe X and B is a fuzzy set on universe Y then the fuzzy relation $R = A \otimes B$ maps universe X to universe Y (i.e., R is a relation on universe X **x** Y). The symbol $\otimes$ denotes a composition operation which computes the strength of the relation between the two sets. Please note that in general $A \otimes B \neq B \otimes A$ and furthermore $A \neq R \otimes B$.

Special Properties. The following are some of the special properties of fuzzy relations.

- A fuzzy set is also a fuzzy relation. For example, if A is a fuzzy set on universe X and there exists I = 1/y as an identity fuzzy set on Universe Y = {y}, then fuzzy relation $R = A \otimes I = A$.
- The same operations and properties valid for fuzzy sets also hold for fuzzy relations.

- Fuzzy logic implication of the form $P \rightarrow Q$ can be also represented by a fuzzy relation since $T(P \rightarrow Q) = T(\bar{P} \vee Q)$ where T is the truth evaluation function. For example, if A is a fuzzy set on universe X and B is a fuzzy set on universe Y then a proposition $P \rightarrow Q$ describing IF A THEN B, is equivalent to the fuzzy relation $R = (A \otimes B) \cup (\bar{A} \otimes Y)$. Fig. 9 shows a graphical representation of this relationship.

Fuzzy Composition

Fuzzy composition operations compute the strength of the relation between two fuzzy relations. To show the most popular composition operators, consider that we have fuzzy sets X, Y, and Z and that R is a fuzzy relation on universe X **x** Y and that S is a fuzzy relation on universe Y **x** Z. To find the fuzzy relation $T = R \otimes S$ on universe X **x** Z we use one of the following composition operations:

- Max-Min: $\mu_T(t_{i,j}) =$

$$\text{Max}\left[\text{Min}\left(\mu_R(r_{k,i}), \mu_S(s_{j,k})\right)\forall k : \leq \beta\right]\forall i, j : i \leq \alpha, j \leq \gamma \qquad (1)$$

- Max-Product: $\mu_T(t_{i,j}) =$

$$\text{Max}\left[\text{Prod}\left(\mu_R(r_{k,i}), \mu_S(s_{j,k})\right)\forall k : \leq \beta\right]\forall i, j : i \leq \alpha, j \leq \gamma \qquad (2)$$

- Sum-Product: $\mu_T(t_{i,j}) =$

$$\sum\left[\text{Prod}\left(\mu_R(r_{k,i}), \mu_S(s_{j,k})\right)\forall k : \leq \beta\right]\forall i, j : i \leq \alpha, j \leq \gamma \qquad (3)$$

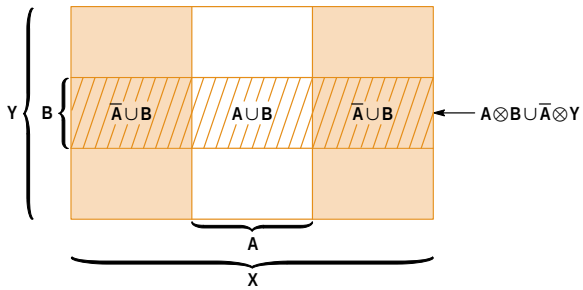


Fig. 9. A graphical depiction of the fuzzy logic implication $R = (A \otimes B) \cup (\bar{A} \otimes Y)$.

where:

α, β, γ	are the number of elements (cardinality) in the fuzzy sets X, Y, and Z
i, j, k	are subscripts for matrix representations for the fuzzy relations.

The Max-Min composition operator selects the maximum membership value from all the minimal values of every corresponding membership pair. For example, the Max-Min value from the membership pairs [(0.4,0.6), (0.2,0.5)] is 0.4. The Max-Prod composition operator replaces the Min function in the Max-Min operator with the Prod function, which performs algebraic multiplication for every membership pair. The Max-Prod value for our example above is 0.24. Finally, the Sum-Prod operator is derived from the Max-Prod operator by replacing the Max function with the Sum function, which adds together the results of the Prod operations on each membership pair. Applying the Sum-Prod operator to the example above gives the value 0.34. The are many other composition operators available, some of which are designed for specific applications.

Deriving Fuzzy Relations. The most difficult part about developing an application using fuzzy relations is obtaining the relations themselves. Some of the methods used to derive fuzzy relations include:

- Intuitive knowledge, human experience, and opinions of experts
- Calculation methods for membership functions
- Fuzzy compositions
- Converted frequencies and probabilities.

Example. The following example illustrates how to derive a fuzzy relationship. Consider the fuzzy sets and universes described earlier:

Universe Y = {beer, wine, spirits}

Universe Z = {Zirconian, Opalinian, Topazian}

We will assume for this example that the relation R on universe Y **x** Z is based on the opinions of experts who know a lot about the drinking habits of the inhabitants of the provinces contained in universe Z.

$$R = \overbrace{\begin{bmatrix} 0.6 & 0.3 & 0.1 \\ 0.4 & 0.8 & 0.3 \\ 0.2 & 0.1 & 0.7 \end{bmatrix}}^{Z} \Big\} Y$$

Although the relation R is derived from intuitive knowledge and experience, we could have used one of the other methods to derive it based on some partial information. Remember that a fuzzy relation captures the pairwise strength of the relation between elements of both universes, which in this case consists of beer, wine, and spirits in rows and Zirconian, Opalinian, and Topazian in columns. For example, there is a strong (0.8) possibility of an Opalinian being a wine lover according to relation R.

Now let's take a beer lover described by the fuzzy set

C = 0.8/beer + 0.2/wine + 0.1/spirits

and perform Max-Min composition on the relation

$H = C \otimes R$

Applying Max-Min yields:

μ_H(Zirconian) =
Max[(Min(0.8,0.6),Min(0.2,0.4),Min(0.1,0.2)] = 0.6

μ_H(Opalinian) =
Max[(Min(0.8,0.3),Min(0.2,0.8),Min(0.1,0.1)] = 0.3

μ_H(Topazian) =
Max[(Min(0.8,0.1),Min(0.2,0.3),Min(0.1,0.7)] = 0.2

Therefore, the resulting fuzzy set H = 0.6/Zirconian + 0.3/Opalinian + 0.2/Topazian might suggest that a beer lover is of predominantly Zirconian heritage with slight linkages to Opalinian influences and very slight Topazian influences based on the experts' opinion represented in relation R.

Fuzzy Family Assignment Heuristic

The goal of our fuzzy family assignment heuristic is to find products with similar components and group them into families. In our family assignment heuristic there are two nested iteration loops: an outer loop for each family being created and an inner loop for selecting the "best-suited" product to assign to the family. The inner loop is terminated when there are no more products to be considered. The outer loop is terminated either when there are no more families or when no more products are being assigned to a particular family. The following is a pseudo-code representation of our algorithm.

```
1.  Family = Primary /* Initialization family variable */
2.  REPEAT           /* Start outer loop                */
3.    Qualify = PCA /* Products to be assigned to a family.*/
4.    WHILE Qualify <> Empty /* Start inner loop. Loop  */
                             /* until there are no more products  */
5.      Find Product from Qualify with the highest selectivity
        measure (si)
6.      IF (a qualified Product is selected AND the slots
          required by the selected Product ≤ slot availability
          of Family
7.      THEN
8.        Assign Product to Family and update slot
          availability of Family
9.        Remove Product from PCA
10.     END IF
11.     Remove Product from Qualify
12.   END WHILE
13.   Family = get_a_new_family(Family)
14. UNTIL (PCA does not change OR no more Families)
```

Product	product being considered for inclusion in a family
get_a_new_family	returns next available family's name and slot availability
slot availability	counter for the number of placement machine slots available to a family (This number is decreased by the number of slots required by each product assigned to the family.)
PCA	list of products to be considered for family assignment (This list is updated each time a product is assigned to a family.)
Qualify	same as PCA except that this variable is

used to determine when to terminate the inner loop and is updated at each iteration.

At the end of this algorithm there still might be products that cannot be assigned to any family because of slot availability, or there might be families with no products assigned.

We used the concepts of fuzzy sets, fuzzy relations, and fuzzy composition to determine which products to select and assign to each family. The variables used in our algorithm include a fuzzy set p_i which represents the printed circuit assembly products, a selectivity measure s_i, which is a value that indicates how each product p_i might fit into a particular family, and finally, the fuzzy relation r, which is used to capture the relation between selectivity s_i and product p_i.

Since the volume, commonality (common parts), and additional slots are the three independent qualifiers that describe a product, they were used to define the product universe P = {commonality, slots, volume}. Thus, a product

$$p_i = m1_i/\text{commonality} + m2_i/\text{volume} + m3_i/\text{slots} \qquad (4)$$

is a fuzzy set on universe P where $m1_i$, $m2_i$, and $m3_i$ are the membership values on the interval <0,1> for product p_i.

We implemented the following computational methods to obtain the membership values for universe P.

- General commonality. General commonality between product p_1 and p_2 is defined as:

$$\text{comm}(p_1,p_2) = \text{ncs}(p_1,p_2)/\text{tns}(p_1) \qquad (5)$$

where ncs is the number of slots common to p_1 and p_2, and tns is the total number of slots required by p_1. It could be deduced that in general:

$$\text{comm}(p_1,p_2) \neq \text{comm}(p_2,p_1) \text{ unless } \text{tns}(p_1) = \text{tns}(p_2).$$

- Commonality during primary family selection:

$$m1_i = \frac{\sum_{j=1}^{j=N \wedge j \neq i} \text{comm}(p_j, p_i)}{N - 1} \quad \text{for product } p_i. \qquad (6)$$

N is the number of products not yet assigned to a family.

- Commonality during nonprimary family selection:

$$m1_i = \frac{\sum_{j=1}^{j=N \wedge j \neq i} \text{comm}(p_i, p_j)}{N - 1} \quad \text{for product } p_i. \qquad (7)$$

N is the same as above.

- Volume:

$$m2_i = \frac{\text{demand}(p_i)}{\underset{j=1}{\overset{j=N}{\text{Max}}}(\text{demand}(p_j))} \quad \text{for product } p_i \qquad (8)$$

where demand(p_i) is the expected volume demand for product p_i and N is the same as above.

- Slots:

$$m3_i = 1 - \frac{\text{slots}(p_i)}{\text{slots\_available}_t} \quad \text{for product } p_i \qquad (9)$$

where slots(p_i) is the number of additional slots required for product p_i if it is selected, and slots_available_t is the number of slots available for a particular family during iteration t of the assignment algorithm.

Selectivity and Fuzzy Relation

Since the selectivity measure s is defined on the universe S = {selectivity}, the selectivity for product p_i is defined as a fuzzy set on universe S:

$$s_i = m_i/\text{selectivity}. \qquad (10)$$

Fuzzy relation r on universe R = P **x** S is used to capture the relation between product selectivity and the product itself. When we translate the general notion of a fuzzy relation into the reality of our problem, we end up with a 3×1 matrix representation of the relation. The column symbolizes the cardinality of universe S and the three rows relate to the product universe P (i.e., commonality, volume, and slots). Since different selection criteria might be desired at different stages of the selection process, we found a need for at least two distinct relations. Thus, based on our experience we selected the following two categories that might require separate fuzzy relations r.

- First product assigned to a primary or nonprimary family
- Nonfirst product assigned to a primary or nonprimary family.

The hardest part about using fuzzy relations is obtaining their membership values. We wanted the membership values derived for the relation r to express the importance assigned to each of the three elements in universe P (i.e., commonality, volume, and slots) during the process of selecting products to add to a particular family. For example the relation:

$$r_i = \begin{bmatrix} 0.7 \\ 0.4 \\ 0.2 \end{bmatrix} \begin{matrix} \text{(commonality)} \\ \text{(volume)} \\ \text{(slots)} \end{matrix}$$

says that for product p_i during an iteration of the assignment algorithm, commonality is to be given greater emphasis in family assignment than volume or slots membership values.

One can use one of many fuzzy composition operators to construct the relation, or one can intuitively guess the fuzzy relation r based on some empirical experience or expertise. In our prototypical implementation, we selected the second approach.

Initially, we experimented with the empirically derived graph shown in Fig. 10 to come up with the membership values for the two categories of fuzzy relations mentioned above. Note in Fig. 10 that the fuzzy relationship values for commonality, volume, and slots are dependent on the slots membership value. For example, a slots membership value of 0.5 would provide the relation matrix:

$$r_i = \begin{bmatrix} 0.8 \\ 0.2 \\ 0.0 \end{bmatrix}.$$

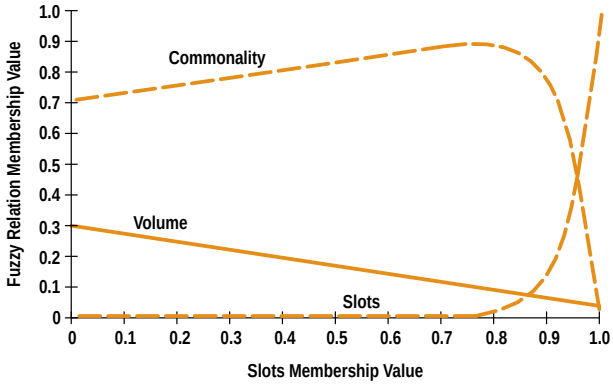


Fig. 10. Fuzzy relation membership values for our experimental functional approach to family assignment.

This approach turned out to be too complex and cumbersome because of our lack of experience with creating membership relations. Consequently, at the end we settled for a constant determination of a relation's individual values to get us started. This approach resulted in the following two fuzzy relations in our prototypical implementation:

- First product assigned to a primary or nonprimary family

$$r_i = \begin{bmatrix} 0.550 \\ 0.150 \\ 0.300 \end{bmatrix} \text{ for product } p_i \ (1 < i < N)$$

- Nonfirst product assigned to a primary or nonprimary family

$$r_i = \begin{bmatrix} 0.550 \\ 0.200 \\ 0.250 \end{bmatrix} \text{ for product } p_i \ (1 < i < N)$$

where N is the number of products not yet assigned.

It is important to notice that in general no restrictions are imposed on fuzzy membership values, but since we used the Sum-Product fuzzy composition operator then the summation of the elements in each relationship matrix must be ≤ 1.

Example

The fuzzy composition for our family assignment problem is $s_i = r_i \otimes p_i$. Although we investigated a number of fuzzy composition operators, we had the most success with the Sum-Product composition operator.

The following example illustrates the actions performed by the assignment algorithm to select a product to assign to a particular family.

- Assume there are three products p_1, p_2, and p_3 and that we are selecting the first product to be assigned to a primary or nonprimary family.
- ncs (number of common slots) for pairs of p_i, p_j for $i \leq 3$, $j \leq 3$ is:

$$\begin{bmatrix} 20 & 10 & 12 \\ 10 & 30 & 21 \\ 12 & 21 & 40 \end{bmatrix}$$

- slots_available = 45
- demand (p_1) = 85, additional slots(p_1) = 20

From equation 4:

$$p_1 = m1_1/\text{commonality} + m2_1/\text{volume} + m3_1/\text{slot}$$

where:

by equation 5: $\text{comm}(p_1, p_2) = 10/20 = 0.5$ and
$\text{comm}(p_1, p_3) = 12/20 = 0.6$.

From equations 7, 8, and 9:

$$m1_1 = \frac{0.6 + 0.5}{2} = 1.1/2 = 0.55$$

$$m2_1 = 85/\text{Max}(85, 100, 60) = 0.85$$

$$m3_1 = 1 - 20/45 = 0.56.$$

Finally,

$$p_1 = 0.55/\text{commonality} + 0.85/\text{volume} + 0.56/\text{slot}$$

- demand(p_2) = 100, additional slots(p_2) = 30, and

$$p_2 = 0.51/\text{commonality} + 1.00/\text{volume} + 0.33/\text{slot}$$

- demand(p_3) = 60, additional slots(p_3) = 40, and

$$p_3 = 0.41/\text{commonality} + 0.60/\text{volume} + 0.11/\text{slot}.$$

- Since $r_{1,2,3} = \begin{bmatrix} 0.550 \\ 0.150 \\ 0.300 \end{bmatrix}$ using equation 3, the Sum-Product operator, the fuzzy composition $s_i = r_i \otimes p_i$ for this example is:

$$s_1 = r_1 \otimes p_1 = 0.55 \times 0.55 + 0.150 \times 0.85 + 0.300 \times 0.56 = 0.598$$

$$s_2 = r_2 \otimes p_2 = 0.529$$

$$s_3 = r_3 \otimes p_3 = 0.348.$$

- Finally, since $\text{Max}(s_1, s_2, s_3) = s_1$, product p_1 has the highest selectivity measure and is therefore assigned to the family being formed during this iteration.

Fuzzy Machine Balancing

After the fuzzy family assignment algorithm assigns the products to their corresponding families, the fuzzy machine balancer tries to assign each family's components to the placement machines. The primary objective is to have each side of the assembled printed circuit assembly use the two series CP-IIIs as equally as possible.

Constraints

Aside from the inherent constraints introduced by families, manufacturing reality brings a few special cases of already predetermined machine assignments and constraints.

Physical Process Constraints. Since the objective is to have as much setup slot room as possible, certain physical process related limitations arise. For example, constraints on the very last slot available on the bank do not allow a two-slot-wide feeder to be mounted on the last slot. If we have one slot still available on each machine and we have to place a two-slot-wide feeder, we need to move a one-slot-wide feeder from one machine to make room for the two-slot-wide feeder. Finally, a component whose package is higher than 3.5 mm must be placed by the second machine since the component height might interfere with the placing nozzle on a densely populated printed circuit assembly.

Primary Family Products. If the printed circuit assembly members in the primary family could be balanced without consideration for the remaining printed circuit assemblies that use a portion of the primary family, balancing could probably be achieved at the expense of the remaining products' imbalance. Thus, it is crucial that balancing for primary family products take into consideration the remaining products.

Nonprimary Family Products. In the case of nonprimary family products, the problem is just the opposite of the problem encountered for primary family products. For nonprimary family products balancing has to incorporate the component assignments already committed by the primary family balancing procedure.

Placement Time Estimation. The true placement time for a printed circuit assembly is a function of the placement sequence, which includes the placement table movement, the placing head rotation speed, and the feeder bank movement. The only information we have available is the placing head rotation speed and even that is an approximation. The maximum allowable speed for the placing head rotation is determined from a component's polarity, presentation, package type, size, and pickup nozzle size. Furthermore, the placing head has 12 two-nozzle stations that are all influenced by the head's speed selection. We approximated the placement speed by obtaining the speed of the head rotation.

Products with Inherent Imbalance. In certain cases only the duplication of a component's availability among the CP-III placement machines would lead to good balance. For example, it is possible that a printed circuit assembly's side requires a placement-intensive component that greatly exceeds the total placement of the remaining components. Only an availability of that component in both of the CP-III setups would provide a shot at a reasonable balance. In our initial implementation we didn't use this approach.

Algorithm Outline

Just as in our family assignment approach we used the concepts of fuzzy sets, relationships, and fuzzy composition to balance the series CP-III loads for each printed circuit assembly side being assembled. The following is a high-level procedural outline of our balancing algorithm.

1. Define component fuzzy set c_i for $1 < i < N$
2. Sort all c_i in decreasing order
3. Initialize fuzzy relation r
4. FOR $1 < i < N$
5. IF c_i has no predetermined matching assignment m_a
6. THEN
7. $m_i = r \otimes c_i$
8. m_i Defuzzification $\Rightarrow m_a$ for c_i
9. END IF
10. Assign component c_i to machine m_a
11. Update the relation r; r = rel_update(c_i, m_a)
12. END FOR
13. Ensure that all machine constraints are satisfied.

c_i is the ith component represented by the fuzzy set c
N is the number of components to be assigned
m_i is the machine fuzzy set obtained for component c_i
m_a is an actual machine a to which the component c_i has been assigned
r describes the relation between c_i and m_i
$\otimes$ is the fuzzy operator.

Nonfuzzy sorting of fuzzy sets is based on $\sum_{i=1}^{N} W_{i,j}$ where $W_{i,j}$ is a value described for all fuzzy components c_i (described below), i is the ith component, and j is the jth product.

Fuzzy Component c

A fuzzy set c_i representing a physical component C_i is defined on universe $P = \{p_1, p_2, \ldots p_q\}$ where p_1, p_2,…p_q represent products. Thus, fuzzy set

$$c_i = \left(w_{i,1}/p_1,\ w_{i,2}/p_2,\ \ldots\ w_{i,j}/p_q\right) \tag{11}$$

where:

$$w_{i,j} = W_{i,j}/\text{norm}(C_i) \tag{12}$$

$$W_{i,j} = w\_place(C_i) \times qty\_per(C_i,p_j) \times no\_images(p_j) \times \log_{10}(demand(p_j)) \tag{13}$$

w_place is a placement time weight factor for physical component C_i
qty_per is the number of times a component C_i is placed on product p_i
no_images is the number of times product p_i appears on a single manufacturing fixture (panel)
demand is the expected volume demand for product p_i

$$\text{norm}(C_i) = \max_{j=1}^{Q}\left(W_{i,j}\right) \tag{14}$$

Q is the cardinality of universe P.

Machine Fuzzy Set m

The machine fuzzy set m is defined on the universe M = {CP3.1,CP3.2}. Consequently, the fuzzy set m_i is defined as a fuzzy set on universe M as:

$$m_1 = w_{i,1}/CP3.1 + w_{i,2}/CP3.2 \tag{15}$$

and it is obtained by $r_t \otimes c_i$, where $\otimes$ is the fuzzy composition operator of choice.

Fuzzy Relation r

Fuzzy relation r on universe R = P **x** M is used to capture the relation between the physical component C represented by fuzzy set c and machine fuzzy set m. We developed the following general equation to obtain membership values for the relation r.

$$r_{k,n} = 1 - \text{assigned\_current}_{k,n}/\text{assigned\_expected}_{k,n} \tag{16}$$

where:

$0 < k < Q$ since Q is the cardinality of universe P
$1 < n < 2$ since universe M has two elements CP3.1 and CP3.2
$\text{assigned\_current}_{k,n}$ is the current assignment for the kth product and the nth physical machine
$\text{assigned\_expected}_{k,n}$ is the expected assignment for the kth product and the nth physical machine.

If $assigned\_current_{k,n} > assigned\_expected_{k,n}$ then $r_{k,n} = 0$ ($r_{k,n}$ should be in the <0,1> interval).

We considered two possible ways to obtain values for $assigned\_current_{k,n}$ and $assigned\_expected_{k,n}$. In the first approach, we only considered the component placement time without any additional consideration for slot space limitations. In the second approach, we tried to incorporate some of the known slot constraints. The following equations show the two approaches for obtaining the values for $assigned\_current_{k,n}$ and $assigned\_expected_{k,n}$:

- Placement time only.

$$assigned\_expected_{k,n} = \left[\sum_{j=1}^{j=N \wedge C_j \notin A} \left(W_{j,k}\right) + ac_{k,n}\right] \times pp_{k,n} \quad (17)$$

$$assigned\_current_{k,n} = \sum_{j=1}^{z=N \wedge C_j \in Mach_n} \left(W_{j,k}\right) + ac_{k,n} \quad (18)$$

where:

$ac_{k,n}$ is the placement time sum for components committed to the nth machine for the kth product

$pp_{k,n}$ is the percentile portion of the kth product preferred to be consumed at the nth physical machine

$Mach_n$ is a crisp set of all physical components C assigned to the nth physical machine

$$A = \bigcup_{n=1}^{n=2} Mach_n \quad (19)$$

$W_{j,k}$ see the definition of the fuzzy component c given above

N is the number of components to be assigned.

- Placement time with slot constraints considered.

$$assigned\_expected_{k,n} = \left[\left[\sum_{j=1}^{j=N \wedge C_j \notin A} \left(W_{j,k} \times s_j\right)\right] \times avail_n + ac_{k,n}\right] \times pp_{k,n}$$

$$assigned\_current_{k,n} = \left(\sum_{j=1}^{j=N \wedge C_j \in Mach_n} \left(W_{j,k} \times s_j\right)\right) \times taken_n + ac_{k,n}$$

where:

$ac_{k,n}$ is the placement time sum for components committed to the nth machine for the kth product

$pp_{k,n}$ is the percentile portion of the kth product preferred to be consumed at the nth physical machine

$Mach_n$ is a crisp set of all physical components C assigned to the nth physical machine

$W_{j,k}$ see the definition of the fuzzy component c given above

s_j is the number of slots component C_j consumes

$avail_n$ is the number of slots available for the nth machine

$taken_n$ is the number of slots already taken at the nth machine

$A = \bigcup_{n=1}^{n=2} Mach_n$ is a crisp set containing components already assigned to some machine.

Note that the $ac_{k,n}$ identifier used in both approaches includes the predetermined components already assigned to a machine. Thus, the fuzzy machine balancer does not actively consider predetermined components for balancing, it simply incorporates their passive balancing impact into the balancing process.

The second approach is very complex and elaborate and tries to control a lot of independent measures simultaneously. Thus we selected the first approach for our prototype because we achieved much better overall balance with this approach even though we have to ensure that slot constraints are satisfied in an independent postbalancing step.

The fuzzy relation r has to be updated every time a component C is assigned to the nth physical machine. This procedure ensures that the current component assignment is going to be reflected by the fuzzy relation r. This update is done fairly quickly by recalculating the $assigned\_current_{k,n}$ value for the corresponding machine n and product p_k ($0 < k < Q$). It is obvious that the update of $assigned\_current_{k,n}$ changes the appropriate $r_{k,n}$ and hence the fuzzy relation r.

Fuzzy Composition

Although the Max-Min and Sum-Prod composition operators were investigated, the Max-Prod fuzzy composition operator performed best for our balancing algorithm, and we used the Max defuzzification approach to select a component to assign to a particular machine.

The following example illustrates our machine balancing algorithm. In this example we are trying to assign component c_{10}, and we have three products p_1, p_2, and p_3 to assemble. Components c_1 to c_9 have already been assigned to one of two placement machines CP3.1 and CP3.2

To simplify our calculations assume that $ac_{k,n}$ is 0 for all k and all n meaning there are no predetermined components on any of the products in question.

From equation 11:

$$c_{10} = w_{10,1}/p_1 + w_{10,2}/p_2 + w_{10,3}/p_3$$

and from equation 12

$$w_{10,j} = W_{10,j}/norm(C_{10})$$

If we assume that $W_{10,(1,2,3)} = (112.72, 150.0, 0.0)$ then using equations 12 and 14:

$w_{10,1} = 112.72/150.0 = 0.75$
$w_{10,2} = 150.0/150.0 = 1.0$
$w_{10,3} = 0.0$.

Thus,

$$c_{10} = 0.75/p_1 + 1.0/p_2 + 0.0/p_3.$$

Assume that after computing equations 17 and 18 we get the following values for each placement machine:

$$\text{assign\_expected} = \begin{array}{c} \begin{matrix} \text{CP3.1} & \text{CP3.2} \end{matrix} \\ \begin{bmatrix} 1713.0 & 1713.0 \\ 2000.0 & 2000.0 \\ 459.0 & 459.0 \end{bmatrix} \end{array} \begin{matrix} p_1 \\ p_2 \\ p_3 \end{matrix}$$

and

$$\text{assign\_current} = \begin{bmatrix} 1273.0 & 498.0 \\ 1782.0 & 1560.0 \\ 150.0 & 450.0 \end{bmatrix}.$$

The assign_expected matrix has the same values in both columns because we want to balance the load equally between the two placement machines for products p_1, p_2, and p_3. Assign_current shows the component balance between the two machines for components c_1 through c_9 at the current iteration of the balancing algorithm.

Using equation 16

$$r = \begin{bmatrix} 0.26 & 0.72 \\ 0.11 & 0.21 \\ 0.67 & 0.02 \end{bmatrix}.$$

Referring to steps 6, 7, 8, and 9 in our machine balancing algorithm, the following items are computed.

- Step 6. From equation 14, $m_i = r_t \otimes c_i$ using the Max_Prod fuzzy composition operator in equation 2:

$$r_{10} \otimes c_{10} = \text{Max}\left\{ \begin{bmatrix} 0.75 & 1.0 & 0.0 \end{bmatrix} \begin{bmatrix} 0.26 & 0.72 \\ 0.11 & 0.21 \\ 0.67 & 0.02 \end{bmatrix} \right\}$$
$$= \text{Max}\left[(0.19 \quad 0.11 \quad 0.0)(0.54 \quad 0.21 \quad 0.0) \right]$$

Thus,

$$m_{10} = 0.19/\text{CP3.1} + 0.54/\text{CP3.2}$$

- Steps 7 and 8. Defuzzification $\Rightarrow m_a$ for c_i is obtained by applying the Max defuzzification method to m_{10}. Thus, $\text{Max}(m_{10}) \Rightarrow m_2 = \text{CP3.2}$ meaning that component c_{10} is assigned to machine CP3.2 since it has the maximal membership value (0.54).
- Step 9. Update the relation r.

  $r = \text{rel\_update}(c_i, m_a)$ and updating

$$\text{assign\_current} = \begin{bmatrix} 1273.0 & 601.72 \\ 1782.0 & 1710.0 \\ 150.0 & 450.0 \end{bmatrix} \text{ at } i = 10$$

  makes

$$r = \begin{bmatrix} 0.26 & 0.64 \\ 0.11 & 0.15 \\ 0.67 & 0.02 \end{bmatrix}$$

  for the next iteration.

Results

For this experiment we used two manufacturing production lines at our site. The first one is denoted as line 1 and the second one as line 2. The total line volume is equivalent between the two lines. The statistics on the two lines include:

- Line 1: 27 products, 13 double-sided, 413 unique components, and on average a component appears on 3.05 products.
- Line 2: 34 products, 11 double-sided, 540 unique components, and on average a component appears on 4.69 products.

Fig. 11 shows the setup families created for the printed circuit assembly products assigned to lines 1 and 2.

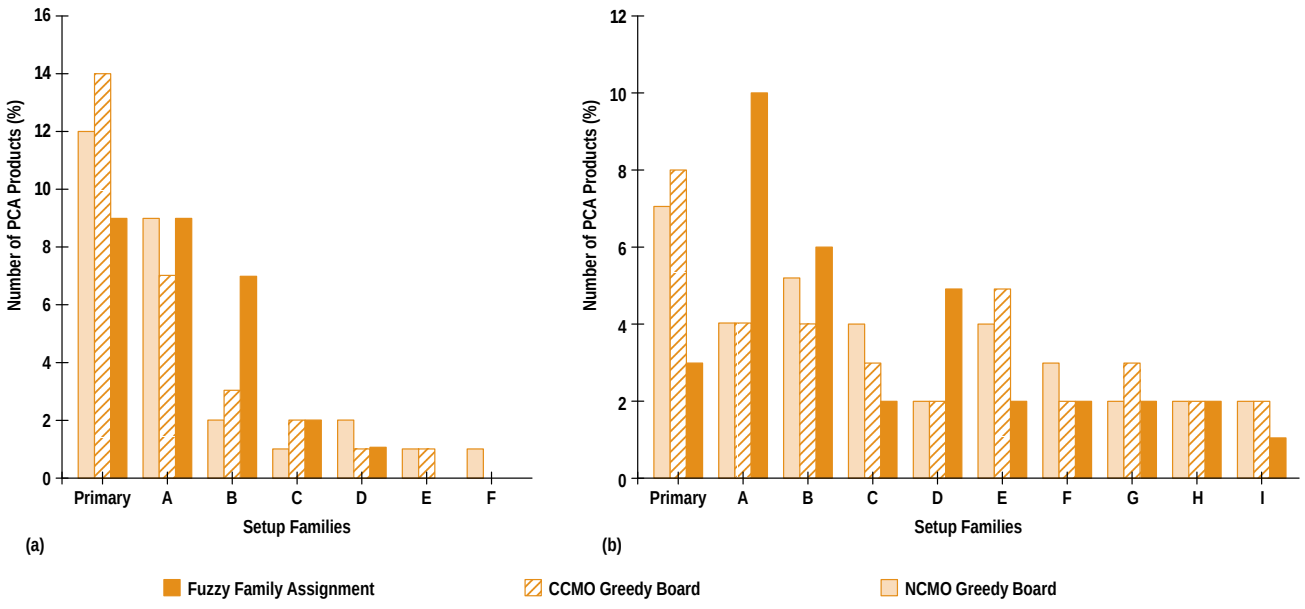


Fig. 11. The setup families created and the number of printed circuit assembly products contained in each family based on the type of family assignment algorithm used. (a) Line 1. (b) Line 2.

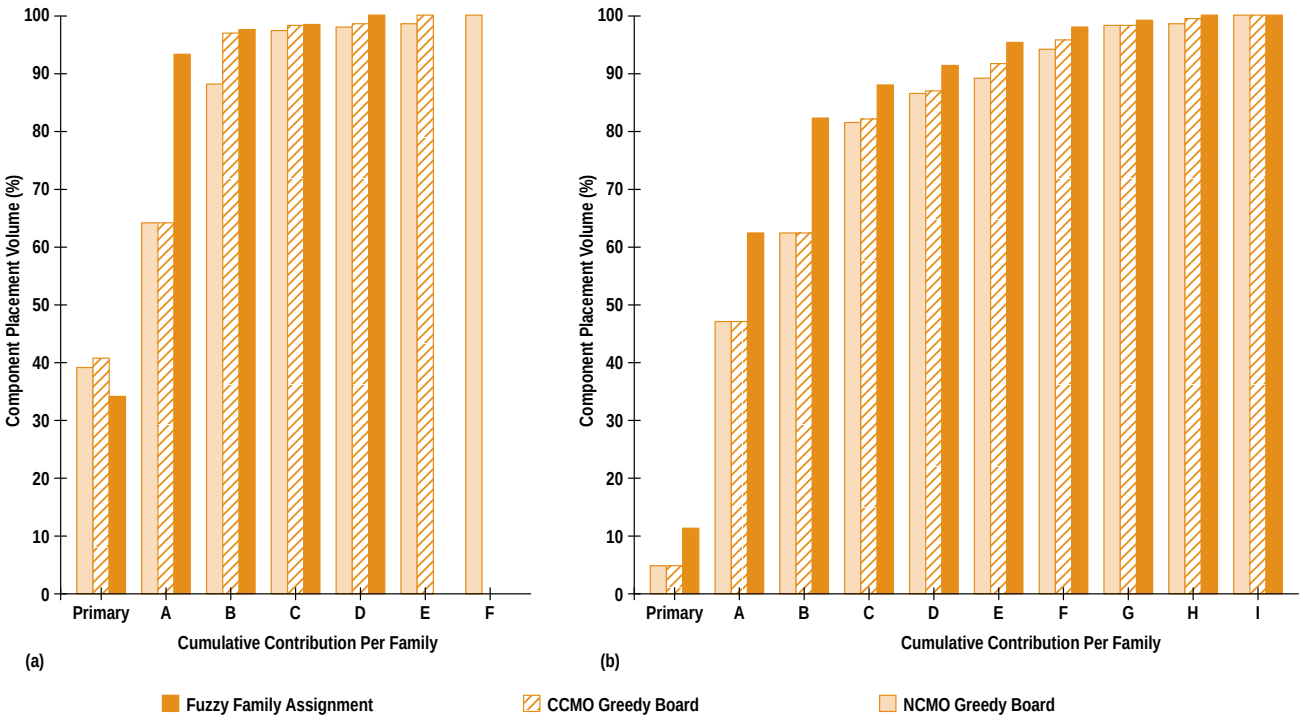


Fig. 12. A cumulative representation of the family assignments for line 1 and line 2 versus component placement volume. (a) Line 1. (b) Line 2.

Family Assignment

Fig. 12 shows the percentage of component placement volume versus the cumulative contribution for each of the family assignment techniques described in this paper.

Line 1. The results for this line were indeed phenomenal. The primary and A families together constitute 95% of component placement volume for the line. This results in no need for setup changeover for 95% of the volume for one

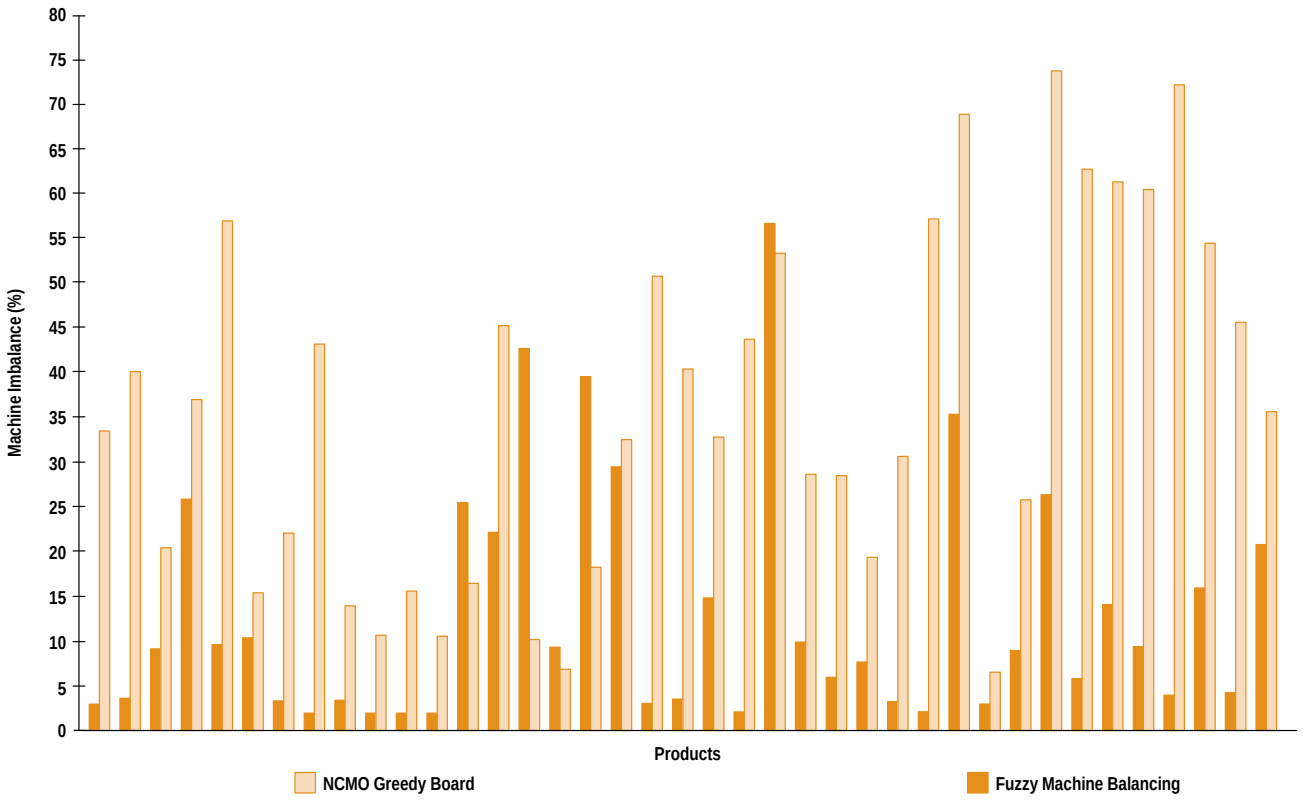


Fig. 13. Machine imbalance for line 1.

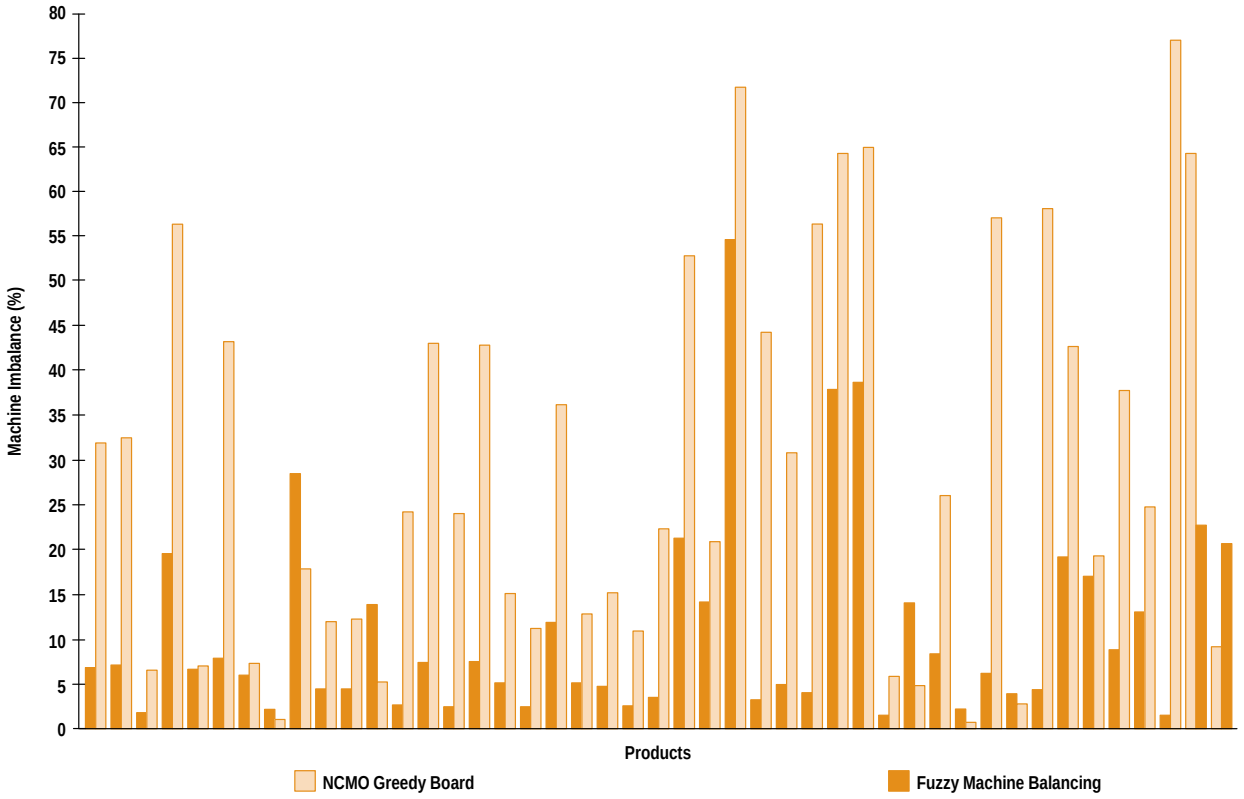


Fig. 14. Machine imbalance for line 2.

month. The 28% reduction of the number of families is a side effect of the fuzzy family assignment optimization.

Line 2. The major achievement of the fuzzy family assignment technique for line 2 was not just the moderate volume improvements over the greedy board and CCMO greedy board, but its ability to produce the same solution we obtained when we manually forced certain products into a primary family using the greedy board method. When we first investigated greedy board capabilities, we allowed hand-picked products to be forced into a family, regardless of their greedy ratio. The forced products were carefully identified based on our intuition and expertise.

Machine Balancing

Figs. 13 and 14 show the percentage imbalance for individual printed circuit assemblies manufactured on lines 1 and 2 respectively. The line 1 average imbalance was 12.75% for the fuzzy machine balancing approach and 35.9% for the balance obtained by the greedy board approach. The line 2 results are 10.73% for the fuzzy machine balancing approach and 29.58% for the greedy board approach. The families are the same ones provided by the fuzzy family assignment method.

Acknowledgment

Although many of my colleagues at HP CCMO have frequently discussed the problems described in this paper with me, I am specially grateful to my colleague Jim Nussbaumer for a lot of informal but exceptionally fruitful discussions, which have brought simplicity and elegance into our fuzzy family assignment and machine balancing approach.

References

1. T. Davis and E. Selep, "Group Technology for High-Mix Printed Circuit Assembly," *IEEE International Electronic Manufacturing Technology Symposium,* October, 1990.

2. M Jamshidi, N. Vadiee, and T. Ross, *Fuzzy Logic and Control: Software and Hardware Applications,* Prentice Hall, 1993.