Line Balancing in the Real World

Emanuel Falkenauer

Optimal Design
Av. Jeanne 19A boîte 2, B-1050 Brussels, Belgium
+32 (0)2 646 10 74
E.Falkenauer@optimaldesign.com

Abstract: Line Balancing (LB) is a classic, well-researched Operations Research (OR) optimization problem of significant industrial importance. It is one of those problems where domain expertise does not help very much: whatever the number of years spent solving it, one is each time facing an intractable problem with an astronomic number of possible solutions and no real guidance on how to solve it in the best way, unless one postulates that the old way is the best way. Here we explain an apparent paradox: although many algorithms have been proposed in the past, and despite the problem's practical importance, just one commercially available LB software currently appears to be available for application in industries such as automotive. We speculate that this may be due to a misalignment between the academic LB problem addressed by OR, and the actual problem faced by the industry.

Keyword: Line Balancing, Assembly lines, Optimization

1 Introduction

Assembly Line Balancing, or simply Line Balancing (LB), is the problem of assigning operations to workstations along an assembly line, in such a way that the assignment be optimal in some sense. Ever since Henry Ford's introduction of assembly lines, LB has been an optimization problem of significant industrial importance: the efficiency difference between an optimal and a sub-optimal assignment can yield economies (or waste) reaching millions of dollars per year.

LB is a classic Operations Research (OR) optimization problem, having been tackled by OR over several decades. Many algorithms have been proposed for the problem. Yet despite the practical importance of the problem, and the OR efforts that have been made to tackle it, little commercially available software is available to help industry in optimizing their lines. In fact, according to a recent survey by Becker and Scholl (2004), there appear to be currently just two commercially available packages featuring both a state of the art optimization algorithm and a user-friendly interface for data management. Furthermore, one of those packages appears to handle only the "clean" formulation of the problem (Simple Assembly Line Balancing Problem, or SALBP), which leaves only one package available for industries such as automotive. This situation appears to be paradoxical, or at least unexpected: given the huge economies LB can generate, one would expect several software packages vying to grab a part of those economies.

It appears that the gap between the available OR results and their dissemination in today's industry, is probably due to a misalignment between the academic LB problem addressed by most of the OR approaches, and the actual problem being faced by the industry. LB is a difficult optimization problem (even its simplest forms are NP-hard – see Garey and Johnson, 1979), so the approach taken by OR has typically been to simplify it, in order to bring it to a level of complexity amenable to OR tools. While this is a perfectly valid approach in general, in the particular case of LB it led to some definitions of the problem that ignore many aspects of the real-world problem. Unfortunately, many of the aspects that have been left out in the OR approach are in fact crucial to industries such as automotive, in the sense that any solution ignoring (violating) those aspects becomes unusable in the industry.

In the sequel, we first briefly recall classic OR definitions of LB, and then review how the actual line balancing problem faced by the industry differs from them, and why a solution to the classic OR problem may be unusable in some industries.

2 OR Definitions of LB

The classic OR definition of the line balancing problem, dubbed SALBP (Simple Assembly Line Balancing Problem) by Becker and Scholl (2004), goes as follows. Given a set of tasks of various durations, a set of precedence constraints among the tasks, and a set of workstations, assign each task to exactly one workstation in such a way that no precedence constraint is violated and the assignment is optimal. The optimality criterion gives rise to two variants of the problem: either a cycle time is given that cannot be exceeded by the sum of durations of all tasks assigned to any workstation and the number of workstations is to be minimized, or the number of workstations is fixed and the line cycle time, equal to the largest sum of durations of task assigned to a workstation, is to be minimized.

Although the SALBP only takes into account two constraints (either the precedence constraints plus the cycle time, or the precedence constraints plus the number of workstations), it is by far the variant of line balancing that has been the most researched. We have contributed to that effort in Falkenauer and Delchambre (1992), where we proposed a Grouping Genetic Algorithm approach that achieved some of the best performance in the field. The Grouping Genetic Algorithm technique itself was presented in detail in Falkenauer (1998).

However well researched, the SALBP is hardly applicable in industry, as we will see shortly. The fact has not escaped the attention of the OR researches, and Becker and Scholl (2004) define many extensions to SALBP, yielding a common denomination GALBP (Generalized Assembly Line Balancing Problem). Each of the extensions reported in their authoritative survey aims to handle an additional difficulty present in real-world line balancing. We have tackled one of those aspects in Falkenauer (1997), also by applying the Grouping Genetic Algorithm.

The major problem with most of the approaches reported by Becker and Scholl (2004) is that they generalize the simple SALBP in just one or two directions. The real-world line balancing, as faced in particular by the automotive industry, requires tackling *many* of those generalizations *simultaneously*.

3 What Differs in the Real World?

Although even the simple SALBP is NP-hard, it is far from capturing the true complexity of the problem in its real-world incarnations. On the other hand, small instances of the problem, even though they *are* difficult to solve to optimality, are a tricky target for line balancing software, because small instances of the problem can be solved *close* to optimality by hand. That is however not the case in the automotive and related industries (bus, truck, aircraft, heavy machinery, etc.), since those industries routinely feature assembly lines with dozens or hundreds of workstations, and hundreds or thousands of operations. Those industries are therefore the prime targets for line balancing software.

Unfortunately, those same industries also need to take into account *many* of the GALBP extensions *at the same time*, which may explain why, despite the impressive OR work done on line balancing, only one commercially available software seems to be currently available for those industries.

We identify below some of the additional difficulties (with respect to SALBP) that must be tackled in a line balancing tool, in order to be applicable in those industries.

3.1 Do Not Balance But Re-balance

Many of the OR approaches implicitly assume that the problem to be solved involves a new, yet-to-be-built assembly line, possibly housed in a new, yet-to-be-built factory. To our opinion, this is the gravest oversimplification of the classic OR approach, for in practice, this is hardly ever the case. The vast majority of real-world line balancing tasks involve existing lines, housed in existing factories – in fact, the target line typically needs to be *re*balanced rather than balanced, the need arising from changes in the product or the mix of models being assembled in the line, the assembly technology, the available workforce, or the production targets. This has some far-reaching implications, outlined below.

3.2 Workstations Have Identities

As pointed out above, the vast majority of real-world line balancing tasks involve existing lines housed in existing factories. In practice, this seemingly “uninteresting” observation has one far-reaching consequence, namely that each workstation in the line does have its own identity. This identity is not due to any “incapacity of abstraction” on part of the process engineers, but rather to the fact that the workstations are indeed not identical: each has its own space constraints (e.g. a workstation below a low ceiling cannot elevate the car above the operators’ heads), its own heavy equipment that cannot be moved spare huge costs, its own capacity of certain supplies (e.g. compressed air), its own restrictions on the operations that can be carried out there (e.g. do not place welding operations just beside the painting shop), etc.

3.3 Unmovable Operations and Zoning Constraints

The need to identify workstations by their position along the line (rather than solely by the set of operations that would be carried out there) is illustrated by the typical need of line managers to define unmovable operations and zoning constraints. An operation is marked as unmovable if it must be assigned to a given workstation. This is usually due to some kind of heavy equipment that would be too expensive to move elsewhere in the shop. Zoning constraints are a generalization of unmovable operations: they express the fact that an operation can only be assigned to a given (not necessarily contiguous) subset of the workstations in the line. A typical example are operations that require the vehicle to be elevated above the operators: such operations can only be assigned to workstations with enough space to contain the elevated vehicle. Zoning constraints are typical in the automotive industry – any algorithm to be applied there must support them.

3.4 Cannot Eliminate Workstations

Since workstations do have their identity (as observed above), it becomes obvious that a real-world LB tool cannot aim at eliminating workstations. Indeed, unless the eliminated workstations were all in the front of the line or its tail, their elimination would create gaping holes in the line, by virtue of the other workstations' retaining of their identities, including their geographical positions in the workshop. Also, it is often the case that many workstations that could possibly be eliminated by the algorithm are in fact necessary because of zoning constraints.

3.5 Need to Equalize Loads

Since eliminating workstations cannot be the aim of the optimization of the line, as pointed out above, it is the equalization or smoothing (indeed "balancing") of the workload among workstations that should be the practical aim of LB.

It is worth noting that the classic objective of minimization of the cycle time, i.e. minimization of the maximum lead time over all workstations, is not necessarily the same objective as load equalization. The aim of the latter usually translates into minimization of the *squared* differences between workstation loads, which means that a small increase in the maximum lead time may yield a substantial reduction in load disbalance, i.e. a better equalization of workload.

The important practical point to be made here, is that the line's cycle time is almost always given by the company's marketing that sets production targets. The maximum cycle time set by marketing cannot of course be exceeded by the line (otherwise the production target would not be met), but it is typically useless to reduce the line's cycle time below that value. In this context then, minimizing the cycle time is only required as long as it exceeds the target – once that objective is met, equalization of the workload should be pursued instead.

3.6 Multiple Operators

In many industries, in particular automotive, the product being assembled is sufficiently voluminous to allow several operators to work on the product at the same time. Since that possibility does exist, not exploiting it would lead to unnecessarily long assembly lead times, implying a reduced productivity. It is therefore often the case that several operators are active on the product simultaneously.

Once a workstation features more than one operator, the workstation's lead time ceases to be a simple sum of durations of all operations assigned to it. First of all, the workstation as a whole will need the time equal to the lead time of its "slowest" operator to complete all operations assigned to the workstation. Needless to say, since operations are indivisible chunks of work, this is certainly not equal to the sum of durations divided by the number of operators.

More importantly though, the precedence constraints that nearly always exist among the operations assigned to a workstation, may introduce gaps of idle (waiting) time between operations, whenever an operator needs to wait for another one to finish a task. These gaps significantly reduce the efficiency of the workstation and must obviously be reduced as much as possible. This transforms the initially trivial computation of a workstation's lead-time (i.e., a simple sum of operation durations) into a full-fledged scheduling problem. The problem is further significantly more complicated in presence of multi-operator operations and/or ergonomic constraints, as described below.

3.7 Multi-operator Operations

Assembly of large products such as cars sometimes requires the collaboration of *several* operators to carry out an operation. A typical example of such multi-operator operation is the mounting of bumper on a car: the part is typically large and heavy and requires two operators, one at each end of the bumper (i.e. on the left and on the right of the car), to mount it.

It would be conceivable to have a second operator assigned to that bumper operation alone, but that would be obviously very inefficient, because for all other operations the operator would be idle. It is therefore desirable to make that operator carry out other operations as well. That, however, significantly complicates the scheduling of operations within the workstation: all the operators in the workstation must be kept as busy as possible, must execute the operations in compliance with the precedence constraints, *and* must be made available *at the same time* to carry out multi-operator operations.

3.8 Ergonomic Constraints (Operator Positions)

3.8.1 Workstation level

A major difficulty in assembly of large products is that they are too bulky to be moved (elevated, rotated) easily. They are therefore kept in a given position during the whole stay in the workstation, i.e. for all the operations the workstation has been assigned.

On the other hand, the large size of the products makes it impossible to carry out certain operations unless the product is positioned in a certain way. For instance, in order to mount the exhaust system on a car, the car must be elevated - otherwise the operator would have to crawl under the car, making the operation extremely inefficient and tiring.

Also, there are situations where the access to the product is limited. For example, a workstation can perform operations taking place on the left *or* the right of the car, but not both. In other situations, the working position is imposed from the outset. For instance, when the conveyor is close to a wall on the right side in a workstation, the workstation can only be assigned operations that take place on the left side of the car.

These considerations give rise to workstation-level ergonomic constraints (WLECs):

- Each operation is defined to be performed in a defined WLEC, or 'ANY'
- Each workstation is assigned a defined WLEC, or 'ANY'
- If a workstation has a defined WLEC, then it can be only assigned operations with the same WLEC, or 'ANY'

- If a workstation has the 'ANY' WLEC, then it can "acquire" any defined WLEC by the operations it is assigned, but all of them must share the same WLEC or have the 'ANY' WLEC.

These rules simply ensure compatibility between the workstation, the operations it is assigned, and the operations among themselves. For instance, if a workstation is assigned an operation that requires the car to be elevated, then all the *other* operations assigned to that workstation must be compatible with that position, otherwise they could not be carried out (since the car *will* be elevated in the workstation, to make the first operation feasible).

The possibility to assign the undefined ('ANY') WLEC to a workstation is crucial: many modern assembly lines are highly flexible, giving the line manager the possibility *not* to impose all WLECs *a priori* but, rather, have it selected by the algorithm in the way that makes the line the most efficient.

Naturally, there may be more than one kind of WLECs at play simultaneously, in which case *all* of them must be satisfied on all workstations in any valid solution. For instance, a typical set of WLECs in the automotive industry includes the following three kinds:

- Side (left, right, front, back, or 'ANY')
- Elevation (low, mid, high, or 'ANY')
- Tilt (tilted, not tilted, or 'ANY').

3.8.2 Operator level

As we have observed in section 3.6 above, there are often several operators working on a large product at the same time. There are, however, certain restrictions that are typically imposed on the operations assigned to a workstation and their scheduling within the workstation, in order to avoid counterproductive "clashes" among operators.

First, it is usually forbidden to have two operators working at the same time in the same spot on the product. For example, during the time an operator is fixing the interior rear-view mirror in a car, it is usually forbidden to have another operator fixing the audio player, because both operators would occupy the cabin of the car at the same time, and there is not enough space in the cabin for both of them.

Second, the line manager often seeks to avoid unproductive travel of operators from one spot on the product to another, as it significantly reduces the efficiency of the workstation and hence of the whole line. This means that an operator can be "fixed" to a given spot on the product (say "front-left") and can only perform operations that take place there.

These considerations give rise to operator-level ergonomic constraints (OLECs):

- Each operation is defined to be performed in a defined OLEC, or 'ANY'
- Each *operator* in a workstation is defined to work in a defined OLEC, or 'ANY'. Note that if there are several operators in a workstation, it will have several *different* OLECs
- If an operator in a workstation has a defined OLEC, then it can only be assigned (by the within-workstation scheduling) operations of the same OLEC, or operations with the 'ANY' OLEC
- If an operator in a workstation has the 'ANY' OLEC, then it can be assigned operations with any OLEC – he or she can therefore change its working place. However, in any case,

- In any workstation, *no two* operations with the same defined OLEC can be scheduled to overlap in time, as this would result in a "clash" between several operators working in the same spot on the product at the same time.

3.8.3 Additional complexity

According to our experience, any software aiming to solve line balancing in automotive and related industries *must* support the ergonomic constraints described above. Indeed, a solution where, for instance, the exhaust system should be mounted at the same workstation as the sunroof, is simply infeasible, as the car cannot be elevated both in the high and the low position at the same time. A solution where the left floodlight should be mounted at the same time as the left front blinker is probably infeasible as well, because the two operators would be occupying each other's workspace.

However, as pointed out above, the ergonomic constraints very significantly increase the level of difficulty of the line balancing problem. Indeed,

- Workstations with the 'ANY' WLEC make NP-hard even the mere problem of finding an assignment of WLECs to these workstations that is feasible under the precedence constraints among operations with defined WLECs. For instance, if an operation with the elevation WLEC "high" is preceded by an operation with "low", itself preceded by a "high" operation, then *any* feasible solution must feature two "high" workstations with a "low" workstation in between
- OLECs may significantly constrain the within-workstation scheduling, creating idle times in many schedules. This makes the search for workstations with little idle time far more difficult. Also, multi-operator operations with OLECs defined for each of its operators can be difficult to assign, as they require the presence of all the OLECs in a single workstation.

3.9 Multiple Products

3.9.1 Multi-product Line Balancing on Average

In today's time of ever more personalized (customized) products, assembly lines assembling just one product are extremely rare. This is especially true for the automotive industry, where numerous variants of the same car, or even different cars, are routinely assembled on the same line. A line balancing tool for automotive and related industries must therefore support multiple products on the line.

In a multi-product line, an operation may be carried out on only a subset of the products assembled in the line. For instance, suppose that two products, A and B, are assembled on the line: operations that must be carried out on A but *not* on B, will only be performed when an A is actually being assembled, and vice-versa.

This is often taken into account by the obvious approach of replacing operation durations by their average durations over the whole production, and balancing the line *as if* it assembled just one product. However, this "balancing of average durations" is fundamentally flawed in presence of multi-operator workstations.

Indeed, the objective of the balancing is to obtain a line that would be well balanced on average, which implies that we need to compute the average lead time of each workstation. The "balancing of average durations" is based on the (correct) observation that the average of sums of durations is equal to the sum of averages. Yet as pointed out in section 3.6 above, with multiple operators per workstation, the lead time is *not* a simple sum of operation durations, but the result of a full-fledged within-workstation scheduling. Since the average of lead times obtained by within-workstation scheduling for each product is *not* necessarily equal to the lead time obtained by within-workstation scheduling of operations of average durations, the premise behind the "balancing of average durations" does not hold and the approach is flawed: the resulting lead times would most probably *not* reflect reality.

In short, in order to obtain the true average workstation lead times, the lead times must be computed separately for all the products being assembled, with true operation durations, and only then averaged according to the respective product percentages.

3.9.2 Peak Time Reduction

As pointed out above, even the simple balancing *on average* becomes difficult in presence of multiple operators in workstations, but the difficulties do not stop there. One of the major difficulties in running a multi-product assembly line is coping with "rare" products, i.e. products that are assembled relatively seldom.

Suppose for instance, that a given product C represents only 10% of the total production. Consequently, C weighs only 10% in the average of workstation lead times. With such a small percentage, the influence of the product in the line balancing on average can easily be diluted to a point where it is neglected. However, suppose that the operations pertaining to assembly of C are assigned in such a way that they induce a workstation's lead time *triple* the line's cycle time. Such an assignment would cause an inordinate amount of time being spent in that workstation each time C would be assembled, and would probably lead to a long stoppage of the whole line once in every ten products assembled – clearly a disastrous performance.

One way of avoiding these problems is to take into account the peak time at each workstation, i.e. the maximum lead time over all products. These peak times *must* be kept at a reasonable level, which can only be achieved by taking them into account in the optimization, together with the average balance. Becker and Scholl (2004) call this approach "horizontal balancing".

3.10 Drifting Operations

While "rare" products are a difficulty that must be dealt with by taking peak times into account, extremely long yet "rare" *operations* are a different kind of nuisance. In many assemblies, in particular in the automotive industry, some operations take so long and are sufficiently "rare" that they are split over several consecutive workstations.

Consider for instance the not-so-distant past when CD audio players were a rarity in a car. In that time, the technology was not very advanced either, so mounting the CD player required a very long assembly time, a multiple of the line's cycle time. The standard way of dealing with that kind of operation is to split it over several consecutive workstations. The idea is simple: at the end of the cycle time on the first workstation to which the operation is assigned, the operation continues while the product enters the next workstation, where it is taken up by another operator – in other words, the operation *drifts* into the next workstation. This is possibly repeated into a third workstation, etc.

In another variant of drifting, the operator drifts with the operation, i.e. he or she visits several consecutive workstations. When the operation is finished, the operator leaves the product and walks back to the workstation where the operation begun. If the operation is indeed sufficiently "rare", all of this can be achieved before the next product requiring the operation enters its first workstation. For instance, an operation that pertains to only one in every three products being assembled on the line, could drift over a total of up to three workstations, provided the products are sequenced consequently.

Needless to say, handling drifting operations represents an additional difficulty in a line balancing algorithm. This is particularly the case when ergonomic constraints (see section 3.8 above) are being handled as well, as assigning a drifting operation to a workstation directly influences the ergonomic constraints on the successive workstations the operation will drift into. Nevertheless, drifting operations are quite pervasive in automotive and related industries, so a tool intended for those industries should be able to handle them.

4 Summing Up

4.1 Complexity of the Problem and Time Requirements

The conclusions in section 3 stem from our extensive contacts with automotive and related industries, and reflect their true needs. Other "exotic" constraints may apply in any given real-world assembly line, but a line balancing tool for those industries must be able to handle at least those aspects of the problem. This is very far from the "clean" academic SALBP, as well as most GALBP extensions reported by Becker and Scholl (2004). In fact, such a tool must *simultaneously* solve several NP-hard problems:

- Find a feasible defined replacement for all undefined ('ANY') ergonomic constraints on workstations, i.e. one compatible with the ergonomic constraints and precedence constraints defined on operations, as well as zoning constraints and possible drifting operations
- Solve the within-workstation scheduling problem on all workstations, for all products being assembled on the line
- Assign the operations to workstations to achieve the best average balance, while keeping the peak times at a manageable level.

Clearly, the real-world line balancing problem described above is extremely difficult to solve. This is compounded by the size of the problem encountered in the target industries, which routinely feature assembly lines with dozens or hundreds of workstations with multiple operators, and hundreds or thousands of operations.

Yet in order to be practically usable, a tool solving the problem must nevertheless be very fast. This is because the line manager has many possibilities of imposing *a priori* any of the constraints discussed above. This enables them to fashion the line to their exact needs in an iterative process, where the influence of an *a priori* choice is assessed by evaluation of the resulting line. Needless to say, the algorithm needs to offer short response times (minutes) to make that iterative process practically feasible.

The uncanny difficulty of the problem, and the practical need to solve it quickly, may explain why just one commercially available software, OptiLine, appears to be available. OptiLine uses the Grouping Genetic Algorithm (GGA) proposed by Falkenauer (1998), to solve the problem with all the aspects discussed above while supplying high-quality solutions in short computation times. We cannot give a detailed description of the GGA technique here, for obvious space reasons – the interested reader should consult the Falkenauer (1998) book.

4.2 The advantage of the Grouping Genetic Algorithm

The GGA used in OptiLine has a fundamental advantage over other methods in handling this kind of complex problems, which may explain its success. Since the optimization itself is performed by gradual improvement through exchange of information between various solutions to the problem (as is the case in any Genetic Algorithm), the GGA has the “liberty” of constructing the various solutions under the sole constraint of feasibility (i.e. compliance with the numerous constraints), *without* much regard to their quality. This is in stark contrast to constructive methods that must care for the quality of the solution, as well as its feasibility, *during* the very process of its construction.

This gives the GGA the advantage of extreme flexibility: as long as feasible solutions can be found, *whatever* their quality, the GGA will deliver. That flexibility makes it possible to take into account constraints as diverse and as difficult to satisfy as those identified above.

5 Conclusions

We have identified a number of aspects of the line balancing problem that are vital in industries such as automotive, yet that have been either neglected in the OR work on the problem, or handled separately from each other. According to our experience, a line balancing tool applicable in those industries must be able to handle *all* of them simultaneously. That gives rise to an extremely complex optimization problem.

The complexity of the problem, and the need to solve it quickly, may explain why there appears to be just one commercially available software for solving it, namely OptiLine by Optimal Design. More information on OptiLine, including its rich graphic user interface, is available at http://www.optimaldesign.com/OptiLine/OptiLine.htm.

References

1 Becker C. and Scholl, A. (2004) `A survey on problems and methods in generalized assembly line balancing', *European Journal of Operations Research*, in press. Available on-line at http://dx.doi.org/doi:10.1016/j.ejor.2004.07.023. Journal article.

2 Falkenauer, E. and Delchambre, A. (1992) `A Genetic Algorithm for Bin Packing and Line Balancing', *Proceedings of the 1992 IEEE International Conference on Robotics and Automation*, May 10-15, 1992, Nice, France. IEEE Computer Society Press, Los Alamitos, CA. Pp 1186-1192. Conference proceedings.

3 Falkenauer, E. (1997) `A Grouping Genetic Algorithm for Line Balancing with Resource Dependent Task Times', *Proceedings of the Fourth International Conference on Neural Information Processing (ICONIP’97)*, University of Otago, Dunedin, New Zealand, November 24-28, 1997. Pp 464-468. Conference proceedings.

4 Falkenauer, E. (1998) *Genetic Algorithms and Grouping Problems*, John Wiley & Sons, Chichester, UK. Book.

5 Garey M. R. and Johnson D. S. (1979) *Computers and Intractability - A Guide to the Theory of NP-completeness*, W.H.Freeman Co., San Francisco, USA. Book.