



A matching approach for replenishing rectangular stock sizes

FJ Vasko^{1,2*}, JA McNamara², RN Parkes², FE Wolf² and LR Woodyatt²

¹Kutztown University, Kutztown, USA; and ²Bethlehem Steel Corporation, Bethlehem, USA

Consider a replenishment problem in which several different rectangular sizes of material are stocked. Customers order rectangles of the material, but the rectangles ordered have a range on specified width as well as on specified length. To satisfy the customer requirements, the stock material can be cut *once* longitudinally in order to satisfy two customer requirements or not cut at all, that is, an entire stock piece of material is used to satisfy one customer requirement. If an exact match is impossible in the current planning period, the unused material must be returned to stock— an expensive and undesirable situation. In this paper, a nonbipartite weighted matching problem formulation will be given for determining the replenishment requirements of rectangular stock sizes. Then, a hybrid solution approach, capable of solving real applications (typically up to 3000 nodes) efficiently, will be discussed. This solution was implemented in September 1998 and has operated successfully since then.

Keywords: graph matching; practice of OR; cutting stock; heuristics

Introduction

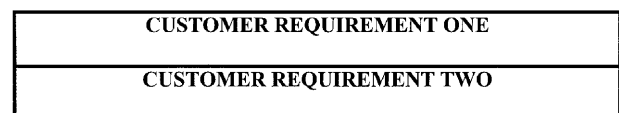
In this paper, an application will be discussed in which the replenishment requirements for rectangular stock material must be determined. This application is interesting because customers specify an acceptable range on both the width and length of their rectangular requirements. The customer requirements are mapped into the stock material such that, at most, two customer requirements can be satisfied from one stock piece of material. The material length for both the customer orders and the stock sizes is much longer than it is wide, that is, there is only one orientation for cutting the material. Therefore, at most, one longitudinal cut is made in the stock material.

A severe limitation of this application is that the stock piece must be used completely, that is, there can be no scrap loss other than that caused by the cut. If an exact match is impossible in the current planning period, then the unused material must be returned to stock— an expensive and undesirable situation. For productivity and yield reasons at the producing facility, the production of wide stock material is desired. For this reason, the majority of stock material is cut longitudinally into two customer requirements (both requirements can be for the same customer). Typical cutting patterns are given in Figure 1.

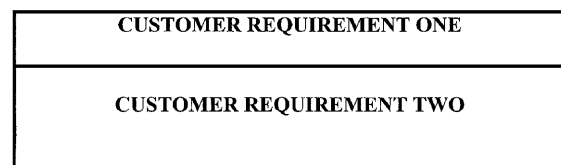
Firstly, the problem will be described and the initially-implemented replenishment system will be reviewed. Then, the application of customer requirements to stock sizes

will be formulated as a nonbipartite weighted matching problem. This will be followed by a brief literature review. Next, a new replenishment system based on an efficient hybrid solution approach for the matching problem that combines constraint-based ideas with a shortest augmenting path algorithm will be discussed. Finally, implementation aspects of the replenishment system based on the hybrid matching algorithm will be given and some conclusions drawn.

PATTERN #1—SYMMETRIC CUT



PATTERN #2—ASYMMETRIC CUT



PATTERN #3—NO CUT (SINGLE APPLICATION)



Figure 1 Typical cutting patterns.

*Correspondence: Dr FJ Vasko, Homer Research Laboratories, Bethlehem Steel Corporation, Bethlehem, PA 18016, USA
E-mail: vasko@kutztown.edu

Problem description

In this application, there are about 30 different categories of material that are stocked as rectangles. Several rectangular sizes are stocked for each category of material, but all stock material, regardless of category, has the same thickness. The approach used to determine what rectangular sizes to stock was analogous to the one discussed in Vasko and Wolf.¹

There are several goals that need to be addressed when mapping customer requirements into stock sizes. A primary concern is to minimise the number of times material must be returned to inventory after a longitudinal cut has been made to a stock rectangular piece, that is, cutting a stock piece longitudinally into two pieces but having to return one of the pieces to inventory because there is no immediate customer requirement for it. This results in excessive material handling. The most desirable situation is to cut a stock rectangle into two rectangles such that both rectangles can be applied to current customer orders, and if possible to the same customer order.

Pairing two customer order requirements into one stock rectangle is favored over mapping only one customer requirement into a stock rectangle (the last pattern in Figure 1 is one customer requirement mapped to a stock rectangle). When only one customer requirement is mapped to a stock rectangle, this is called a 'single' application. For productivity and yield reasons at the producing facility, replenishment and use of wide stock rectangles is definitely preferred to the replenishment and use of narrow width stock rectangles.

In order to avoid returning material to inventory, a requirement for a near-term future customer order, usually the next week, is allowed to be matched in a stock rectangle with a requirement for a current week customer order.

In summary, the solution to the replenishment problem, while meeting all dimensional constraints, must address the following goals given in order of importance:

1. the current week order requirements matched together;
2. the current week order requirement matched as a single application;
3. the current week order requirement matched with a future week order requirement;
4. for productivity reasons, a match in a large width stock rectangle is preferred over a match in a narrower width stock rectangle;
5. for logistical reasons, a match involving two customer rectangle requirements for the same order is preferred over a match involving two different orders.

Initial replenishment system

As part of a large comprehensive production planning system that resides on a mainframe computer, there was a module used to determine replenishment quantities for the

rectangular stock sizes. In this initial replenishment system, customer requirements were assigned to rectangular stock sizes based on a simple greedy heuristic that sequentially paired order requirements together into stock sizes. This sequential approach did not undo pairings that were made early in the process even if they caused very poor matches to be made later in the process or if they resulted in customer requirements not being able to be matched into rectangular stock sizes at all. All customer requirements that could not be matched into rectangular stock sizes represent requirements that are cut from a wide stock size and the unused rectangle being returned to inventory. Because of its non-global nature, the existing system's replenishment recommendations tended to be higher (over-estimated) than what was actually required.

This system was executed daily as part of a daily execution of the larger production planning system. For each of the categories of material (about 30), six replenishment problems were solved daily. For each material category, six two-week overlapping time periods were analysed. For example, if the current week was week 32 of the calendar year, then the first replenishment problem would treat week 32 orders as current week orders and week 33 orders as future orders. The replenishment solution indicates *how many rectangles of each stock size are needed* to meet week 32 orders. Any week 33 orders that were assigned are deleted from week 33 requirements, and any material returned to inventory would be available as stock rectangles. If rectangles were returned to inventory in the first replenishment problem, then week 33 orders are applied to these rectangles *before* the second replenishment problem is defined. The second replenishment problem has week 33 requirements as the current week and week 34 requirements as future orders. Continuing in this manner, the sixth replenishment problem has week 37 requirements as the current week and week 38 requirements as future orders.

By looking out six time periods and executing the program daily, any changes in the pattern of customer orders are detected in a timely manner. However, because the existing system consistently overestimated the inventory requirements, an alternate approach was sought by management.

A nonbipartite weighted matching formulation

As an alternative to the greedy-type solution approach originally implemented, a graph-theoretic matching formulation appeared to be more appropriate. To formulate a nonbipartite weighted matching problem for mapping customer rectangular requirements into stock rectangle sizes, a complete graph $G = (V, E)$ with node set V and edge set E is defined based on the goals discussed earlier. By complete graph we mean that each pair of distinct nodes is connected by an arc. There is one node for each current week customer requirement and one node for each

customer requirement associated with an order from the next schedule week. Also, there is a node for each current week customer requirement that can be met from a stock rectangle as a 'single' application. That is, a 'single' node matched with a current week customer requirement would represent a 'single' application.

Without loss of generality, the number of nodes, N , in G can be taken to be an even number. A subset $M \subset E$ is called a perfect matching if no two edges of M are incident to the same node and every node is incident to an edge in M . The edge $e_{ij} \in E$ that connects nodes i and j has a 'cost' c_{ij} (non-negative) associated with it. This cost, a weighted composite of the goals discussed previously, is a measure of how well nodes i and j match into a stock size when the stock sizes are considered in a preferred order, that is, two customer requirements might form a feasible match for more than one stock size. *The nonbipartite weighted matching problem minimises the sum of the c_{ij} over all perfect matchings of G .*

Since this is a complete graph, all nodes are connected by an arc and a cost must be defined for each arc. If two customer requirements for current week orders (each is a node in the graph) cannot be matched into any stock rectangular size, then a large positive number is assigned as the corresponding arc cost and this candidate match is referred to as a 'no match'. If this match would be in the solution, despite its high cost, then it represents each of these two customer requirements being cut from wide stock sizes and two unused rectangles being returned to inventory.

All future-to-future, future-to-single, and single-to-single candidate matches are considered feasible: however, care must be taken in assigning a cost to these matches in order to avoid undesirable solutions. For example, two matches, each involving a current week order with a future week order, is preferred to two current week orders matching as a 'no match' and the two future week orders matching together.

Literature review

The problem introduced above is really a special type of cutting stock problem. The problem is unique in the fact that customer rectangular requirements are not fixed, but have a range on width as well as length requirements. The cutting patterns although very simple, either one or two customer pieces per stock rectangle, have the unique constraint that no scrap is allowed, that is, customer orders must fit exactly within their allowable width and length range.

In Vasko *et al.*,² a one dimensional cutting stock algorithm is discussed that is used to assist customers in deciding what size and how many master steel coils to order and what slitting patterns to use on these coils. Vasko and Wolf¹ developed a procedure for determining what

master rectangular sizes to stock and how to cut the master rectangles into smaller rectangles.

We are aware of only one paper that uses a matching approach in order to solve a cutting stock problem. Fritsch and Vornberger³ solved a special type of two-dimensional cutting stock problem by solving a series of weighted nonbipartite matching problems. At each step of their procedure, customer rectangle requirements are matched creating meta-rectangles. These meta-rectangles are then matched to create new, larger meta-rectangles.

In fact, there are very few reported applications that are formulated as weighted nonbipartite matching problems and all the reported applications, as in Fritsch and Vornberger,³ repeatedly solved a number of matching problems. Wark *et al.*⁴ used repeated matching on an aircrew scheduling problem. Barnhart and Daeki⁵ used repeated matching to solve a vehicle routing problem. Vasko *et al.*⁶ used repeated matching to solve one stage of a multi-stage approach for scheduling continuous casters in the steel industry.

Matching-based solution procedures

To solve the matching problem described above, the natural choice was the FORTRAN code provided in the book by Burkard and Derigs.⁷ This was an obvious choice because we had extensive experience using the code in one stage of a multi-stage scheduling strategy.⁶ Specifically, repeated matching problems are solved in order to group customer requirements for steel products into compatible production batches.

This code, authored by G Kazakidis, uses a shortest augmenting path algorithm that was adapted from a labeling and shrinking procedure developed by Edmonds and Johnson⁸ for solving the Chinese Postman Problem. The validity of this approach for the nonbipartite weighted matching problem was demonstrated by Derigs.⁹ A thorough discussion of the nonbipartite weighted matching problem as well as the outline of an algorithm, based on a primal-dual method, can be found in Papadimitriou and Steiglitz.¹⁰

As mentioned in the last section, this replenishment application involves approximately 30 different material categories, and six matching problems, per category, that are to be solved daily. The matching problem for the highest volume category of material typically involved a graph with between 1000 and 3000 nodes. In testing the system on a 150 MHZ PC, this problem alone required about 26 minutes of execution time. Even with the mainframe being slightly faster, because of time constraints, when the system would be executed on the mainframe in production mode, a more efficient solution procedure needed to be implemented.

After discussing our need, via e-mail, to solve large non-bipartite matching problems with Derigs,¹¹ he generously

sent us a FORTRAN code, based on his 1988 paper.¹² However, as Derigs himself indicated,¹¹ and our computational experience authenticated, this code did not perform significantly faster than the code contained in Burkard and Derigs⁷ for our particular application.

At this point the following four options were considered:

1. Code the approach of Derigs and Metz¹³ that is based on a modified shortest augmenting path method that uses a new assignment start procedure and a two-phase strategy that should be faster than his earlier codes.
2. Code an algorithm based on the primal-dual method discussed in Papadimitriou and Steiglitz¹⁰ for which we had no computational results.
3. Decompose large matching problems into two or three matching problems, solving each one with the shortest augmenting path code, and then linking the answers together.
4. For large matching problems, develop an approach that matches the most constrained customer requirements until the problem size has been adequately reduced, then solve the remaining (smaller) problem using the shortest augmenting path code. Sufficiently small problems would continue to be solved entirely by the shortest augmenting path code.

An empirical analysis of the matching problems generated for the 30 material categories revealed that typically only three materials had matching problems with more than 200 nodes, and only one material had matching problems with more than 500 nodes. The average size (over 30 materials and six problems per material) matching problem had about 100 nodes.

Based on this information and the fact that minimising material returned to stock was a primary goal, option four was implemented. Option four was selected because the vast majority of the matching problems we needed to solve could be solved directly using the existing shortest augmenting path code that we had successfully used before. For large problems, we could reduce the size of the problem by matching the most constrained customer requirements until the remaining problem size was small enough to solve quickly using the shortest augmenting path code. In a sequential procedure, addressing the most constrained customer requirements first, that is the customer requirements that do not match very well into rectangular stock sizes, reduces the amount of material that has to be returned to inventory.

For large matching problems, this constraint-based (CB) approach first identifies the current week customer requirement that has the fewest candidate matches, that is the most constrained, and the lowest cost match among its candidates is made. This continues until either all current week customer requirements have been matched, or until the number of unmatched nodes that still have candidate matches available, drops below a certain threshold

number of nodes. If there are still current week customer requirements to be matched when the number of unmatched nodes drops below the threshold, then the unmatched nodes are input to the shortest augmenting path routine for matching. An outline of this hybrid constraint-based/shortest augmenting path approach (the hybrid matching solution approach) is given in steps 1 to 6.

Outline of the hybrid matching solution approach

- Step 1:* Given a matching problem, if the number of nodes does not exceed the threshold, call the shortest augmenting path routine. Otherwise, go to Step 2.
- Step 2:* For each node, determine the number of feasible candidates it has for matching.
- Step 3:* From among nodes corresponding to current week orders, select the node that has the smallest positive number of feasible candidates for matching and match it with its lowest cost match.
- Step 4:* Update the number of feasible matches remaining for all unmatched nodes. Also, update the number of unmatched nodes corresponding to current week orders that still have feasible candidates for matching and the total number of unmatched nodes that still have feasible candidates for matching.
- Step 5:* If there are no unmatched nodes corresponding to current week orders, then terminate. Otherwise, go to Step 6.
- Step 6:* If the total number of unmatched nodes that still have feasible candidates for matching is greater than the threshold, go to Step 3. Otherwise, call the shortest augmenting path routine to solve the matching problem for the unmatched nodes that still have candidates for matching.

A computational study was performed to determine where the threshold should be set. Table 1 shows the relationship between threshold and execution time. For example, if the threshold was set at 600, then the average execution time for all 30 material types was about 228 seconds.

Table 1 Threshold value *vs* execution time

Threshold value	Average execution time (seconds on 150 MHz PC)
2500	1800
2100	960
1800	672
1500	540
600	228
300	120
200	102
100	96

Only one material category was impacted by a threshold value greater than 500 and only three material categories were impacted by a threshold value greater than 200. For the highest volume material category, as the threshold value decreased to 200, the matching solution remained either essentially the same or, *at most*, one additional rectangle was returned to stock. Below a threshold of 200, the matching solution for the highest volume material category rapidly degraded. The next two largest volume material categories behaved in a similar manner as the threshold value was decreased.

Based on these results, it was decided to set the threshold at 200 for production purposes. Since the average matching problem size is about 100 nodes and typically only three material categories have problem instances exceeding 200 nodes (only one above 500 nodes), it is not surprising that a threshold of 200 gave both excellent solution results as well as execution times. In other words, the execution time of the program was significantly decreased with negligible degradation in solution quality.

Implementation of the hybrid matching solution approach

A replenishment system based on the hybrid matching solution approach discussed in the previous section was tested on the mainframe. The system was tested setting the threshold value high enough so that the constraint-based logic was not invoked *vs* a threshold of 300, 200, and 100. The results were analogous to the PC test results with a 200 threshold giving the best results and total CPU reduction of about 74% compared to solving the problem entirely using the shortest augmenting path code (This also implies a 74% reduction in CPU charges). It was decided to implement this system with a threshold of 200.

To build user confidence in the new replenishment system, both the original and the new matching-based system were executed in parallel for a short period of time during which the new system demonstrated improved performance. The new matching-based replenishment system has been in routine use since September 1998 and has proven to be both an effective and efficient method for providing useful information regarding the replenishment of rectangular stock sizes. In particular, it provides a more realistic estimate of replenishment requirements thus reducing inventory and increasing customer service.

In fact, suggestions have been made by the user to enhance the procedure, and have, in all cases, simply required parameter changes to the model. In other words, as the user's utility function and production planning goals became better known to us, we were able to bring the matching model more in line with reality largely by adjusting the function used to calculate edge weights in the graph.

Conclusions

A system was implemented for determining replenishment requirements for rectangular stock sizes based on an intuitive greedy type algorithm recommended by the client. In production use this approach tended to over-estimate inventory requirements for the rectangular stock sizes and so an alternative approach that would more accurately estimate replenishment requirements was sought.

In response to this request, a nonbipartite graph matching formulation and hybrid solution approach was implemented in September 1998. The fact that this matching-based replenishment system is less intuitive than the original system is offset by its more accurate inventory replenishment estimates

References

- 1 Vasko FJ and Wolf FE (1994). A practical approach for determining rectangular stock sizes. *J Opl Res Soc* **45**: 281–286.
- 2 Vasko FJ, Wolf FE, Stott KL and Ehrsam O (1992). Bethlehem Steel combines cutting stock and set covering to enhance customer service. *Mathe Comp Mode* **16**: 9–17.
- 3 Fritsch A and Vornberger O (1995). Cutting stock by iterated matching, operations research proceedings. In: Design U, Batham A and Drexl A (eds). *Selected Papers of the International Conference on OR94*. Springer-Verlag: Germany, pp 92–97.
- 4 Wark P, Holt J, Ronnqvist M and Ryan D (1997). Aircrew schedule generation using repeated matching. *Eur J Opl Res* **102**: 21–35.
- 5 Barnhart C and Daeki K (1995). Routing models and solution procedures for regional less-than-truckload operations. *Annals of Opns Res* **61**: 67–70.
- 6 Vasko FJ, Stott KI, Wolf FE and Woodyatt LR (1996). An Optimization-based Approach for Scheduling Continuous Casters, 14th Triennial Conference of The International Federation of Operational Research Societies (IFORS), Vancouver, B.C., Canada, July 8–12.
- 7 Burkard RE and Derigs U (1980). *Assignment and Matching Problems: Solution Methods with FORTRAN-Programs*. Springer-Verlag: Berlin, Germany.
- 8 Edmonds J and Johnson EI (1973). Matching, euler tours and the chinese postman. *Mathe Program* **5**: 88–124.
- 9 Derigs U (1979). A shortest augmenting path method for solving minimal perfect matching problems, Report 79-06, Mathematisches Institut der Universitat zu Koln.
- 10 Papadimitriou CH and Steiglitz K *Combinatorial Optimization: Algorithms and Complexity*. Prentice-Hall, Inc.: Englewood Cliffs, NJ.
- 11 Derigs U (1997). private communication.
- 12 Derigs U (1988). Solving non-bipartite matching problems via shortest path techniques. *Annals of Opns Res* **13**: 225–264.
- 13 Derigs U and Metz A (1991) Solving (large scale) matching problems combinatorially. *Mathe Program* **50**: 113–121.

Received September 1998;
accepted November 1999 after three revisions