PRODUCTION MIX PROBLEMS: FORMULATION AND SOLUTION STRATEGIES

Graeme Warren[1]
Department of Industrial and Systems Engineering
University of Pretoria

ABSTRACT

This paper examines the sub-optimality of generalized greedy heuristics for solution of a certain well-known production mix problem formulation. The drawbacks of *not* modelling routing in the mix problem formulation are discussed, and an alternative mix problem formulation (with associated solution strategies) is proposed.

OPSOMMING

Hierdie artikel verduidelik die sub-optimaliteit van 'n algemene gulsige heuristiek vir 'n sekere bekende produkmengsel probleemformulering. Die nadele van weglating van produk roete inligting in die formulering is bespreek, en 'n alternatiewe formulering (met geassosieerde oplossingsstrategiëe) is voorgestel.

[1] This research was supported by a visiting assistant professorship from the School of Industrial Engineering, Purdue University, West Lafayette, Indiana, USA, and a research grant from the University of Pretoria.

1. INTRODUCTION

The production mix problem (PMP) that we consider is a mathematical program that aims to identify, in multi-workstation and multi-product manufacturing environments, the proportion that each product should form of the total production volume in order to maximize production revenues over a finite-length planning period. The optimal mix is constrained by market demand and workstation capacities.

Formally, the PMP is defined as follows:

$$PMP: \quad \max \sum_i r_i x_i \tag{1}$$

$$\text{subject to:} \quad \sum_i t_{ij} x_i \le C_j \ \forall j \tag{2}$$

$$x_i \le D_i \ \forall i \tag{3}$$

$$x_i \in \aleph_+ \forall i \tag{4}$$

where x_i denotes the quantity of type i product that should be produced, r_i denotes the revenue accruing from production of one unit of type i product, t_{ij} denotes the capacity consumption (i.e., time) per unit of product i on workstation j, C_j denotes the capacity available on workstation j, and D_i denotes the maximum market demand for product i. $\aleph_+$ denotes the non-negative integers.

The objective function (1) requires that the total revenue be maximized. Constraint set (2) requires that the capacity constraints of each workstation be respected. Constraint set (3) requires that the market demand be treated as an upper bound on production of each product type (this will not be appropriate in some environments, notably in produce-to-stock environments; in such cases constraint set (3) is discarded). Finally, constraint set (4) requires that production volumes be integer-valued and non-negative. The assumptions are as follows: there are at least two products; there are at least two workstations;

$$0 < C_j < \infty \ \ \forall j; \quad 0 < r_i < \infty \ \ \forall i; \quad 0 \le t_{ij} < \infty \ \ \forall i, j; \quad \sum_i t_{ij} > 0 \ \ \forall j, \text{ and } D_i > 0 \ \ \forall i .$$

When constraint set (4) is relaxed to $x_i \in \Re_+ \forall i$ we call the resultant program the *relaxed* PMP. The relaxed PMP is a linear program (LP).

The work of Goldratt [9,10] on the "theory of constraints" has increased general awareness of the product mix problem in the last several decades. The theory of constraints (TOC) holds that production environments can be effectively managed by identifying and managing constraining bottleneck workstations, and has served to popularise some well-known basic results from queueing theory among production management professionals. One of the key issues in constraint-based production management is the identification of bottleneck

workstations, and the use of these bottleneck workstations as the basis for aggregate capacity planning and product mix calculations.

The first thing to notice about the PMP is that it ignores several important issues: for example, setups, routing, date targets, batching, priorities, and work in process constraints are not incorporated in the formulation. Many alternative definitions of the product mix problem have been proposed in the literature. For example, Khouja, Mehrez and Rabinowitz [13] consider a similar formulation with a convex objective function; also, Kasilingam [12] considers a stochastic demand formulation of the problem with alternative process plans. For computational tractability, few formulations incorporate all the practical issues that must be resolved in an actual multi-product and multi-workstation production environment as scheduling and sequencing problems are notoriously intractable (see, for example, Garey and Johnson [7]). The PMP is championed by Goldratt as an easily-solvable formulation that contains enough useful information to allow the calculation of near-optimal product mixes. Even though it is sometimes useful to restrict production quantities to the non-negative integers in discrete part manufacturing, the heuristic approach of solving the relaxed PMP is unsurprising. Posnack [25] claims that the real-valued solution of the relaxed PMP can be obtained and rounded up or down with little negative practical consequence in many cases.

In early work, Goldratt proposed the use of a greedy heuristic for finding solutions to the PMP. Several articles (Fredenhall and Lea [6], Maday [18], Posnack [25], Lee and Plenert [15], and Plenert [24]) have compared the performance of the greedy heuristic and other algorithms for solving the PMP. Lee and Plenert [15] showed that the greedy heuristic sometimes provides sub-optimal solutions. Lee and Plenert's claim was later erroneously disputed by others. This paper explains *why* greedy heuristics may fail to yield optimal solutions for some instances of the PMP, and shows that the greedy heuristic can in fact perform arbitrarily poorly (i.e., no minimum performance guarantee can be given for the greedy heuristic). The obvious conclusion arising from these observations is that the PMP should be solved using an exact LP solver *only*.

The PMP does not model routing constraints. Key consequences of this omission are discussed, namely the opportunity to optimize the product-workstation routing strategy is foregone, and the possibility of unexpected massive work-in-progress accumulations is raised (drawing on results from the theory of queueing network stability). An alternative mix problem formulation (the rate mix problem (RMP), which has computational requirements similar to that of the PMP) is described that explicitly incorporates routing information. Solution strategies for optimisation of both the routings and mix using the RMP are described, and guidelines are provided for the preliminary identification of possible pathological work-in-process effects in the mix and routing solutions.

In summary, the objectives of this paper are as follows: to revisit and generically and fundamentally *explain* Plenert's claim that greedy heuristics are a poor choice of solver for the PMP; to point to a key drawback in the PMP formulation and the consequences thereof; to propose an alternative mix problem formulation that allows optimisation of both the mix and routing strategy, and to provide simple practical guidelines for preliminary identification of pathological work-in-process accumulations.

The organization of this paper is as follows: the performance of greedy heuristics for mix problems is analysed in §2. The consequences of omitting routing information in mix problem formulations is presented in §3. An alternative mix problem formulation and associated solution strategies appears in §4. Guidelines to check system stability are provided in §5. Conclusions are rendered in §6.

2. PMP SOLUTIONS

2.1 GREEDY HEURISTICS

Pure greedy heuristics *build* a solution to a problem over several iterations, assigning the numeric value of exactly one variable in each successive iteration, by optimising a *surrogate* objective measure (that is, a measure other than that contained in the objective function). The iterative process continues until all variables' values have been fixed (possibly set equal to zero). There is no mechanism in pure greedy heuristics for "undoing" the variable value assignments of preceding iterations.

Specifically, the pure greedy heuristic (minor variations are possible, but do not materially impact the structure of the solution) for an n-product and m-workstation PMP is:

Pure Greedy Heuristic for the PMP

Step

0: Initialize, letting $c_j = C_j$ for $j = 1,\ldots,m$ denote the remaining workstation capacities; let $d_i = D_i$ for $i = 1,\ldots,n$ denote the unsatisfied market demands; and let $x_i = 0$ for $i = 1,\ldots,n$ denote the production levels of each product.

1: Identify the critically-constrained (bottleneck) workstation. The bottleneck is workstation j^* where:

$$j^* = \arg\max_j \left\{ \sum_{i=1}^{n} d_i t_{ij} - c_j \right\}$$

2a: Exploit the bottleneck workstation j^* by maximizing production of product i^* where:

$$i^* = \arg\max_{i|t_{ij^*}\neq 0} \left\{ \frac{r_i}{t_{ij^*}} \right\}$$

2b: Set the production level of product i^* at:

$$x_{i^*} = \min\left\{ d_{i^*}, \min_j \left\{ \frac{c_j}{t_{i^*j}} \right\} \right\}$$

3: Update the remaining workstation capacities and unsatisfied market demands, letting:

$$c_j = c_j - \sum_{i=1}^{n} t_{i^*j} x_{i^*} \quad \forall j = 1,\ldots,m$$

and $d_{i^*} = d_{i^*} - x_{i^*}$. If x_{i^*} changed in the current iteration, go to step 1, else stop.

Broadly speaking, the pure greedy heuristic (see Plenert [23]) identifies a solution to the PMP by identifying a bottleneck workstation, and then maximizes the throughput of the product generating the best profit per unit of bottleneck workstation capacity. In step 1 of the greedy heuristic we consider just one of several ways of identifying bottleneck workstations (there the bottleneck is the defined as the most overloaded workstation). Alternative "static" definitions are possible. Bottlenecks may shift (Lawrence and Buss [14]) dynamically over time if the "bottleneck" workstation at any given instant in time is defined as the most congested workstation. The pure greedy heuristic for the PMP does not account for dynamic bottlenecks.

Luebbe and Finch [17] compare the performance of the pure greedy heuristic to optimal solutions of an instance of the PMP first published by Goldratt. Luebbe and Finch's erroneous conclusion, based upon the comparison of the greedy heuristic performance with the optimal solution of just one instance of the PMP, is essentially that the greedy heuristic is *not* out-performed by LP solvers. Other authors have arrived at the same incorrect conclusion (e.g., Geyser [8]). Plenert [24] concludes that the greedy heuristic and exact LP solvers will perform identically, based upon two instances of the PMP (Plenert compared the performance of linear program solvers, the greedy heuristic, and a traditional accounting algorithm). Lee and Plenert [15] appear to have provided the first instance of the PMP where an LP solution out-performs the greedy heuristic. Dissenting commentary on Lee and Plenert [15] was published by Maday [18], and Posnack [25]. Plenert [23] later published other instances of the PMP where the greedy heuristic is out-performed by linear (or integer) program solvers. Fredenhall and Lea [6] afterward proposed a revised greedy heuristic for the PMP based upon exhaustive search in the solution neighbourhood of the greedy heuristic solution, seemingly suggesting that heuristics are still worth a look, despite the fact that Plenert and Lee's examples provide concrete proof that the greedy heuristic returns sub-optimal solutions on some instances of the PMP (and despite the fact that an exact polynomial-time solver is available). While the superiority of LP solvers over the greedy heuristic is clear to some authors (e.g., Miltenburg [20]), it is worth revisiting the issue in more depth to understand *why* the greedy heuristic may fail and to quantify its performance bounds, if any. In the next subsection this paper explains the sub-optimality (and potentially disastrous solution quality) of greedy heuristics, concluding that that the PMP should be solved using an exact (LP) solver *only*. The paper later addresses modelling issues, and suggests that even the PMP formulation should be overlooked in favour of an alternative rate mix formulation that explicitly encodes routing information.

2.2 PERFORMANCE OF THE PURE GREEDY HEURISTIC

To recapitulate the importance of Lee and Plenert's findings, consider the following 2-product, 2-workstation instance of the PMP:

PMP Instance:

$$\max 2x_1 + x_2 \tag{5}$$

subject to:

$$3x_1 + x_2 \leq 75 \tag{6}$$

$$x_1 + 2x_2 \leq 100 \tag{7}$$

$$x_1 \leq 50 \tag{8}$$

$$x_2 \leq 100 \tag{9}$$

$$x_1, x_2 \in \aleph_+ \tag{10}$$

Here (6) and (7) are capacity constraints corresponding to the constraint set of equation (2), while (8) and (9) are demand constraints corresponding to the constraint set of equation (3). Constraint (10) corresponds to the constraints in equation (4) in the prototype PMP formulation.

The optimal solution for this instance is $(x_1, x_2) = (10, 45)$, with a revenue total of 65. Note that the optimal solution requires 100% utilisation of both workstations. Now, consider the solutions generated by the greedy procedure: workstation 1 is the bottleneck (step 1), and the marginal returns per unit of bottleneck capacity (step 2a) are 2/3 and 1 for product 1 and 2 respectively. The heuristic indicates that we should produce 50 units of product 2 (step 2b), and eventually terminates with the sub-optimal solution of $(x_1, x_2) = (0, 50)$ that generates a revenue total of only 50 (since $x_2 = 50$ forces $x_1 = 0$ in equation (7)). Did we identify the correct bottleneck originally? To check, suppose that workstation 2 is the bottleneck. The marginal returns per unit of bottleneck capacity are 2 and 0.5 for product 1 and 2 respectively, and the production of $x_1 = 25$ is indicated. The heuristic eventually terminates with the sub-optimal solution $(x_1, x_2) = (25, 0)$ which again produces a total revenue of only 50 (since $x_1 = 25$ forces $x_2 = 0$ in equation (6)). Neither choice of workstation as bottleneck produces a solution that is even within 80% of the optimal revenue level for this instance.

Artificial instances of the PMP can be constructed to make the greedy heuristic look arbitrarily bad. That is, we are unable to find a lower bound on the performance of the greedy heuristic. Consider the following n-product and n-workstation instance of the PMP (note that $m = n$): let $r_i = 1$ for $i = 1,\ldots, n$; $C_j = 100$ for $j = 1,\ldots,n$; $D_i = 100$ for $i=1,\ldots,n$ and let

$$t_{ij} = \begin{cases} \varepsilon & \text{if } i \neq j \\ 1 & \text{otherwise} \end{cases}$$

Writing this out we get:

PMP Instance: $\max x_1 + x_2 + \ldots + x_n$

subject to:

$$x_1 + \varepsilon x_2 + \varepsilon x_3 + \cdots + \varepsilon x_{n-1} + \varepsilon x_n \leq 100$$

$$\varepsilon x_1 + x_2 + \varepsilon x_3 + \ldots + \varepsilon x_{n-1} + \varepsilon x_n \leq 100$$

$$\vdots$$

$$\varepsilon x_1 + \varepsilon x_2 + \varepsilon x_3 + \ldots + \varepsilon x_{n-1} + x_n \leq 100$$

$$x_i \leq 100 \quad i = 1,\ldots,n$$

$$x_i \in \aleph_+ \quad i = 1,\ldots,n$$

Let ε be a small constant strictly less than one, (for example, choose $\varepsilon = (n\text{-}1)^{-1}$ and note that we assumed earlier that $n \geq 2$ to avoid trivialities). All machines have the same initial load for the purposes of bottleneck identification. Arbitrarily suppose that machine k is "the" bottleneck. Any product except for product k now has a return per bottleneck hour of ε^{-1} (which is greater than or equal to one, and the return per bottleneck hour for product k is exactly one). The greedy heuristic will select any product $l \neq k$ as the most profitable product (per bottleneck hour), and assign $x_l = 100$ to satisfy the demand constraint on product l and all workstation capacity constraints. The heuristic eventually terminates with the solution of $x_i = 0\ \forall i \neq l$ and $x_l = 100$, producing a total revenue of 100 (note that the revenue is independent of n). It is easy to confirm that the solution $(x_1,\ldots,x_n) = (50,\ldots,50)$ is a feasible solution for this instance, and that the total revenue from this solution is $50n$ (which is greater than 100 if $n>2$, and this is not even the optimal solution). As $n \rightarrow \infty$ the greedy heuristic produces arbitrarily poor solutions for this instance relative to the optimal solution (since the heuristic produces a fixed revenue, while the optimal revenue grows as a function of n). This example proves that no minimum performance guarantees can be found for the greedy heuristic.

Greedy heuristics are principal-direction (axis) edge-following algorithms, and the greedy heuristic for the PMP is no exception. The variable assignment trajectory of the PMP greedy heuristic (which commences at the origin, with all variables initially unassigned) is graphically depicted in Figure 1 using thick black lines for two sample *relaxed* PMP polytopes. The relaxed PMP polytope is located in the positive orthant, and is generally comprised of a hypercube induced by constraint set (3) cut by facets induced by constraint set (2).

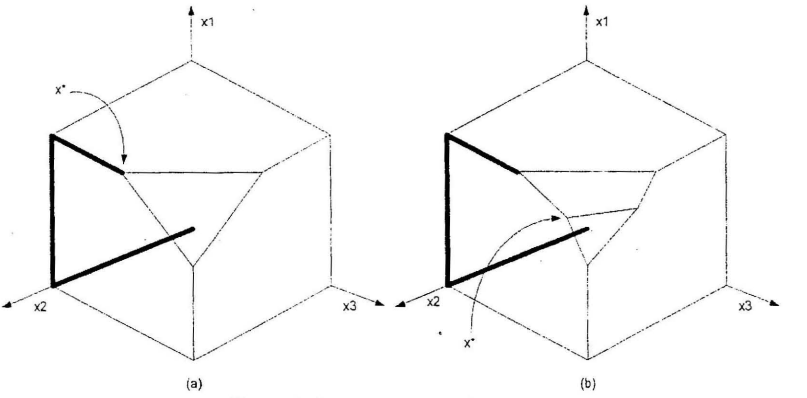

Figure 1: Sample PMP polytopes

The limitations of the greedy heuristic are immediately evident in Figure 1: the greedy heuristic can never reach a solution via an extremal edge that is not a principal direction. In

polytope (b) the optimal solution (marked x^*) is thus unreachable by a greedy heuristic. It is precisely the failure to accommodate the possibility of optimisation of several variables *simultaneously* that accounts for the sub-optimality of the greedy heuristic on polytope (b). That is, when multiple workstation bottlenecks exist, the greedy heuristic may return sub-optimal solutions.

The behaviour of the greedy heuristic can be partially characterized: let

$$F_D = \left\{ x \in \Re_+^n \mid 0 \le x_i \le D_i; \quad i = 1,\ldots,n \right\}.$$

(F_D denotes the hypercube induced by constraint set (3)), and let:

$$F_{PMP} = \left\{ \quad x \in \Re_+^n \mid \sum_{i=1}^n t_{ij} x_i \le C_j, 0 \le x_i \le D_i; \quad i = 1,\ldots,n; j = 1,\ldots,m \right\}.$$

(F_{PMP} denotes the relaxed PMP polytope). Let $cl(x)$ denote the closure of set x, and $A \backslash B = \{x \in A; x \notin B\}$ where A and B are sets.

Claim 1: if $cl(F_D \setminus F_{PMP})$ is convex then the greedy heuristic will return the optimal PMP solution.

The proof of the claim appears in the appendix. The claim is highly instructive since it implies that at least two workstations must be utilised in the optimal PMP solution (to ensure that $cl(F_D \setminus F_{PMP})$ is non-convex) before the greedy heuristic may return a sub-optimal solution. Moreover, these workstations must be "connected" by product routings in the sense that at least two different products are produced on each of these workstations in the optimal mix, i.e., the optimal mix must involve at least two bottleneck workstations before the greedy heuristic may fail. Hopefully, this situation has not been frequently encountered in practice! Unfortunately, there is at least one industrial plant with multiple bottlenecks (where the greedy heuristic under-performed), and it seems plausible that there may be many more (Bischoff [2]).

Although the greedy heuristic may perform well on many PMP instances, there can be no justification for its use in view of the proliferation of software in office suite software for optimally solving LPs, and the potentially disastrous financial consequences of *not* finding the optimal production mix.

3. MIX PROBLEM FORMULATIONS: ISSUES

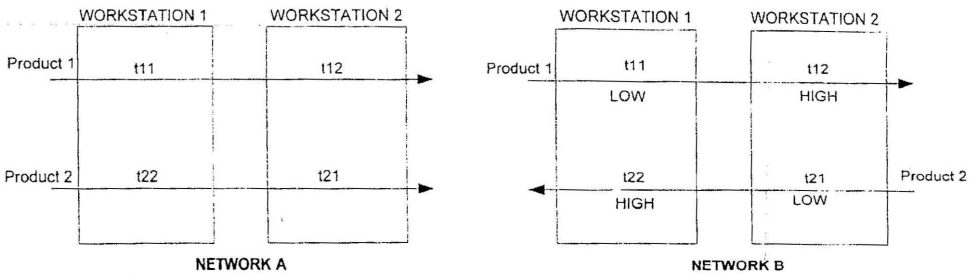

Figure 2: Two routing configurations arising from a single PMP instance

The PMP does not incorporate product routing constraints or strategies. This omission has at least two potentially crucial consequences:

1. the PMP cannot be used to optimise with respect to routing strategy when routing alternatives exist, and

2. the implied routing must be inferred from the PMP solution and technological considerations. It is not hard to see that multiple routing configurations can potentially be inferred from a single PMP instance. For example, the two networks in Figure 2 (which differ only in the order of workstations visited by product 2) have the same PMP formulation. Unfortunately, this is not a trivial observation because an unstable (this is explained below) routing strategy may be extrapolated from the PMP solution.

A queueing network is said to be unstable if the network does not possess a stationary queue length distribution. In practice an unstable queueing network will experience highly undesirable and wildly oscillating queue length sizes that eventually drift off to infinity. Informally, this phenomenon would be characterized in a production setting by pathological and increasingly massive fluctuations (oscillations) in work-in-process inventory at two or more workstations, and by a steady overall increase in the work in process (toward infinity) as time increases *if the facility were left unmanaged.* In practice there is considerable capacity management intervention in any production facility that averts such disasters. Nevertheless, it is natural to prefer a stable mix and routing strategy design over an unstable design as a stable design will reduce the need for, and frequency of management interventions, and may potentially result in a higher maximal achievable facility throughput.

Until recently it was thought that a queueing network would be stable for any work-conserving (workstations may not be idle when work is awaiting service there) service discipline if the utilisation of every station is less than 100% (if the required utilisation exceeds 100% the ensuing inventory explosion is obvious). Unfortunately, research on the stability of queueing networks has shown that networks exist that are unstable even though the utilisation of every station is strictly less than 100%, for certain scheduling policies.

Suppose that $t_{11} = t_{22} = 0$ and $t_{12} = t_{21} = 0.5$ in both networks in Figure 2. It turns out (see Dai [5]) that Network A in Figure 2 is stable under any work-conserving scheduling policy, whereas Network B is unstable for several scheduling policies. For example, the priority policy (with product type 2 having higher priority than product type 1 at workstation 1, and product 1 having higher priority at workstation 2 in Network B, as indicated in the figure) results in an unstable network. In fact, Network A has a stable achievable capacity of 100% at both workstations, while no more than 50% utilisation can be achieved at either station in Network B (because high priority service effectively blocks low priority work from entering the system)! Service policies play a huge role in determining whether a queueing network will be stable or otherwise, and it is interesting to note that Bramson [3] and others have shown that even the extremely common first-in-first out (FIFO) scheduling policy when implemented at all machines, does not guarantee network stability in all networks!

Unfortunately, no general technique is known at the present time that will identify unstable mix problem solutions. For the practitioner the easiest way to eliminate the possibility of an unstable system is to generate a product mix and routing strategy solution, and then check if it is stable by:

a) checking to see if (or forcing) the routing structure meets certain structural requirements that guarantee stability. Guidelines for performing this check are set out in §5; or

b) employing carefully screened (using simulation) work sequencing and allocation policies at every workstation that guarantee stability irrespective of the routing structure. This possibility is delicate and many companies prefer to rather simplify the routing structure to guarantee stability (Louis [16]).

Fortunately, the deficiencies of the PMP formulation in respect of routing modelling can be substantially remedied. The next section describes an alternative mix problem formulation based upon an infinite-length planning period and production rates (rather than volumes) that incorporates explicit probabilistic routing constraints. Solution strategies for obtaining both the optimal mix and routing strategy solutions using this formulation are discussed in §4.

4. THE RATE MIX PROBLEM (RMP)

4.1 FORMULATION

A finished product in general requires a series of processing operations (and possibly assembly or disassembly operations also); assign each such operation in the routing of a product a unique class identifier so that each class is served at exactly one workstation. We now view the environment as a network of m workstations that together process M classes of "customers" (intermediate or end products). Clearly, there are in general more classes in the network than products in the corresponding PMP formulation (i.e, $M \geq n$). Write σ_j for the set of classes served at workstation j.

Assume for now that routings are *known* and probabilistic: every time a class-i customer completes service it is routed for service as a class-j customer with probability p_{ij}. This is a simplified routing model that incorporates the widely-used special case of deterministic

routing (where $p_{ij} \in \{0,1\}$, $\forall i,j$), but that cannot approximate state-dependent (e.g., inventory-dependent) routing strategies. Let $\boldsymbol{P} = (p_{ij})$ denote the $M \times M$ routing matrix. The probability that a customer leaves the network after class-i service is $1 - \sum_{j=1}^{M} p_{ij}$. Assume $\boldsymbol{P}$ to be transient. This means that $\boldsymbol{I} - \boldsymbol{P}$ ($\boldsymbol{I}$ is the identity matrix) is invertible, or in other words, all products entering the production facility require processing on at most a finite number of workstations before leaving the network with probability one. This assumption implies that the network is *open*. Let $\boldsymbol{b} = (b_i)$ denote the M-dimensional vector of unit class processing times with b_i denoting the unit processing time of class i (to relate this to our previous notation, note that $b_i = t_{ij}$ where $i \in \sigma_j$).

The PMP is a finite-horizon problem (that is, the Ds and Cs in the PMP formulation are specified for a fixed-length planning period). A rate-based approach facilitates the formulation of an associated infinite-length period problem that we call the rate mix problem (RMP), which is defined as follows:

RMP:
$$\max \sum_i r_i' \lambda_i \tag{11}$$

subject to:
$$\lambda_i = \alpha_i + \sum_j \lambda_j p_{ji} \quad \forall i \tag{12}$$

$$\sum_{i \in \sigma_j} b_i \lambda_i < \tilde{\rho}_j \quad \forall j \tag{13}$$

$$0 \le \alpha_i \quad \forall i \tag{14}$$

$$0 \le \lambda_i \le \lambda_i^D \quad \forall i \tag{15}$$

where r_i' is the marginal unit revenue for class-i production (usually, $r_i' = 0$ for any class i that is not an end product), α_i (a variable in the problem) is the exogenous arrival rate of class-i customers to the network (i.e., customers that are completely new to the system), and λ_i (a variable) is the effective arrival rate of class-i customers at the workstation that serves class-i customers. The effective arrival stream of any class is the superposition of exogenous and endogenous (routed) streams of customers from other workstations. The load $\tilde{\rho}_j^C$ is the maximal permissible utilisation (expressed as a fraction) of station j, and λ_i^D is the maximal market demand rate for class-i production. Let
$\alpha = (\alpha_1, \ldots, \alpha_m)$ and let $\lambda = (\lambda_1, \ldots, \lambda_m)$. The assumptions are:

$$0 \le r_i' < \infty \quad \forall i, 0 < \tilde{\rho}_j^C \le 1 \quad \forall j, 0 \le b_i < \infty \quad \forall i, \sum_{i \in \sigma_j} b_i > 0 \quad \forall j \text{ and } 0 \le \lambda_i^D \quad \forall i.$$

The first constraint set (12) specifies the dependence of the effective arrival rate of class-i customers on the rate of exogenous and probabilistically-routed endogenous customer streams. The second constraint set (13) requires that the load on each workstation be strictly less than the maximal workstation load capacity (which must be less than or equal to unity by assumption). The third constraint set (14) requires that all exogenous arrival rates are non-

negative. Constraint set (15) requires that the departure rates of a class be less than the rate that the market (demand) can absorb. The objective function (11) calls for the maximization of the total marginal rate of return by optimising with respect to the variables, $\alpha_i \ \forall i$ and $\lambda_i \ \forall i$.

4.2 RMP SOLUTION STRATEGIES

The RMP formulation (with pre-specified probabalistic routings) is an LP and the optimal mix can be obtained in polynomial time (in practice, even extremely large RMPs will be optimally solved in a negligible amount of time on a computer using an LP solver, *never* a heuristic).

One of the most attractive features of the RMP formulation is that it may be adapted to explore and optimise the maximal achievable throughput in a production facility, and the sensitivity thereof to routings. While routing options may be limited in many production settings, they play a key role in determining the achievable throughput capacity in a variety of industries with multiple workstations of the same type. An example of such an environment is a brewery, where there are typically a number of fermenting vessels (of possibly dissimilar size) that can produce a variety of beer products, and a number of storage tanks (of possibly dissimilar size) in which beer is stored after fermentation, and prior to bottling. The routing (and the timing thereof) of beer from a fermentation vessel to (possibly multiple) storage tanks profoundly impacts the achievable throughput at some breweries (Meyer [19]).

Consider expanding the RMP which is obtained by adding the constraint set $0 \leq p_{ij} \leq 1, \forall i, j$ to the RMP and treating the routing probabilities (p_{ij}s) and mix variables (α_is and λ_is) as optimisation variables. The expanded RMP is a non-linear program (since constraint set (12) is now non-linear) that apparently cannot be efficiently solved in large dimensions. One workaround for this problem is to enumerate some, or possibly all feasible routings, and treat each product routing alternative as a unique phantom product in the RMP. Minor modifications to the RMP formulation are required to ensure that the phantom products associated with a real physical product do not exceed the demand rate constraints (constraint set (15)). This approach requires that we again need only solve an LP. Although the workaround is exponentially explosive in the number of workstations and products, industrial settings usually offer only limited scope for flexible routings (due to plant layout or technological considerations of the equipment). Alternatively, it is easy to devise an iterating scheme that alternately solves for the mix (while fixing the routings) and for properties of the routings (e.g., identification of highway routings, or dispersion routings) while fixing the mix.

Finite planning periods are largely a result of the convenience thereof for accounting practices and computerized planning software systems that have been used over the last several decades. In general there is usually no compelling *manufacturing* reason to insist on using finite production planning periods. If an RMP solution *must be* converted to a finite planning period volume, then the period volume can normally be found by simply multiplying the optimal product rates by the length of the planning period. Unfortunately, this approach is confounded by requirements for specified large batch sizes (such as those found in breweries, where whole tanks/vessels of beer product must be routed). Meyer [19] has addressed this problem in a case study, scheduling the finite-period production volumes (subject to a variety of additional practical constraints such as due dates, sequence-dependent setups and batching considerations) using a shifting bottleneck (SB) procedure (see, for example, Adams, Balas

and Zawack [1], Pinedo and Singer [22], and Uszoy and Wang [26]) to minimize the full-schedule makespan. Meyer found, in limited experimentation, that the solutions obtained in this manner produce required makespans that exceeded the actual planning period by only 7% on average. Given that the finite-period schedules are not optimal (SB solution techniques heuristically decompose the production environment into a collection of one-machine scheduling sub-problems, and the order in which the sub-problems are solved impacts the final schedule), Meyer's techniques seem promising.

5. QUEUEING NETWORK STABILITY GUIDELINES

Several quick checks can be effected to evaluate whether a PMP or RMP solution is potentially unstable: if the routing is of the feed-forward type (see Dai [5]) (that is, if the routing graph, when sketched in similar fashion to Figure 2, with workstations represented by vertices and routing requirements by directed edges in the graph, contains no dicycles), then the solution of the PMP is attainable in the sense that the network will be stable. This result was proven by Dai [5] among others. A feed-forward routing structure permits no rework loops, and work may never reenter a production line from a downstream workstation.

Unfortunately, non-feed-forward routings are unavoidable in some situations due to extremely high machine costs and processing requirements that require that some products revisit a particular machine as many as 30 times (this is typical in semi-conductor wafer fabrication). In such cases service policies can be specified that guarantee stability: if the unit class processing times are identical at a station, and if this is true at all stations (the processing times may differ between, but not within a workstation) then the network will be stable if the FIFO service discipline is employed at all workstations (Bramson [3]).

The application of universally stable (Warren and O'Cinneide [27]) scheduling and sequencing policies (such as the head-of-line-processor-sharing (HOL-PS) policy (Bramson [4])) at every station will ensure that the optimal solution of the PMP is attainable and that a stable network will result, irrespective of the routing topology.

Unfortunately, priority policies, which are commonly used in practice, are particularly fragile from a stability perspective, and extreme care must be exercised when priority, polling or FIFO policies are to be used in networks with non-feed-forward routings. While it is possible to stably implement any feasible PMP or RMP mix and routing solution through the careful choice of scheduling and sequencing policies, the matter is delicate, and simulations are recommended in practical settings. The blunt approach of using a feed-forward routing strategy (circumventing the need for analysis) is preferred and most widely used in practice, particularly in assembly plants, even though it comes with a potential associated reduction in maximal achievable throughput capacity.

6. CONCLUSIONS

The use of greedy heuristics for solving the PMP is fundamentally flawed. Given that significant revenue may be foregone by using the greedy heuristic, the use of an exact solver is unavoidable.

The PMP formulation fails to account for routing. The routing strategy has to be inferred from PMP solutions, and the PMP formulation does not permit the optimisation of routing strategies.

The RMP is an alternative to the PMP formulation that explicitly models assembly, disassembly, and probabilistic (including deterministic) routing in a rate-based formulation that avoids possible bin-packing (Nemhauser and Wolsey 1998) effects that must be dealt with at planning boundaries in revolving implementations of PMP solutions. As in the case of the PMP, though, the RMP ignores a variety of practical aspects such as sequence-dependent setups, due dates, and batching. The incorporation of many such practical issues may force a formulation that would be hard or impossible to solve to optimality in large dimensions.

When optimising for the mix *only*, the RMP is an LP with computational complexity similar to that of the PMP, and solution using any (exact) LP solver is straightforward and can be expected to consume a negligible amount of computer (PC) time, even for the largest practical production mix problems. As in the case of the PMP, RMP solutions must also be checked to ensure that they do not produce an unstable production environment.

Optimisation strategies are discussed that permit optimisation of both the mix and the routing strategy. Deeper investigation of this possibility is merited.

ACKNOWLEDGEMENTS

The author acknowledges interesting discussions with Pieter Pretorius and thanks one anonymous reviewer for valuable comments.

REFERENCES

[1] Adams, J, Balas, E, and Zawack, D, 1998. "The shifting bottleneck procedure for job shop scheduling," *Management Science*, Vol. 34, No. 3, pages 391-401.

[2] Bischoff, K, 1997. Personal communication, Telcast, Kempton Park, South Africa.

[3] Bramson, M, 1996. "Convergence to Equilibria for Fluid Models of FIFO Queueing Networks," *Queueing Systems: Theory and Applications*, 22, pages 5-45.

[4] Bramson, M., 1996. "Convergence to Equilibria for Fluid Models of Head-of-Line Proportional Processor Sharing Queueing Networks," *Queueing Systems: Theory and Applications*, 23, pages1-26.

[5] Dai, J G, 1995. "On the Positive Harris Recurrence for Multiclass Queueing Networks: A Unified Approach Via Fluid Limit Models," *Annals of Applied Probability*, Vol. 5, pages 49-77.

[6] Fredenhall, L D and Lea, B R, 1997. "Improving the Product Mix Heuristic in the Theory of Constraints," *International Journal of Production Research*, Vol. 35, No. 6, pages 1535-1544.

[7] Garey, M R and Johnson, D S, 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman, San Francisco.

[8] Geyser, G, June-July 1991. "The Theory of Constraints," *Productivity SA*, page 17-22.

[9] Goldratt, E M, 1992. *The Goal*, Croton-on-Hudson, New York, North River Press, 2nd edition.

[10] Goldratt, E M and Fox, R, 1986. *The Race*, Croton-on-Hudson, New York, North River Press.

[11] Goldratt, E M, 1990. *The Haystack Syndrome*, Croton-on-Hudson, New York, North River Press.

[12] Kasilingam, R G, 1995. "Product Mix Determination in the Presence of Alternate Process Plans and Stochastic Demand," *Computers and Industrial Engineering*, Vol. 29, No.1-4, pages 249-253.

[13] Khouja, M, Mehrez, A and Rabinowitz, G, 1994. "A Nonlinear Model for Capacity Allocation and Throughput Determination in Cellular Manufacturing Systems," *Engineering Optimization*, Vol. 23, pages 125-139.

[14] Lawrence, S R and Buss, A H, Winter 1994. "Shifting Production Bottlenecks: Causes, Cures and Conundrums," *Production and Operations Management*, Vol. 3, No. 1.

[15] Lee, T and Plenert, G, 1993. Optimizing theory of constraints when new product alternatives exist," *Production and Inventory Management Journal*, Vol. 34, No. 3, pages 51-57.

[16] Louis, M, 1994. Personal communication, Hewlett Packard, Boise, Idaho, USA.

[17] Luebbe, R and Finch, B, 1992. "Theory of Constraints and linear programming: a comparison," *International Journal of Production Research*, Vol. 30, No. 6, pages 1471-1478.

[18] Maday, C J, 1994. Proper use of Constraint Management," *Production and Inventory Management Journal*, Vol. 25, No. 1, page 84.

[19] Meyer, W, 1998. "Developing a generic scheduling paradigm for the brewing environment," undergraduate thesis, Department of Industrial and Systems Engineering, University of Pretoria.

[20] Miltenburg, J, 1997. "Comparing JIT, MRP, and TOC, and embedding TOC into MRP," *International Journal of Production Research*, Vol. 35, No. 4, pages 1147-1169.

[21] Nemhauser, G L and Wolsey, L A, 1998. *Integer and Combinatorial Optimization*, Wiley.

[22] Pinedo, M and Singer M, 1999. "A shifting bottleneck heuristic for minimizing the total weighted tardiness in a job shop," *Naval Research Logistics*, Vol. 46, No. 1, pages 1-17.

[23] Plenert, G, 1993. "Optimizing theory of constraints when multiple constrained resources exist," *European Journal of Operations Research*, 70, pages 125-133.

[24] Plenert, G, 1996. "Maximizing product mix profitability -- what's the best analysis tool," *Production Planning and Control*, Vol. 7, No. 6, pages 547-553.

[25] Posnack, A J, 1994. "Theory of Constraints: Improper Applications Yield Improper Conclusions," *Production and Inventory Management Journal*, Vol. 35, No. 1, pages 85-86.

[26] Uzsoy, R and Wang C-S, 2000. "Performance of decomposition procedures for job shop scheduling problems with bottleneck machines," *IJPR*, Vol. 38, No. 6, pages 1271-86.

[27] Warren, G M H and O'Cinneide, C A, 14-17 July 1997. "Universally Stable Processor Sharing Fluid Policies," *EURO XV/INFORMS XXIV conference*, Barcelona, Spain.

APPENDIX

Proof of Claim 1: F_D is convex since it is by definition and assumption a non-empty hypercube in the positive orthant. F_{PMP} is convex since it is the non-empty set induced by the constraints of an LP, and the feasible region of any LP is convex [21]. Note that $cl(F_D \backslash F_{PMP})$ is the closure of the set that is obtained by removing F_{PMP} from F_D. If:

a) $cl(F_D \backslash F_{PMP})$ is convex and non-empty (for example, see Figure 1(a)): then the intersection of $cl(F_D \backslash F_{PMP})$ and F_{PMP} is a simple planar facet since F_{PMP} is a subset of F_D, and F_{PMP} is convex. Hence the PMP will have at least one optimal extremal point (a whole polytope edge or even a whole plane of polytope points may be optimal) that is an extremal edge point of F_D, and the greedy heuristic will succeed in locating this point by traversing the extremal edges of F_D which all lie on principal (axis) directions.

b) $cl(F_D \backslash F_{PMP})$ is empty then $F_{PMP} = F_D$ and the greedy heuristic will succeed in finding the optimal extremal edge points(s) of the PMP by following the principal-direction oriented extremal edges of the F_D hypercube.