Unified Concept of Bottleneck

Saral Mukherjee and A. K. Chatterjee

W.P. No. 2006-05-01
May 2006

INDIAN INSTITUTE OF MANAGEMENT
AHMEDABAD – 380015
INDIA

Unified Concept of Bottleneck

Saral Mukherjee and A. K. Chatterjee

Abstract

The term 'bottleneck' has been extensively used in operations management literature. Management paradigms like the Theory of Constraints focus on the identification and exploitation of bottlenecks. Yet, we show that the term has not been rigorously defined. We provide a classification of bottleneck definitions available in literature and discuss several myths associated with the concept of bottleneck. The apparent diversity of definitions raises the question whether it is possible to have a single bottleneck definition which has as much applicability in high variety job shops as in mass production environments. The key to the formulation of an unified concept of bottleneck lies in relating the concept of bottleneck to the concept of shadow price of resources. We propose an universally applicable bottleneck definition based on the concept of average shadow price. We discuss the procedure for determination of bottleneck values for diverse production environments. The Law of Diminishing Returns is shown to be a sufficient but not necessary condition for the equivalence of the average and the marginal shadow price. The equivalence of these two prices is proved for several environments. Bottleneck identification is the first step in resource acquisition decisions faced by managers. The definition of bottleneck presented in the paper has the potential to not only reduce ambiguity regarding the meaning of the term but also open a new window to the formulation and analysis of a rich set of problems faced by managers.

Keywords: Bottleneck, Production, Scheduling, Theory of Constraints, Average shadow price

1 Introduction

The roots of bottleneck focused approach in operations management can be traced back to the days of Henry Ford. In his effort to deliver an affordable car, Ford introduced the moving assembly line which exploited the economies of scale involved in producing a standard product in high volume. It was understood that the workstation with the maximum processing requirement, denoted as the bottleneck, would constrain the output of the system. This understanding was inherent in the attempt to 'balance capacity' by ensuring that the total work was allocated equally among the workstations. The operating tool employed to achieve this allocation was the Assembly Line Balancing technique (Salveson (1955)). The focus on bottlenecks was implicitly captured by the importance given to the capacity utilisation metric as the prime tool for managerial planning and control in such high volume low variety environment.

The complexity of operations has increased tremendously since the days of Henry Ford. Single model assembly lines have given place to mixed model assembly lines. Inflexible

transfer lines have yielded ground to various forms of flexible manufacturing systems. The need for constant product innovation and the resultant product proliferation has resulted in an increase in the variety in the product mix. The identification of bottleneck becomes much more difficult as we move from the high volume low variety repetitive manufacturing scenario towards low volume high variety job shops and finally to the project environment. Job shops and projects primarily compete on the dimension of time as opposed to a mass production system where throughput at the lowest cost is the primary concern. Thus the definition of bottleneck has to be agreed upon given that 'capacity' is no longer approachable in job shops and projects in the same way as in assembly lines.

For a practicing manager running a production or service set up, focusing on the bottleneck is an intuitive way of managing this complexity. Yet, as we show in this paper, management science literature does not provide a bottleneck definition which is universally valid for all production/service scenarios. The absence of an universally applicable definition implies the absence of any universally applicable bottleneck focused approach. This void was partially filled when Goldratt and Cox (1984) proposed the Theory of Constraints (TOC). As an operations management paradigm, TOC centered mainly on the identification and exploitation of bottlenecks. For the practicing manager, the acceptance of such a theory comes naturally as it exploits the fundamental intuition about bottlenecks carried over from the days of Henry Ford. This acceptance is reflected in the phenomenal success of Goldratt's novel *The Goal*.

In order to demonstrate the universal applicability of TOC, Goldratt published the novel *Critical Chain* (Goldratt (1997)) aimed at applying the bottleneck focused approach in project environments. As Raz et al. (2003) explain, the *critical chain* is the set of tasks or activities that results in the longest path to project completion after resource leveling. Several authors have pointed out that the concept of *critical chain* is nothing but old wine in new bottle (McKay and Morton (1998), Trietsch (2005a)). Identifying the critical path after resource leveling as the bottleneck in a project is too simplistic and is dependent on the realised schedule (Herroelen et al. (2002)). The reader is referred to Herroelen and Leus (2001) for a balanced understanding of the merits and pitfalls of critical chain scheduling.

The criticism of Goldratt's bottleneck focused approach has centered around the lack of rigour in his work. The format of his primary works, the novel, does not allow a display of scientific rigour. This choice of format is not accidental and matches a disregard for the scientific process of acknowledging and building on the work of others. Previous research work was not acknowledged, topics like resource optimisation, sequence optimisation, investment optimisation were branded as 'academic' and by implication, irrelevant. Any academic work displaying similar characteristics would have been summarily consigned to oblivion by the research community. The very fact that researchers have been forced to devote time, energy and valuable journal pages to point out the lack of rigour, is due to the popularity of Goldratt's approach among practitioners. We would like to make a distinction between the bottleneck focused approach *per se* and Goldratt's implementation of the bottleneck focused approach. A practitioner should not be forced to compromise rigour in order to follow a bottleneck focused approach. Management science researchers need to provide a scientifically rigorous bottleneck focused approach as a viable alternative to Goldratt's approach. The work of Trietsch (2005b) is a step in this direction. Our effort in this paper

is on similar lines and is grounded in the understanding that the bedrock of a bottleneck focused approach is the definition of bottleneck itself.

In our view, Goldratt has not erred in his belief about the universal applicability of the bottleneck focused approach. However, the attempt at formulating an universally valid bottleneck focused approach was not preceded by the rigorous formulation of an universally valid bottleneck definition. Instead, a generic way to approach the problem of identifying bottlenecks is presented in Goldratt and Fox (1986): "A capacity constraint manifests itself in all of the major business issues. An analysis of the major business issues can be used to identify the capacity constrained resources". Trietsch (2005a), while quoting the above, pointed out that the definition was rather simplistic for a complex system, making it virtually impossible to apply.

The objective of this paper is to establish an universally valid bottleneck definition. The term 'universal' has several connotations for this paper. Primarily our intention is to search for a bottleneck definition which has applicability in diverse production/service scenarios spanning project environment to continuous flow lines. Secondly, we hope that an universally accepted bottleneck definition would bridge the gap between theoreticians and practitioners. We show that the popularly used capacity based and critical path based bottleneck definitions are special cases of the universally valid bottleneck definition. As a result, we sensitise the practicing managers against overextending the definition valid for a particular scenario. At the same time, the existence of an universal definition having scientific validity means that practitioners do not have to sacrifice rigour while employing a bottleneck focused approach to problem solving.

The foundation of this unified approach lies in understanding the similarity between the concepts of bottleneck and shadow price of resources. The shadow price for a resource in a mathematical programme provides the same managerial insights as does a bottleneck measure. The bottleneck definition proposed by us is an application of the concept of average shadow price in the context of operations management. We show how the definition can be applied in diverse production/service environments and discuss the procedure for determination of bottleneck values. The determination of the average shadow price is much more difficult than determining the marginal shadow price. The two prices would be equal if the Law of Diminishing Returns holds for a production/service environment. We provide an example to show that the Law of Diminishing Returns does not necessarily hold for all production/service environments. An interesting research question having significant implication for practitioners would then be to characterise environments where the two shadow prices are equivalent. We make a start by proving the equivalence of the average and marginal shadow prices for four specific environments.

The rest of the paper is presented as follows. In Section 2 we discuss the fundamental characteristics of bottlenecks and formulate a bottleneck definition which is in conformity with the idea of shadow price. In Section 3 we provide a classification of bottleneck definitions available in literature and show that many bottleneck definitions do not conform to the idea of bottleneck as a constraining resource. The literature on shadow prices for integer programmes is reviewed in Section 4. An universally valid bottleneck definition is introduced in Section 5 . Section 6 highlights the procedure for determination of bottleneck values. The relationship between the average and marginal shadow price is studied in Section 7. Finally we conclude in section 8.

2 Fundamental characteristics of bottlenecks

> **bot-tle-neck** (bot′$^{\partial}$l-nek′), *n.* **1.** a narrow entrance or passageway. **2.** a place or a stage in a process at which progress is impeded. *-v.t.* **3.** to hamper or confine by or as by a bottleneck. *-v.i.* **4.** to become like a bottleneck; be hindered by or as by a bottleneck. [1]

The term bottleneck has been extensively used in operations management literature. Yet there are few instances where it has been explicitly defined. However, practitioners have a clear understanding of the implications of a resource being a bottleneck. According to Goldratt, "An hour lost at a bottleneck is an hour lost for the entire system. An hour saved at a non-bottleneck is a mirage". In our opinion, this statement captures the essence of the concept of bottleneck. It communicates in very simple terms the crucial role played by bottlenecks. Moreover, the statement is 'actionable' in that it provides a clear focus for managerial planning and control activities.

We choose this statement as the guiding principle in formulating an universally valid bottleneck definition. This action on our part is not because we consider Goldratt's words as infallible. Instead, we realise that this statement is nothing but an application of the well-known complimentary slackness conditions (Tucker (1956)) in a production/service context. The complimentary slackness conditions for linear and non-linear mathematical programmes state that dual variables (shadow prices) exist if and only if the corresponding constraints are binding. In the context of a production/service system, this translates to the fact that an improvement for the whole ('entire system') is possible if and only if there is an improvement for a crucial part ('bottleneck') of the whole. We have not come across a single published article where this connection between Goldratt's characterisation of bottlenecks and the fundamental Operations Research concept of complimentary slackness conditions has been highlighted.

We propose the following bottleneck definition as our working definition till we provide a more rigorous definition later in the paper.

Definition 1 *A bottleneck constrains the performance of a system.*

Note that a resource defined as a bottleneck according to Definition 1 would be in accordance with our guiding principle. The definition is also in accordance with the complimentary slackness conditions for the mathematical programme representing the production/service situation at hand. If a resource constrains the performance of a system then a manager can profitably utilise an additional availability of that resource. Hence the resource is a bottleneck. On the other hand, the manager cannot profitably employ extra capacity for a resource which already has enough slack. Hence such a resource is a non-bottleneck. While Definition 1 is intuitive, its theoretical validity rests on the existence of a valid shadow price definition. Since most production/service environments are likely to be formulated as integer programmes, we need a valid shadow price definition for integer programmes as a fundamental construct on which we can claim the validity of Definition 1. Such a construct is provided by the concept of average shadow price, discussed later in this paper.

[1] The Random House Dictionary of the English Language, College Edition

While Definition 1 is stated in abstract terms, it can be readily made to conform to the needs of a particular application area. For example, consider the general job shop problem denoted as $J||C_{max}$ where a set of n jobs $J_1, J_2, \ldots, J_n$ require processing on a set of m machines. Each job J_i has a processing time p_i. The routing of each job, also known as the machine sequence of each job, is fixed. A machine can process only one job at a time. We need to determine the job sequence of each machine such that the makespan, the maximum completion time of all jobs, is minimised. Thus, for the $J||C_{max}$ environment, the 'performance' is measured by how quickly one can finish the processing of jobs in the system, i.e. makespan. Hence an application of Definition 1 for the $J||C_{max}$ environment would be as follows.

Definition 2 *For the $J||C_{max}$ environment, a machine is termed a bottleneck if it constrains the achievement of a lower makespan.*

The bottleneck machine constrains the performance of the system by not allowing more than one operation to be performed at a given time. While this characteristic is typical of all machines in a job shop, it is only for some machines that this characteristic comes in the way of achieving a lower makespan. We call such machines as bottleneck machines. Note that it is not necessary that a bottleneck has to be a resource. It can be any constraint which constrains the performance of the system. For example, as shown in Section 5.1, the duration of an activity on the critical path of a project network can be a bottleneck. We now provide a classification of the bottleneck definitions in operations management literature and show that many of the definitions do not conform to Definition 1.

3 A classification of bottleneck definitions

We classify the existing bottleneck definitions into five major groups. These are (i) Capacity based definitions (ii) Critical path based definitions (iii) Structure based definitions (iv) Algorithm based definitions and (v) System performance based definitions. Our focus is on production/service environments and we ignore references to bottlenecks in other Operations Research fields like the bottleneck assignment problem or the bottleneck traveling salesman problem.

3.1 Capacity based bottleneck definitions

A representative definition of bottleneck, occurring in many text books, can be stated as follows: "A bottleneck is defined as any resource whose capacity is less than the demand placed upon it" (Chase et al. (2006)). Similar definitions of bottleneck appear in TOC literature, e.g. according to the APICS Dictionary (The American Production and Inventory Controls Society (1995)), a bottleneck is a "facility, function, department, or resource whose capacity is less than the demand placed upon it. For example, a bottleneck machine or workcenter exists where jobs are processed at a slower rate than they are demanded". Such definitions propagate the following myth.

Table 1: Processing time data of a 2 job 2 machine flow shop

Job	M_1	M_2
J_1	1	3
J_2	4	3

Myth 1: Any resource whose capacity is less than the demand placed on it is necessarily a bottleneck.

Reality: A resource can be a non-bottleneck even if its capacity is less than the demand placed on it.

Example 1 Consider a service scenario where arriving customers are first served by Server A and then by Server B. Customers arrive at a steady rate of 12 customers per hour. Servers A and B can process at the rate of 6 and 4 customers per hour respectively. According to the bottleneck definitions expounded in Chase et al. (2006) and the APICS Dictionary, both servers are bottlenecks. However, Server A is not a bottleneck according to Definition 1 since the system output would not increase if its capacity is increased.

Lawrence and Buss (1994) proposed several capacity related bottleneck definitions based on the time horizon of analysis. In the short term, a machine is a bottleneck if it is temporarily under-capacitated. Similarly, in the long term, a machine is termed a bottleneck if it has the greatest long-run utilisation among all machines.

Myth 2: The resource having highest capacity utilisation is necessarily the bottleneck.

Reality: A resource can be a non-bottleneck even if its capacity utilisation is highest.

In Example 1, both servers have 100 per cent capacity utilisation. Yet, as we have shown already, Server A is not a bottleneck. A similar result also holds for bottleneck definitions based on workload of a machine, as in Uzsoy and Wang (2000) where test problems with bottleneck machines were created by manipulating the workload on machines. The workload of any machine is the sum of processing times of jobs on that machine.

Myth 3: The resource having highest workload is necessarily the bottleneck.

Reality: A resource can be a non-bottleneck even if its workload is highest.

Example 2 Consider minimising the makespan in a flow shop with processing times as provided in Table 1. The optimal solution using Johnson's Rule (Johnson (1954)) is $J_1 - J_2$ with makespan 8. Consider a two-stage flow shop with identical parallel machines in stage 2. A lower bound on makespan for such flow shops is the sum of processing times on stage 1 and the smallest processing time of stage 2. Since this lower bound equals 8, a decrease in makespan cannot be obtained by increasing machine availability at stage 2. Hence machine M_2 cannot be a bottleneck according to Definition 1 even though it has the highest workload.

In the intermediate term, Lawrence and Buss (1994) define a bottleneck machine as the one having most jobs or customers. Thus the machine with maximal queue length is the intermediate term bottleneck. Lawrence and Buss (1994) note that in their experience production managers describe production bottlenecks in terms of L, the number of jobs at a workcentre (in-process or in queue) or the time W required to complete processing of all waiting jobs. These two parameters are essentially same since they are related by Little's Law.

Myth 4: Any resource with a queue before it is necessarily a bottleneck.

Reality: A resource can be a non-bottleneck even if an infinite queue forms before it.

If the demand placed on a resource is more than its capacity, there would exist a queue before that resource. But the mere presence of a queue does not make a resource a bottleneck. This is demonstrated in Example 1 where Server A is not the bottleneck even though the queue length before it monotonically tends to infinity.

Myth 5: The resource having the biggest queue before it is necessarily the bottleneck.

Reality: A resource can be a non-bottleneck even if the queue before it is bigger than all other queues.

The maximal queue length based definition is indeed extensively used by practitioners as reported in Lawrence and Buss (1994). Server A in Example 1 will always have higher queue length than Server B. Yet, as already shown, Server A is not the bottleneck.

3.2 Critical path based bottleneck definitions

The critical path in a project restricts the achievement of a lower project completion time. Similarly, the critical path in a job shop defines the makespan of the schedule. It is well understood that any improvement of the overall objective of minimising project completion time or the makespan is conditional on shortening the critical path(s). The critical path considering resource availability constraints is termed the *critical chain* by Goldratt, who considered it as bottleneck for project environment. The concept of criticality can be extended from activities and paths to resources. A resource required by a critical activity is termed a critical resource. In the context of job shops, Adams et al. (1988) have pointed out that a partitioning of resources into critical and non-critical leaves a lot to be desired. Such a partitioning presents criticality as a 'yes or no property' rather than a matter of degree. Moreover, equating the concept of criticality with the concept of bottleneck is misleading.

Myth 6: A critical activity, path, chain or resource is necessarily a bottleneck.

Reality: A critical activity, path, chain or resource may not be a bottleneck.

Table 2: Project Data

Activity	Processing Time	Immediate Predecessor	Resource Required
A	2		R_A
B	1		R_B
C	1	B	R_C

Example 3 Consider the data for a three-activity project presented in Table 2. The optimal project completion time of 2 results from starting activities A, B and C at times 0, 0 and 1 respectively. All activities, paths, chains and resources are critical in the optimal schedule. Yet no activity in itself is a bottleneck. Let S and T be the dummy start and finish nodes. The critical paths (equivalently the *critical chains*) $S-A-T$ and $S-B-C-T$ constrain the project *together*. None of the critical paths is a bottleneck in itself. Similarly, all resources are critical but the project completion time cannot be decreased by increasing availability of any resource.

A bottleneck in such a situation is the 'package' constituting all the critical activities rather than any critical activity on its own. Similarly the 'package' of bottleneck paths can be defined. The difference between the concepts of criticality and bottleneck is further highlighted by discussing the following myth.

Myth 7: A bottleneck resource is necessarily a critical resource.

Reality: A resource can be a bottleneck even if it is non-critical.

Consider changing the processing time of J_1 on M_2 in Table 1 to 2. The optimal makespan remains 8 and the new optimal digraph does not contain disjunctive arcs belonging to M_2. Yet the optimal makespan can be reduced to 7 by increasing the availability of M_2 by one unit and setting sequence J_2-J_1 on M_1. Hence M_2 is a bottleneck even though it is non-critical. Comparing with Example 2, an interesting counterintuitive result emerges. A decrease in workload of machine M_2 has changed a non-bottleneck to a bottleneck!

3.3 Structure based bottleneck definitions

Several authors have defined a bottleneck based on the inherent structure of the production environment. For example, Drobouchevitch and Strusevich (2000) consider a job shop where the processing route for each job consists of two operations at most. The number of machines is arbitrary, and one of the operations of each job has to be processed on a particular machine, the same for all jobs. Such a machine is termed the bottleneck machine. In the next example we show that this definition of bottleneck does not conform to Definition 1.

Example 4 Consider the processing time data provided in Table 3 where 4 jobs have to be scheduled on 4 machines to minimise the makespan. Each job has two operations, the second one has to be done on machine B. Clearly, machine B is the bottleneck machine

Table 3: Processing times of 4 jobs on 4 machines

Job	M_1	M_2	M_3	B
J_1	5			2
J_2		7		3
J_3			100	2
J_4			200	3

Table 4: Processing time data of a 4 job 2 machine flow shop

Job	M_1	M_2
J_1	4	16
J_2	4	1
J_3	4	1
J_4	4	1

in accordance with the bottleneck definition advanced by Drobouchevitch and Strusevich (2000). But for any schedule, the addition of any number of machines of type B in parallel will not result in a decreased makespan. Hence B cannot be a bottleneck machine according to Definition 1.

Bottleneck definitions have also been proposed based on the structure of processing times. For example, while proposing branch and bound algorithms for the permutation flow shop problem, Carlier and Rebai (1996) have defined bottleneck as a machine on which jobs have higher processing times than on others. If all jobs have higher processing times on a particular machine then that machine is likely to be a bottleneck. However, the result is not so straightforward even if all but one job have higher processing time on a particular machine.

Example 5 Suppose 4 jobs have to be processed in a two-machine flow shop with minimisation of makespan objective. The processing time data for the problem is presented in Table 4. The optimum solution to this problem is $J_1 - J_x - J_y - J_z$ where $x, y, z \in \{2, 3, 4\}, x \neq y \neq z$. The optimal value of makespan is 23 and cannot be decreased by increasing the availability of M_1. Thus M_1 cannot be the bottleneck machine according to Definition 1 even though all but one job have highest processing time on M_1. By changing the processing time of J_1 on M_2 from 16 to 8 we can similarly show the fault with identifying the bottleneck as the machine on which the operation with maximum processing time is scheduled.

Grosfeld-Nir and Gerchak (2002) define a 'single bottleneck system' as a serial multistage production system where all stages, except one, have zero setup costs. The stage with non-zero setup cost is defined to be the bottleneck. Two-bottleneck and zero-bottleneck systems are similarly defined (Grosfeld-Nir (2005)). In the next example we show that the existence of setup time or setup cost on a machine does not automatically make it a bottleneck even if setup times and costs are zero for other machines.

Table 5: Processing time data of a 2 job 2 machine flow shop

Job	M_1	M_2
J_1	1	4
J_2	1	4

Example 6 Consider a 2 job 2 machine flow shop with processing time data presented in Table 5. The objective is to schedule the operations so as to minimise the makespan. Both jobs on machine M_1 have identical setup time of 1. Jobs do not require setup on machine M_2. It can be easily verified that machine M_1 is not a bottleneck according to Definition 1 even though it is the only machine with non-zero setup times and hence non-zero setup costs.

In certain cases, instead of defining a bottleneck, a 'non-bottleneck' machine has been defined. Strusevich and Hall (1997) consider the two-machine open shop scheduling problem in which one of the machines is non-bottleneck. A non-bottleneck machine is one such that an arbitrary number of jobs can undergo processing on that machine simultaneously. The concept is further clarified as "An alternative interpretation is to view the non-bottleneck machine as a collection of $m > n$ parallel identical machines, so that whenever it is desired to start the processing of some job, there is always a machine available. We denote this problem by $O2|NB|C_{max}$".

The condition specified for a machine to be non-bottleneck by Strusevich and Hall (1997) is a sufficient but not necessary condition. Consider a machine which has the characteristic that an arbitrary number of jobs can undergo processing on it at any point of time. Such a machine is definitely a non-bottleneck according to Definition 1. However, a machine which does not have this characteristic can still become a non-bottleneck machine. Machine B in Example 4 is an example of a non-bottleneck machine even though it does not allow an arbitrary number of jobs to be processed simultaneously.

3.4 Algorithm based bottleneck definitions

The most prominent algorithm based definition of bottleneck is contained in the Shifting Bottleneck (SB) heuristic of Adams et al. (1988). The SB procedure involves solving a series of machine-based decomposition problems with minimisation of maximum lateness (L_{max}) objective. At any iteration, the choice of next machine to be scheduled is based on the machine with the highest L_{max} value, termed the bottleneck machine.

Myth 8: The Shifting Bottleneck heuristic is a bottleneck focused approach.

Reality: The term 'bottleneck' in the Shifting Bottleneck heuristic is a misnomer.

The SB heuristic starts with the condition that none of the machines is scheduled. The makespan at this stage is the critical path assuming infinite resource availability. At this scheduling instant $t_{now} = 0$, let M_k be identified as the bottleneck by using the 'highest L_{max}' rule. According to Definition 1, this implies that there would be a reduction in

makespan if another machine of the same type as M_k is made available at this scheduling instant $t_{now} = 0$. This is impossible since the current makespan is computed assuming infinite resource availability. Hence referring to M_k as the bottleneck machine violates Definition 1.

The SB heuristic uses the 'highest L_{max}' rule not just for the scheduling instant $t_{now} = 0$ but also for every iteration. Hence all machines would be identified as a bottleneck at one iteration or other. Thus admitting Myth 8 as true is equivalent to denying the existence of a non-bottleneck machine as a concept.

The concept of using the highest L_{max} value as the bottleneck predictor can be traced back to the paper by Lageweg et al. (1977). Among two different branching schemes presented for solving the job shop problem, one was that of settling essential conflicts. This involved selecting a branching pair from a 'conflict set' containing pairs of operations whose processing intervals overlapped. To select a particular member of this conflict set, Lageweg et al. proposed a two-stage strategy. In the first stage, the machines were ordered according to non-increasing value of the one machine lower bound. This was done since the authors felt it was 'a natural way' to select a machine on which at least one conflict exists. In the second stage, the choice was restricted to members of the conflict set which were to be processed on the machine with highest value of one machine lower bound.

The ordering of machines by the one machine lower bound is equivalent to the ordering by L_{max} value. Thus, in the initial stage of the SB heuristic when none of the machines is scheduled, the machine with the highest L_{max} value is the machine with the most amount of conflict. Hence it is understandable that the SB heuristic aims at settling the schedule on the most conflicting machine. In this manner, the main idea of the SB heuristic is similar to that of Lageweg et al. (1977), except that in the latter only one disjunctive pair is settled while in the former, all the disjunctions belonging to the selected machine are settled.

Perhaps a better descriptor of the machine-based decomposition procedure adopted by Adams et al. (1988) would be the Shifting Conflict heuristic. The term conflict gives us an idea about the internal state of a system when some operations are yet to be scheduled. The term bottleneck is not interchangeable with the term conflict. In fact, for any feasible schedule, none of the machines would have any conflict. Yet there may exist more than one bottleneck.

In the absence of a formal bottleneck definition acceptable to all, there is a danger that terms like 'Shifting Bottleneck' may mean many things to many people. For example, temporal shifting of bottlenecks in production environments has been a well researched area. Such shiftiness can arise out of various causes like changes in the product mix, machine breakdowns and other random events and even management decisions based on performance objectives (Hurley and Kadipasaoglu (1998)). Temporal shiftiness of bottlenecks has nothing to do with the 'shiftiness' of the Shifting Bottleneck heuristic. Yet books (Morton and Pentico (1993), p. 28) and papers (Lawrence and Buss (1994)), (Moss and Yu (1999)) discussing the temporal shiftiness of bottlenecks talk in the same breath about the bottleneck shiftiness in SB heuristic. This highlights the fact that practitioners and researchers not conversant with the intricacies of the SB heuristic may erroneously ascribe certain characteristics to the terms 'shifting' and 'bottleneck' which were not envisaged by the original researchers.

3.5 System Performance based bottleneck definitions

Several authors have used bottleneck definitions based on the performance of a system. Billington et al. (1986) define a bottleneck as the workcentre that limits the production rate of the entire system. Chiang et al. (1998) state that "Intuitively, bottleneck (BN) of a production line is understood as a machine that impedes the system performance in the strongest manner". Wu (2005) define bottleneck as the constraint that prevents a factory from attaining its production goals. The author notes that different kinds of bottlenecks may exist depending on different production goals. "A throughput bottleneck prevents a factory from achieving a higher throughput rate, while a cycle time bottleneck prevents it from achieving shorter total cycle time." Definition 1 is clearly similar to this idea of a bottleneck. A bottleneck is ultimately defined in terms of the decision maker. Given the same job shop, the decision maker can be interested in minimising makespan or the average flow time. The bottleneck machine for the minimisation of makespan objective can be different from the one for minimisation of average flow time objective.

The bottleneck definition adopted by us falls under the category of system performance based bottleneck definitions. While we do not claim to be the first to link the definition of bottleneck to system performance, our approach differs from others in providing a rigorous shadow price based theoretical framework on which the definition is based.

It may be pointed out that all our arguments are based upon the validity of Definition 1. A definition cannot be proved or disproved, it requires acceptance and a basic criteria of acceptance is whether the definition proposed is in conformity with other definitions used in that branch of science. Our belief in Definition 1 stems from the fact that it matches the idea of shadow price for a constraint in mathematical programming. We review the literature on shadow prices for integer programmes in the next section.

4 Average shadow price

Kantorovich (1939) introduced the concept of shadow price and demonstrated that at optimality we can associate a shadow price with every resource. He provided an economic interpretation of the dual variables as "guides for the coordination of allocative decisions", (Koopmans (1976)). The active constraints in the optimal solution of a linear programme could be thought of as bottlenecks which constrain the achievement of a better objective function value. Unfortunately, the concept of shadow price in integer programming is not as straightforward as its linear counterpart.

The first attempt to determine shadow prices in integer programming was by Gomory and Baumol (1960). But their shadow prices suffered from various theoretical imperfections like a free good having a positive price. Alcaly and Klevorick (1966) could rectify some, but not all, of these imperfections. Geoffrion (1974) applied Lagrangian relaxation based approaches, but the shadow prices determined were not unique and a free good could have a non-zero shadow price. A shadow price for integer programming with valid economic interpretation eluded researchers, until Kim and Cho (1988) introduced the concept of average shadow price.

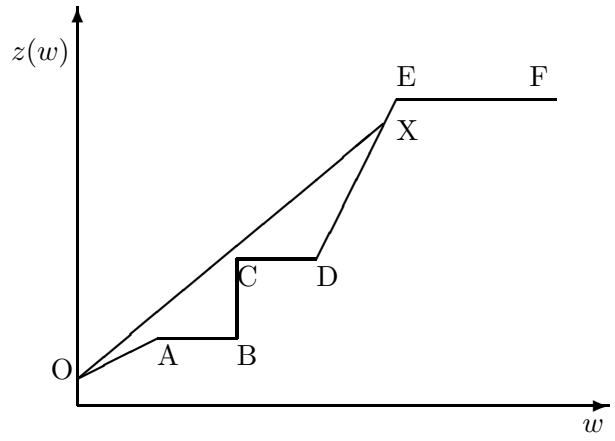

Figure 1: Graphical representation of Average Shadow Price

Consider the integer programming formulation $z_{IP} = max\{cx : Ax \leq b, x \in S\}$, where $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$, $c \in \mathbb{R}^n$ and $S = \{x : Gx \leq h, x_j(j = 1, \ldots, n)$ are nonnegative integers$\}$. For resource k, the perturbation function z_k is defined as $z_k(w) = max\{cx : a_i x \leq b_i(i \neq k), a_k x \leq b_k + w, x \in S\}$. Then the Average Shadow Price $(ASPC)$ for continuous perturbation of resource k is defined as $ASPC_k = sup\{(z_k(w) - z_k(0))/w\}$, $w > 0$. In Figure 1, $O - A - B - C - D - E - F$ is the piecewise linear curve obtained by plotting the objective function value $z_k(w)$ as w increases. Let X be any point on this curve. The average shadow price $ASPC_k$ is then the maximum gradient of the straight line OX.

Kim and Cho (1988) showed that $ASPC_k$ has the characteristic that its existence and uniqueness in the mathematical sense can be guaranteed. They also obtained a version of the complementary slackness theorem in integer programming for such average shadow prices. Crema (1995) extended the concept of average shadow price to the Mixed Integer Linear Programme (MILP) domain. Instead of perturbing one resource at a time, he considered perturbing a 'package' (i.e. combination) of resources.

Most models of right hand side (RHS) parametric analysis (Geoffrion and Nauss (1977), Crema (1999)) focus on perturbing the RHS in a continuous fashion. However, continuous perturbation does not have a realistic significance for problems where the resources are available in discrete units. Such problems can be of varied nature like the knapsack problem, the capacitated plant location problem and scheduling problems. For production/service environments where fractional availability of resources have no realistic significance, we need only be concerned with an integer vector b. Specifically, a Integral RHS Integer Linear Programme (IRILP) and its parametric version are defined as follows.

IRILP: Given rational $m\times n$ matrix A, integer vector b and rational vector c of conformable dimensions, determine $z_{IR} = max\{cx : Ax \leq b; x \in \mathbb{Z}^n\}$.

PIRILP(w)**:** Given rational $m \times n$ matrix A, integer vector b and rational vector c of conformable dimensions, with $w \in \mathbb{Z}^1_+$ and $\Delta b \in \mathbb{Z}^m$, determine $z_{IR}(w) = max\{cx :$

$Ax \leq b + w\Delta b; x \in \mathbb{Z}^n\}$. The vector Δb is constructed such that its ith element is strictly positive if and only if the corresponding constraint represents a resource which is being perturbed.

Sensitivity and parametric analysis in linear programmes are carried out under the assumption that w can be varied continuously. In contrast, any shadow price calculated for an IRILP will have a realistic significance only for integral w. In order to take care of integrality of resource availability, Mukherjee and Chatterjee (2006) introduced the Average Shadow Price for Integer Resource Availability (*ASPIRA*). Specifically, $ASPIRA = sup\{(z_{IR}(w) - z_{IR}(0))/w\}$, $w \in \{\mathbb{Z}_+^1\}$ for a maximisation problem. For a minimisation problem, $ASPIRA = sup\{(z_{IR}(0) - z_{IR}(w))/w\}$, $w \in \{\mathbb{Z}_+^1\}$. Mukherjee and Chatterjee (2006) also introduced the Marginal Unit Shadow Price (*MUSP*) as the difference between $z_{IR}(1)$ and $z_{IR}(0)$. It corresponds to the concept of Marginal Shadow Price (*MSP*) in linear programmes where resource availability can be varied infinitesimally. The extension of the concepts of *ASPIRA* and *MUSP* to MILP problems is straightforward.

5 An universal bottleneck definition

Definition 3 *A set of constraints with strictly positive average shadow price is defined as a bottleneck.*

Definition 3 identifies a bottleneck as a specific set of constraints. This set can be composed of a single constraint or a collection of constraints which together represent a resource. For example, the set of constraints specifying the availability of a resource for each time bucket in a time indexed formulation of a scheduling problem may be a bottleneck. Given a set of constraints, the mode of perturbation of the RHS would determine whether *ASPC* or *ASPIRA* would be the valid measure of average shadow price. Using *ASPC* assumes that a manager can alter resource availability fractionally. In case this assumption is violated, *ASPIRA* would be more relevant measure of average shadow price.

Since Definition 3 is based on the concept of average shadow price, it has the characteristics that (i) it respects the complimentary slackness conditions and (ii) its existence and uniqueness is guaranteed. In addition, the integral availability of resources can be handled using *ASPIRA*. Furthermore, the attractiveness of this definition stems from its wide applicability. It can be used for finite planning horizon project and job shop environments as well as infinite horizon repetitive manufacturing environments. It does not assume any structural properties of the shop floor nor are decision maker's objectives hard coded into it. It does not specify any particular mode of capacity addition. Definition 3 provides a single reference point from which bottleneck definitions in particular production/service situations can be derived.

Given a production/service environment, there may be several alternate ways of improving the performance of the system. For example, a manager may evaluate two options of increasing the throughput of a shop - (i) by increasing the availability of a particular machine M_k and (ii) by increasing the processing speed of M_k. The bottleneck values of both options can be determined using *ASPIRA* if the speed of M_k can only be increased

in discrete steps. If the speed of M_k can be uniformly varied, $ASPC$ would be the relevant measure for the bottleneck value of the second alternative. In this paper, when we determine the bottleneck value of a resource, we assume that the only way of increasing system performance is by increasing resource availability.

5.1 Project environment

The simplest form of a project environment involves scheduling activities given a set of precedence relationships such that the project completion time is minimised. The duration of each activity can be thought of as a constraint that prevents a lower project completion time. In order to identify the bottleneck activity we need to ask the question 'Would there be a decrease in project completion time if the duration of a particular activity is *decreased*?' The $ASPC$ values of each activity can be determined by quantifying the extent of decrease in project completion time as the activity duration is shortened.

Theorem 1 *An activity is a bottleneck if and only if it is present on all critical paths in the optimal project schedule.*

Proof: If the activity is present on all critical paths then the project duration can be decreased by decreasing the duration of the activity. Thus this activity would have a strictly positive $ASPC$ and hence would be a bottleneck. The project duration cannot be shortened if all critical paths are not simultaneously shortened. Hence $ASPC = 0$ for an activity not present on all critical paths and as a result it is a non-bottleneck. □

The set of activities with strictly positive $ASPC$ values would be equivalent to the set of activities on the critical path for a project with a single critical path. It would be a subset of the set of critical activities if multiple critical paths exist. If the activity durations can only be decreased in integer steps then we would replace $ASPC$ with $ASPIRA$. The bottleneck values of activities can be similarly determined for time-cost trade-off ('crashing') problems with no resource constraints. The following theorem identifies situations when a *critical chain* can be termed a bottleneck.

Theorem 2 *A* critical chain *is a bottleneck if and only if shortening the* critical chain *shortens all critical paths simultaneously.*

Proof: If shortening the *critical chain* shortens all critical paths simultaneously, then the *critical chain* would have a strictly positive $ASPC$ value and hence would be a bottleneck. On the other hand if there exists at least one critical path which remains unaffected, the project makespan remains unaffected. Hence the $ASPC$ value for the *critical chain* is zero implying a non-bottleneck Hence the result. □

Several types of resources have been considered by the Resource Constrained Project Scheduling Problem (RCPSP) literature. Talbot (1982) presented time-resource trade-off

models involving renewable, non-renewable and doubly-constrained resources. Partially renewable resources (Böttcher et al. (1999)) cover the renewable and non-renewable resource constraints as special cases. Ahn and Erenguc (1998) combined time-cost and time-resource trade-offs into one formulation. We can define a bottleneck for a RCPSP as any resource with strictly positive *ASPIRA* value. While the RCPSP literature considers scarce resources, we are unaware of any study which considers the value of a scarce resource. If a project manager wishes to augment capacity, a need may arise to choose a subset of the scarce resources due to budgetary limitations. In such a situation, the *ASPIRA* value of a resource would provide valuable guidance to the project manager.

The capacity of a system is not only dependent on the capacity of individual resources but also on the flexibility of resources. Increasing the flexibility of resources would imply an increase in capacity of the entire system. Given a production/service system, a manager may be interested in identifying the resource which would contribute the most to the objective function if the resource is made flexible. A framework for determining the value of resource flexibility in the resource constrained project environment is presented in Vairaktarakis (2003). The bottleneck in such an environment can be identified by determining the suitably defined *ASPIRA* value for each resource.

5.2 Job Shops

With Definition 3, we are also able to surmount the difficulties faced by Adams et al. (1988) in defining a bottleneck in job shop scheduling. They had identified three measures of bottleneck value which included (i) criticality of a machine (ii) marginal utility of a machine in reducing makespan and (iii) the 'highest L_{max}' measure discussed earlier in this paper. Comparing the bottleneck measures, they noted the following.

> In order to prioritize the machines, we need a concept that expresses the bottleneck quality as a matter of degree rather than a yes or no property. This quality could be measured, for instance, by the marginal utility of the machine in reducing the makespan, were it not for the practical difficulty of assessing the later.

The concept of Marginal Unit Shadow price (*MUSP*) introduced in Mukherjee and Chatterjee (2006) is a rigorous definition of marginal utility of a machine as perceived by Adams et al. (1988). However, Definition 3 is based on the average and not the marginal shadow price. In fact, as we show in Example 7, *ASPIRA* and *MUSP* may differ for a scheduling problem. Hence Definition 3 represents an advancement in the understanding of what constitutes a bottleneck in scheduling.

5.3 Batch Production

Batch production environments are characterised by moderate product volume and variety. Such environments occur, for example, in chemical, pharmaceutical and food processing

industries where large number of products share the same production assets and are produced cyclically in batches. Certain intermediate products may need to be stored in tanks having limited capacity. In such cases we can determine the bottleneck value not only for each machine, but also for the storage tanks.

5.4 Assembly Lines

Given an assembly line, a manager may wish to identify the bottleneck workstation. For a single model assembly line, this analysis is similar to the one presented for Continuous Flow Lines in next subsection. For mixed model assembly lines, Drexl and Kimms (2001) provide a mathematical formulation where at most H_o out of N_o successively sequenced units may require an option $o \in O$. Then H_o operators or installation teams are required for installation of option o. We can derive the *ASPIRA* values for all such installation teams and identify the bottleneck team.

A different type of bottleneck analysis can be performed for labour intensive assembly lines with worker cross functionality. In such a case, a worker can be assigned to any workstation and we assume that the number of workstations equals the number of direct workers. Type II Assembly Line Balancing problems (Uğurdağ et al. (1997)) consider the objective of minimisation of cycle times (equivalently, maximisation of throughput) given a number of workstations. The bottleneck value of labour as a resource class can be computed if we parametrically solve the Type II problem by changing the number of workstations allowed.

5.5 Continuous Flow Lines

In the most extreme form, a Continuous Flow Line may produce only one product day in and day out. More general are Continuous Flow Lines that process a small number of product varieties. The dominant objective in such a production environment is to compete on price by reaping economies of scale. This translates to an objective of maximising throughput of the line. There can be identical parallel processors at a particular stage and it makes more sense to talk about the bottleneck stage rather than the bottleneck machine.

Consider the process flow of a 1-commodity Continuous Flow Line. We construct a network $G = (V, E)$ where node $v \in V$ is a processing stage having capacity c_v. Two nodes are connected with an arc $e \in E$ if they are adjacent stages in the processing route. Two nodes S and T, denoting the source and sink nodes, are added to G. The source and sink nodes model the supply and demand rates for the commodity and c_S and c_T are set to supply and demand rates respectively. Then the 1-commodity Continuous Flow Line can be represented as the problem of maximising flow of the commodity over the network G. Given a feasible allocation of activities to workstations, a single model assembly line can be similarly modeled. In such a situation, the capacities of each workstation would be the inverse of the workload of that station.

A 'cut' in network G is a collection of nodes whose removal disconnects the source-sink pair of nodes and no proper subset of the cut should have this property. The capacity of

the cut is the sum of capacities of nodes constituting the cut. The node-constraint version (Hu (1969), p. 216) of the celebrated 'max-flow min-cut' theorem of Ford and Fulkerson (1956) states that for single commodity network flows, the maximum flow over the network is equal to the capacity of the minimum cut.

Theorem 3 *For a production/service environment where the 'max-flow min-cut' theorem holds, the bottleneck(s) are the node(s) that constitute the minimum cut.*

Proof: If node v is not a constituent of the minimum cut, an addition of capacity at node v would not change the capacity of the minimum cut and hence that of the maximum flow value. Thus the *ASPIRA* value of node v would be zero. On the other hand, if node v is a constituent of the minimum cut, addition of capacity at node v changes the maximum flow value, implying a strictly positive *ASPIRA* value for node v. Hence the result. □

Corollary 1 *For a single model assembly line or a 1-commodity Continuous Flow Line, the bottleneck is the stage with lowest capacity.*

Proof: Both the single model assembly line and the 1-commodity Continuous Flow Line can be modeled as the problem of maximising flow over a 1-commodity network. Hence the 'max-flow min-cut' theorem holds for both environments. The network structure for both environments imply that each cut consists of a single node. Hence there exists an equivalence between cuts and stages. Using Theorem 3, the bottleneck is the node having lowest capacity. Hence the result follows. □

We are not the first to relate the concept of bottleneck to the capacity of the minimum cut (Hu (1969), p. 107). Our contribution lies in showing that a valid definition of bottleneck for maximal flows in networks is a special case of an universally valid bottleneck definition. Further, operations managers are oriented towards stage-based capacity calculations and hence tend to identify one of the stages as the bottleneck. However, Theorem 3 defines bottlenecks in terms of cuts and not stages. If we have multiple commodities with different routings, a cut may consist of several nodes. Consequently, the bottleneck in such cases may be a combination of stages constituting the minimum cut. In case if supply or demand are constrained, the bottleneck may even be a combination of stages and/or supply capacity/demand rate. For example, consider the cranberry processing situation described in Shapiro (2002). Identifying the drying operation as the bottleneck is incomplete since the cut consists of the dryers along with the dry berry supply rate.

Several complications arise as we move from 1-commodity network flows to the multi-commodity case. Firstly, it is not necessary that the 'max-flow min-cut' theorem holds for all production/service environments which can be modeled as maximal flows in networks. Network structures where the theorem does not hold are presented in Schrijver (1990). Secondly, in certain cases like mixed-model assembly lines, we may need to model integer flows. Thirdly, the literature on multi-commodity network flows mostly consider the case where a commodity can take any route between its source and sink nodes. However, in

modeling a k-commodity production environment as a flow over a network, we may need to make sure that a commodity can move from source to sink over a specific route only. Hence, to derive a result similar to Corollary 1, we need to check whether the 'max-flow min-cut' theorem holds for a production/service environment. Finally we need to explore the possibility of special cases where minimum cut capacity remains a valid bottleneck definition even when the 'max-flow min-cut' theorem does not hold for the underlying network. Of particular interest can be situations where approximate versions of the theorem are known to be valid (Leighton and Rao (1999)).

6 Determination of bottleneck values

Consider a production/service environment which can be represented as a minimisation problem. The *ASPC* value of a resource can be determined by solving a series of non-parametric MILP problems (Crema (1995)). If availability of resource R can only be changed in discrete steps, we consider the perturbation of the production/service environment represented by the PIRILP $z_{IR}(w) = min\{cx : Ax \leq b + w\Delta b; x \in \mathbb{Z}^n\}$; where the Δb vector is suitably defined. The bottleneck value of resource R can be determined as follows.

Step 1: Perform an iterative procedure of determining the optimal solution value $z_{IR}(w)$ for particular values of $w = 1, 2, 3, \ldots$.

Step 2: Determine $ASPIRA = sup\{(z_{IR}(0) - z_{IR}(w))/w, w \in \{\mathbb{Z}^1_+\}\}$

There are two main ways in which this analysis can be performed: (i) mathematical programming based procedures and (ii) other algorithm based procedures. Within mathematical programming based approaches, time-indexed formulations allow us to model the resource availability as a right hand side (RHS) vector, similar to the representation in $z_{IR}(w)$. Time-indexed formulations with machine availability modeled as a RHS vector are available for projects (Talbot (1982), Böttcher et al. (1999)), single machine scheduling problem (Sousa and Wolsey (1992), Van den Akker et al. (2000)), job shop problem (Fisher (1973)), batch production (Kondili et al. (1993)) and assembly lines (Drexl and Kimms (2001)). A general framework for modeling production systems has been proposed by Hackman and Leachman (1989). It provides a meta model where resource availability is modeled as a RHS vector. Hence the *ASPIRA* values can be determined for any production situation which can be formulated in terms of this general framework.

The time-indexed formulation based procedure has its drawbacks. For any reasonable problem size, the number of variables in the time-indexed formulation would become exceedingly large rendering difficult any such effort (Van den Akker et al. (2000)). Note that the optimal solution value of a mathematical programme is independent of its formulation type. The *ASPIRA* value ultimately depends on the optimal objective function value and hence exists irrespective of whether we could model the resources as right hand side availability.

Non-mathematical programming approaches require an optimal solution procedure which can handle multiple units of a resource type. Most RCPSP solution techniques have this desirable property. However, in scheduling research, parallel machine scheduling problems

are relatively less extensively studied than their single machine counterparts. Hence, in such situations, we need to first identify or develop an optimal solution procedure for the parallel machine version. For example, suppose we are concerned with determining the *ASPIRA* value for the $1|r_j|L_{max}$ problem, a single machine scheduling problem with release times and due dates. We need to find an optimal solution procedure for the $P|r_j|L_{max}$ problem where identical machines are available in parallel. If there are n jobs to be performed on this machine, then we need to solve a maximum of n $P|r_j|L_{max}$ problems to determine the *ASPIRA* value. A solution method for the $P|r_j|L_{max}$ problem has been discussed in Carlier (1987). Mokotoff (2001) provides a review of parallel machine scheduling problems. Hybrid flow shop problems (Linn and Zhang (1999)) deal with multistage flow shops with multiple machines in each stage. The multistage job shop problem with identical parallel machines in each stage is a specific case of the Flexible Job Shop problem (Sule and Vijayasundaram (1998)).

For repetitive manufacturing environment, the five-step focusing procedure of Goldratt can be adapted for determination of bottleneck values. One of the five steps, 'Elevate the constraint', is similar to the perturbation analysis that we propose. The significant difference is that the five-step procedure does not quantify the value of elevating the constraint. This value has managerial significance because one may decide against elevating the constraint if the value gained is less than the cost involved. Secondly, the five-step procedure does not help us determine the quantum of additional resource that should be employed to elevate the constraint. The work presented in this paper fills this gap. If the *ASPIRA* value of a production/service stage equals a/b then the value gained by adding b additional resources at that stage is precisely a.

7 Equivalence of the average and marginal shadow price

The determination of the applicable marginal shadow price (*MSP* or *MUSP*) may be far easier than the applicable average shadow price(*ASPC* or *ASPIRA*). Hence our first effort in any production/service environment would be to check whether the two shadow prices are equivalent. Even if it is not possible to prove the equivalence for a problem class, we may be able to deduce the equivalence for a particular problem instance. In this section we first provide some general results and then proceed to investigate the equivalence or otherwise for specific environments.

7.1 General results

Consider the PIRILP $z_{IR}(w) = min\{cx : Ax \leq b + w\Delta b; x \in \mathbb{Z}^n\}$ representing the perturbation of a production/service environment. Note that while the resource availability is changed by suitably changing the RHS, the parameters A and c remain unchanged. Thus the 'technology' employed is deemed to be constant. The Law of Diminishing Returns (LDR) holds for resource R if $z_{IR}(w) - z_{IR}(w+1) \geq z_{IR}(w+1) - z_{IR}(w+2)$ for any $w \in \{\mathbb{Z}^1\}$. The law can be similarly defined for maximisation problems as well as for MILP problems. As a concept, the Law of Diminishing Returns may appeal to practitioners as

intuitive. However, it may not hold for a production environment as shown in Example 7 later in this paper.

Theorem 4 *The Law of Diminishing Returns is a sufficient but not necessary condition for the equivalence of average and marginal shadow prices.*

Proof: Example 7 shows a situation where $ASPIRA = MUSP$ even when LDR does not hold. If LDR holds, $MUSP = z_{IR}(0) - z_{IR}(1) = sup_w\{z_{IR}(w) - z_{IR}(w+1)\}$, $w \in \{\mathbb{Z}^1\} = sup_w\{(z_{IR}(0) - z_{IR}(w))/w\}$, $w \in \{\mathbb{Z}^1\} = ASPIRA$. The equivalence of *ASPC* and *MSP* can also be derived if LDR holds (Mukherjee and Chatterjee (2006)). Hence the result follows. □

LDR holds for any production/service environment which can be modeled as linear programme. It also holds for any integer programme where (i) the coefficient matrix A is totally unimodular (TUM) and (ii) the RHS is integer (Mukherjee and Chatterjee (2006)). An integer b vector implies that apart from resource constraints, the RHS of all other constraints are also integral. This can be ensured by suitable algebraic modifications as long as all such RHS values are rational. Our survey of mathematical formulations of production environments suggests that this assumption is not unrealistic. The concept of *ASPIRA* is valid even if this assumption is violated. However, in such a situation, we cannot claim the equivalence of *ASPIRA* and *MUSP* by showing A as TUM. In any case, it is unlikely that majority of production/service environments can be represented as a linear programme or an integer programme with TUM coefficient matrix.

We need not solve for all possible w values to check the equivalence of the two shadow prices. Consider a parallel machine scheduling problem where the objective function needs to be minimised. For such problems, $z_{IR}(\infty)$ may be easy to determine as it represents the case where sufficient number of machines are available for each job. A similar situation exists for project scheduling subject to a single resource constraint. Let $K = z_{IR}(0) - z_{IR}(\infty)$ and $k = K/(z_{IR}(0) - z_{IR}(1))$. We start by determining $z_{IR}(0)$, $z_{IR}(1)$ and set $ASPIRA = (z_{IR}(0) - z_{IR}(1))$. At iteration w, we check if $(z_{IR}(0) - z_{IR}(w))/w > ASPIRA$ and update *ASPIRA* accordingly.

Theorem 5 *A maximum of k iterations are required to determine* ASPIRA *value.*

Proof: Without loss of generality, let $ASPIRA = (z_{IR}(0) - z_{IR}(w))/w, w \in \{\mathbb{Z}^1_+\}$. Hence $(z_{IR}(0) - z_{IR}(w))/w \geq z_{IR}(0) - z_{IR}(1)$. From definition of k, $z_{IR}(0) - z_{IR}(\infty) = k(z_{IR}(0) - z_{IR}(1))$. But $z_{IR}(0) - z_{IR}(\infty) \geq z_{IR}(0) - z_{IR}(w)$.
$\Rightarrow k(z_{IR}(0) - z_{IR}(1)) \geq z_{IR}(0) - z_{IR}(w) \geq w(z_{IR}(0) - z_{IR}(1))$
$\Rightarrow k \geq w$. □

Theorem 6 ASPIRA = MUSP *if* $k \leq 2$.

Proof: For $k = 2$, $z_{IR}(0) - z_{IR}(1) = z_{IR}(1) - z_{IR}(\infty)$. Hence $ASPIRA = MUSP$ since $z_{IR}(0) - z_{IR}(1) \geq (z_{IR}(0) - z_{IR}(w))/w, w \in \{\mathbb{Z}^1_+\}$. For $k < 2$, using Theorem 5, the only

Table 6: Data for identical parallel machine scheduling problem

Job	r_i	p_i	q_i
J_1	1	5	6
J_2	2	6	4
J_3	4	7	8
J_4	5	8	5
J_5	13	2	2

feasible w value is $w = 1$. Hence the result follows. □

Let $f(1)$ represent objective function value of any feasible solution to the minimisation problem with one additional machine in parallel. Let $f = K/(z_{IR}(0) - f(1))$.

Corollary 2 ASPIRA = MUSP *if* $f \leq 2$.

Proof: Follows from Theorem 6 noting that $f \geq k$ since $z_{IR}(0) - f(1) \leq z_{IR}(0) - z_{IR}(1)$. □

Consider the data for an identical parallel machine scheduling problem provided in Table 6. Each job J_i has a release time r_i, a processing time p_i and a 'tail' q_i. The objective is to schedule the jobs such that makespan is minimised. The non-preemptive version of this scheduling problem is known to be NP-hard (Carlier (1987)). We determine $z_{IR}(\infty) = max_i\{r_i + p_i + q_i\} = 19$. The optimal sequence is $J_1 - J_2 - J_3 - J_4 - J_5$ with makespan of 31 when only one machine is available. When two identical parallel machines M_1 and M_2 are present, a feasible schedule is obtained by scheduling jobs $J_1 - J_4 - J_5$ on machine M_1 and $J_2 - J_3$ on machine M_2, resulting in a makespan of 23. Hence, using Corollary 2, $ASPIRA = MUSP$ for this problem instance as $f = (31 - 19)/(31 - 23) = 1.5$. The actual determination of the $ASPIRA$ value would then require solving only one NP-hard problem. We do not need to solve $z_{IR}(w)$ for $w = 2, 3, 4$.

7.2 Project crashing

Consider a project without any resource constraints. The project makespan can be decreased by crashing the duration of an activity, but an incremental crash cost has to be incured. Assume convex cost-time trade-off curves. The duration of activity i can vary between the normal duration N_i and crash duration M_i. Two different bottleneck values would be associated for each activity. ASP_i^M (ASP_i^N) would specify the bottleneck value of *decreasing* M_i (*increasing* N_i). A strictly positive ASP_i^N value implies that the project duration can be compressed if activity i is lengthened.

Theorem 7 $ASP_i^M = MSP_i^M$ *and* $ASP_i^N = MSP_i^N$ *for unconstrained project crashing with convex cost-time trade-off.*

Table 7: Solution of parametric $P||C_{max}$ problem instance

Number of machines $(w+1)$	Allocation	Makespan	Improvement	$(z_{IR}(0) - z_{IR}(w))/w$
1	(5)	5		
2	(3, 2)	3	2	2
3	(2, 2, 1)	2	1	1.5
4	(2, 1, 1, 1)	2	0	1
5	(1, 1, 1, 1, 1)	1	1	1

Proof: The time-cost trade-off problem in a project environment with convex cost-time trade-off and no resource constraints can be represented as a linear programme with M_i and N_i in RHS (Wiest and Levy (1977), p. 79). Since LDR holds for any linear programme, the result follows using Theorem 4. □

7.3 The $P||C_{max}$ problem

Consider the $P||C_{max}$ problem where jobs have to scheduled on identical parallel machines to minimise makespan. All job are available at start time and preemption is not allowed. While the $1||C_{max}$ problem is trivial, the $P||C_{max}$ problem is known to be NP-hard (Garey and Johnson (1979)). The number of identical machines available for scheduling is an input to the $P||C_{max}$ problem. In reality, an operations manager may have m number of identical machines available for scheduling. We denote such a situation as the $P||C_{max}$ problem at base m. We now show in Example 7 that (i) in general $ASPIRA \neq MUSP$ for the $P||C_{max}$ problem and (ii) for the $P||C_{max}$ problem at base 1, $ASPIRA = MUSP$ even though LDR does not hold.

Example 7 Consider a single machine scheduling problem with minimisation of makespan objective. Five identical jobs are to be scheduled on this machine, each job having unit processing time. All jobs are available for scheduling. The makespan for this trivial problem is 5. The *ASPIRA* value is determined by considering the identical parallel machine perturbation of this problem. Table 7 shows solution of the trivial parallel machine problems. An allocation of (3, 2) implies that three jobs are scheduled on machine M_1 while two are scheduled on M_2. The improvement column provides the decrease in makespan achieved by adding one more machine.

It is obvious from the improvement column that the Law of Diminishing Returns does not hold for this problem instance. In this case $ASPIRA = max\{2/1, 3/2, 3/3, 4/4\} = 2/1$ and hence $ASPIRA = MUSP$. However, if our base problem was to schedule five identical unit processing time jobs on three machines, then for that problem, $MUSP = 0$ while $ASPIRA = 1/2$ and hence $ASPIRA \neq MUSP$. Note that we represent *ASPIRA* values as a fraction rather than a real number as it provides additional information.

7.4 The $1||C_{max}$ problem

Even though the equivalence of *ASPIRA* and *MUSP* does not hold in general for the $P||C_{max}$ problem, there can be certain problem subsets where *ASPIRA* = *MUSP*. Specifically we are interested in the $1||C_{max}$ problem which can be looked upon as a $P||C_{max}$ problem with only one machine available in the base configuration. Consider the PIRILP $z_{IR}(w)$ for the $1||C_{max}$ problem with n jobs, job J_i having processing time p_i. If $L = \sum_i p_i$, then $LB(w) = L/(w+1)$ is a lower bound for $z_{IR}(w)$. Let p_{max} denote the maximum processing time among all jobs.

Theorem 8 $z_{IR}(1) - LB(1) \leq (1/2)p_{max}$.

Proof: Without loss of generality, let M_1 be the most loaded machine and let J_k be any job scheduled on M_1 in the optimal schedule. Since $LB(1) = L/2$, sum of processing times on M_2 equals $LB(1) - (z_{IR}(1) - L/2)$. Then $p_k \geq z_{IR}(1) - (LB(1) - (z_{IR}(1) - L/2)) = 2(z_{IR}(1) - LB(1))$ since otherwise it would be possible to reduce makespan by shifting job J_k to M_2. Hence the result follows. □

Theorem 9 ASPIRA = MUSP *for* $1||C_{max}$ *problem.*

Proof: There exists two possibilities:
Case I: Suppose $p_{max} < L/3$. Then using Theorem 8, $z_{IR}(1) - LB(1) \leq (1/2)p_{max} < L/6$. Hence $z_{IR}(1) < 2L/3$ since $LB(1) = L/2$. Thus $z_{IR}(0) - z_{IR}(1) > L - 2L/3 = L/3$. Since $z_{IR}(0) - z_{IR}(w) \leq L$ for any integer w, $(z_{IR}(0) - z_{IR}(w))/w \leq L/3$ for any integer $w > 2$. For $w = 2$, $LB(2) = L/3$ and hence $z_{IR}(0) - z_{IR}(2) \leq z_{IR}(0) - LB(2) = 2L/3$. Hence *ASPIRA* = *MUSP* since $z_{IR}(0) - z_{IR}(1) \geq (z_{IR}(0) - z_{IR}(w))/w$ for any integer w.
Case II: Suppose $p_{max} \geq L/3$. Then $z_{IR}(0) - z_{IR}(\infty) = L - p_{max} \leq L - L/3 = 2L/3$. Hence $(z_{IR}(0) - z_{IR}(w))/w \leq (z_{IR}(0) - z_{IR}(\infty))/w \leq L/3$ for any integer $w \geq 2$. A feasible solution for the $2||C_{max}$ problem involves scheduling the job with largest processing time on one machine and all other jobs on the other. If $f(1)$ denotes the makespan of this feasible schedule then $f(1) = max\{p_{max}, L - p_{max}\}$. If $p_{max} \geq L/2$ then $f(1) = z_{IR}(1) = p_{max}$ and hence *ASPIRA* = *MUSP* since $z_{IR}(w) = p_{max}$ for any integer w. If $p_{max} < L/2$, then $f(1) = L - p_{max}$. Thus $z_{IR}(0) - z_{IR}(1) \geq z_{IR}(0) - f(1) = p_{max} \geq L/3$. Hence *ASPIRA* = *MUSP* since $z_{IR}(0) - z_{IR}(1) \geq L/3 \geq (z_{IR}(0) - z_{IR}(w))/w$ for any integer w. □

The implication of Theorem 9 is that 'the marginal utility of the machine in reducing the makespan' (Adams et al. (1988)) is indeed the rigorous definition of bottleneck for the $1||C_{max}$ problem. Having established *MUSP* as the applicable bottleneck measure, the next step would be to determine the bottleneck value for a problem instance. This involves solving the $2||C_{max}$ problem, known to be NP-hard (Garey and Johnson (1979)). An optimal $O(2^n)$ solution procedure for the $2||C_{max}$ problem has been presented in Ho and Wong (1995). The difference in optimal makespans of the $1||C_{max}$ problem and the corresponding $2||C_{max}$ problem would then be the valid bottleneck value for the $1||C_{max}$ problem instance.

7.5 Continuous Flow Lines

A result similar to Theorem 9 can also be derived for any production/service environment where the 'max-flow min-cut' theorem holds. Using Theorem 3, each member of the minimum cut is a bottleneck. Without loss of generality, let the minimum cut be cut k and the next lowest capacity be for cut j. Let resource R be a constituent of cut k. We assume that the capacity of resource R can only be added in discrete steps of C. Let $z_k(w)$ denote the maximum flow through this system when the capacity of cut k is incremented by adding w units of resource R, each having capacity C.

Lemma 1 *For any production/service environment where the 'max-flow min-cut' theorem holds,* $z_k(w+1) - z_k(w) \leq C;\ \forall w \in \{\mathbb{Z}^1\}$.

Proof: Either of two possibilities exist.
Case I: At iteration w, cut k remains the bottleneck after addition of capacity C. Thus the difference in capacity of the minimum cut before and after addition of capacity equals C. Hence, using the 'max-flow min-cut' theorem, $z_k(w+1) - z_k(w) = C$.
Case II: Otherwise, cut j becomes the bottleneck and hence $z_k(w+1) - z_k(w) < C$. Further, $z_k(p+1) - z_k(p) = 0;\ \forall p \geq w+1,\ p \in \{\mathbb{Z}^1\}$. Combining, the result follows. □

Theorem 10 ASPIRA = MUSP *for each resource in a production/service environment where the 'max-flow min-cut' theorem holds.*

Proof: If resource R is not part of the minimum cut then $ASPIRA_R = MUSP_R = 0$. Otherwise two possibilities exist.
Case I. $z_k(1) - z_k(0) = C$. Then using Lemma 1 $z_k(1) - z_k(0) = sup_w\{z_k(w+1) - z_k(w)\}, \forall w \in \{\mathbb{Z}^1\}$ and hence the result.
Case II. $z_k(1) - z_k(0) < C$. Then the bottleneck shifts to cut j after addition of capacity C at cut k. Hence $z_k(w+2) - z_k(w+1) = 0;\ \forall w \in \{\mathbb{Z}^1\}$. The result then follows since $z_k(1) - z_k(0) = sup_w\{z_k(w+1) - z_k(w)\},\ \forall w \in \{\mathbb{Z}^1\}$. □

Corollary 3 ASPIRA = MUSP *for single model assembly lines and 1-commodity Continuous Flow Lines.*

Proof: Follows from Theorem 10 and the fact that the 'max-flow min-cut' theorem holds for both environments. □

8 Conclusions

In this paper we have highlighted the existence of several bottleneck definitions in operations management literature. We have provided a classification of these bottleneck definitions.

Many of these definitions do not conform to the idea of bottleneck as a constraining resource. We have proposed an universal bottleneck definition in conformance with the concept of shadow price of a resource. We have identified general conditions under which the average and marginal shadow prices would be equivalent. The equivalence of the two shadow prices has been established for four specific environments - project crashing under convex cost-time trade-offs, the $1||C_{max}$ problem, the single model assembly line and the 1-commodity Continuous Flow Line.

Further research needs to directed towards investigating the equivalence of average and marginal shadow prices for diverse production/service environments. For the job shop environment, we need more research on the perturbations of classical scheduling problems where a bank of identical machines are available in parallel. For production/service environments which can be represented as a network flow problem, we need to identify situations when the minimum cut defines the bottleneck.

Finally, a valid bottleneck definition provides the foundation on which a rich set of resource acquisition problems can be formulated. The identification of bottleneck is the first step towards the 'de-bottlenecking' process. The aim is to change the availability of resources so as to meet management objectives. Which resource to choose and by how much its availability needs to be changed are important managerial decisions. This paper provides the groundwork on which an universally valid bottleneck focused approach can be developed.

References

The American Production and Inventory Controls Society. 1995. *APICS Dictionary.* 8th edn. The American Production and Inventory Controls Society, Falls Church, VA.

Adams, J., E. Balas, D. Zawack. 1988. The Shifting Bottleneck Procedure for Job Shop Scheduling. *Management Sci.***34** 391–401.

Ahn, T., S. S. Erenguc. 1998. The resource constrained project scheduling problem with multiple crashable modes: A heuristic procedure. *Eur. J. Oper. Res.***107** 250–259.

Alcaly, R. E., A. K. Klevorick. 1966. A note on the dual prices of integer programs. *Econometrica***34** 206-214.

Billington, P. J., J. O. McClain, L. J. Thomas. 1986. Heuristics for multilevel lot-sizing with a bottleneck. *Management Sci.***32** 989–1006.

Böttcher, J., A. Drexl, R. Kolisch, F. Salewski. 1999. Project scheduling under partially renewable resource constraints. *Management Sci.***45** 543–559.

Carlier, J. 1987. Scheduling jobs with release dates and tails on identical machines to minimize the makespan. *Eur. J. Oper. Res.***29** 298–306.

Carlier, J., I. Rebai. 1996. Two branch and bound algorithms for the permutation flow shop. *Eur. J. Oper. Res.***90** 238–251.

Chase, R. B., F. R. Jacobs, N. J. Aquilano. 2006. *Operations Management for Competitive Advantage*, 11th edn. McGraw-Hill/Irwin, NY.

Chiang, S. -Y., C. -T. Kuo, S. M. Meerkov. 1998. Bottlenecks in markovian production lines: A systems approach. *IEEE Trans. on Robotics and Automation***14** 352–359.

Crema, A. 1995. Average shadow price in a mixed integer linear programming problem. *Eur. J. Oper. Res.***85** 625–635.

Crema, A. 1999. An algorithm to perform a complete right-hand-side parametrical analysis for a 0-1-integer linear programming problem. *Eur. J. Oper. Res.***114** 569–579.

Drexl, A., A. Kimms. 2001. Sequencing JIT mixed model assembly lines under station-load and part-usage constraints. *Management Sci.***47** 480–491.

Drobouchevitch, I. G., V. A. Strusevich. 2000. Heuristics for the two-stage job shop scheduling problem with a bottleneck machine. *Eur. J. Oper. Res.***123** 229–240.

Fisher, M. L. 1973. Optimal solution of scheduling problems using Lagrange multipliers: Part I. *Oper. Res.***21** 1114–1127.

Ford, L. R., D. R. Fulkerson. 1956. Maximal flow through a network. *Canadian J. Mathematics***8** 399–404.

Garey, M. R., D. S. Johnson. 1979. *Computers and intractability: A guide to the theory of NP-completeness.* Freeman, San Francisco, CA.

Geoffrion, A. M. 1974. Lagrangean relaxation for integer programming. *Math. Programming Study***2** 82–114.

Geoffrion, A. M., R. Nauss. 1977. Parametric and postoptimality analysis in integer linear programming. *Management Sci.***23** 453–466.

Goldratt, E. M. 1997. *Critical Chain*, North River Press, Great Barrington, MA.

Goldratt, E. M., J. Cox. 1984. *The Goal*, North River Press, Croton-on-Hudson, NY.

Goldratt, E. M., R. E. Fox. 1986. *The Race*, North River Press, Croton-on-Hudson, NY.

Gomory, R. E., W. J. Baumol. 1960. Integer programming and pricing. *Econometrica***28** 521–550.

Grosfeld-Nir, A. 2005. A two-bottleneck system with binomial yields and rigid demand. *Eur. J. Oper. Res.***165** 231–250.

Grosfeld-Nir, A., Y. Gerchak. 2002. Multistage production to order with rework capability. *Management Sci.***48** 652–664.

Hackman, S. T., R. C. Leachman. 1989. A general framework for modeling production. *Management Sci.***35** 478–495.

Herroelen, W., R. Leus. 2001. On the merits and pitfalls of critical chain scheduling. *J. Operations Mangement.***19** 559–577.

Herroelen, W., R. Leus, E. Demeulemeester. 2002. Critical chain project scheduling: Do not oversimplify. *Project Management J.***33** 48–60.

Ho, J. C., J. S. Wong. 1995. Makespan minimization for m parallel identical processors. *Naval Res. Logist.***42** 935–948.

Hu, T. C. 1969. *Integer Programming and Network Flows*, Addison-Wesley, Reading, MA.

Hurley, S. F., S. Kadipasaoglu. 1998. Wandering bottlenecks: Speculating on the true causes. *Production Inventory Management J.***39** 1–4.

Johnson, S. M. 1954. Optimal two- and three-stage production schedules with set up times included. *Naval Res. Logist. Quart.***1** 61–68.

Kantorovich, L. V. 1939. Mathematicheskie metody organizatsii i planirovania proizvodstva. *Leningrad State University Publishers* translated as: Mathematical methods in the organisation and planning of production *Management Sci.***6** 366–422.

Kim, S., S. Cho. 1988. A shadow price in integer programming for management decision. *Eur. J. Oper. Res.***37** 328–335.

Kondili, E., C. C. Pantelides, R. W. H. Sargent. 1993. A general algorithm for short-term scheduling of batch operations - I. MILP formulation. *Comput. Chem. Engrg.***17** 211–227.

Koopmans, T. C. 1976. Concepts of optimality and their uses. *Math. Programming***11** 212–228.

Lageweg, B., J. K. Lenstra, A. H. G. Rinnooy Kan. 1977. Job shop scheduling by implicit enumeration. *Management Sci.***24** 441–450.

Lawrence, S. R., A. H. Buss. 1994. Shifting production bottlenecks: Causes, cures, and conundrums. *Production Oper. Management***3** 21–37.

Leighton, T., S. Rao. 1999. Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. *J. ACM***46** 787–832.

Linn, R. W. Zhang. 1999. Hybrid flow shop scheduling: A survey. *Comp. Indust. Engg.***37** 57–61.

McKay, K. N., T. E. Morton. 1998. Critical Chain. *IIE Transactions***30** 759–762.

Mokotoff E. 2001. Parallel machine scheduling problems: A survey. *Asia-Pacific J. Oper. Res.***18** 193–242.

Morton, T. E., D. W. Pentico. 1993. *Heuristic scheduling systems: with applications to production systems and project management*, John Wiley & Sons, NY.

Moss, H. K., W. B. Yu. 1999. Toward the estimation of bottleneck shiftiness in a manufacturing operation. *Production Inventory Management J.***40** 53–58.

Mukherjee, S., A. K. Chatterjee. 2006. The Average Shadow Price for MILPs with integral resource availability and its relationship to the Marginal Unit Shadow Price. *Eur. J. Oper. Res.***169** 53–64

Raz, T., R. Barnes, D. Dvir. 2003. A critical look at critical chain project management. *Project Management J.***34** 24–32.

Salveson, M. E. 1955. The assembly line balancing problem. *J. Industrial Engg.***6** 18–25.

Schrijver, A. 1990. Homotopic routing methods. Korte, B., L. Lovász, H. J. Prömel, A. Schrijver (eds). *Paths, Flows, and VLSI-Layout.* Springer-Verlag, Berlin.

Shapiro, R. D. 2002. National Cranberry Cooperative (Abridged). Harvard Business School Case 9-688-122, Harvard University, Cambridge, MA.

Sousa, J. P., L. A. Wolsey. 1992. A time indexed formulation of non-preemptive single machine scheduling problems. *Math. Programming***54** 353–367.

Strusevich, V. A., L. A. Hall. 1997. An open shop scheduling problem with a non-bottleneck machine. *Oper. Res. Lett.***21** 11–18.

Sule, D. R., K. Vijayasundaram. 1998. A heuristic procedure for makespan minimisation in job shops with multiple identical processors. *Comp. Indust. Engg.***35** 399–402.

Talbot, F. B. 1982. Resource-constrained project scheduling with time-resource tradeoffs: The nonpreemptive case. *Management Sci.***28** 1197–1210.

Trietsch, D. 2005a. Why a critical path by any other name would smell less sweet? Towards a holistic approach to PERT/CPM. *Project Management J.***36** 27–36.

Trietsch, D. 2005b. From management by constraints (MBC) to management by criticalities (MBC II). *Human Systems Management.***24** 105–115.

Tucker, A. W. 1956. Dual systems of homogenous linear relations. Kuhn, H. W., A. W. Tucker. *Linear Inequalities and Related Systems.* Princeton University Press, Princeton.

Uğurdağ, H. F., R. Rachamadugu, C. A. Papachristou. 1997. Designing paced assembly lines with fixed number of stations. *Eur. J. Oper. Res.***102** 488–501.

Uzsoy, R., C. Wang. 2000. Performance of decomposition procedures for job shop scheduling problems with bottleneck machines. *Internat. J. Production Res.***38** 1271–1286.

Vairaktarakis, G. L. 2003. The value of resource flexibility in the resource-constrained job assignment problem. *Management Sci.***49** 718-732.

Van den Akker, J. M., C. A. J. Hurkens, M. W. P. Savelsbergh. 2000. Time-indexed formulations for machine scheduling problems: Column generation. *INFORMS J. Comput.***12** 111-124.

Wiest, J. D., F. K. Levy. 1977. *A Management guide to PERT/CPM: with GERT/PDM/DCPM and other networks.*, 2nd edn. Prentice-Hall, Englewood Cliffs, NJ.

Wu, K. 2005. An examination of variability and its basic properties for a factory. *IEEE Trans. Semiconductor Manufacturing***18** 214–221.