Node Generation and Capacity Reallocation in Open Jackson Networks

Imry Rosenbaum [†], Irad Ben-Gal [†] and Uri Yechiali [†, ☼]

† Tel-Aviv University (imryro@yahoo.com, bengal@eng.tau.ac.il, uriy@post.tau.ac.il)

☼ Afeka College of Engineering (uriy@afeka.ac.il)

Abstract

We investigate methods for reallocation of service capacities in open Jackson networks in order to minimize system's mean total work-in-process (WIP), or its response time. We focus mainly on a method called *node generation*, by which capacity can be transferred from a node in echelon *j* to a *newly generated* node in echelon *j*+1. We compare this new procedure with the more conventional *capacity redistribution* method, by which capacity can be transferred from any node in echelon *j* to existing successor nodes in echelon *j*+1. Formulation of each method as a mathematical programming problem reveals the structure of the optimal solution for both problems. The motivation for considering these approaches stems from real-life settings. In particular, from a production line or supply chains where the two types of capacity reallocation are applied. We develop heuristic methods to solve relatively large networks in tractable time. Numerical results and analyses are presented.

1. Introduction & Literature Review

In this work we analyze the mean total work-in-process (WIP) of an open network of queues. We consider hierarchical Jackson networks without feedbacks, where each node in the network is modeled as a single server $M/M/1$ queue that routes its incoming flow to lower level nodes. That is, we assume that jobs flow into the network following a Poisson process, and service times in each node originate from an exponential distribution (specific for each node). Our objective is to minimize the mean total work-in-process (WIP) in the system (or equivalently to minimize the system's response time) - a problem known as a complex one (Bretthauer, 2000, Jane and Srinivasa Raghavan 2009). This goal is achieved through a capacity reallocation approach within the network, either by transferring capacity to *existing nodes* in downstream levels (called *capacity redistribution*) or by generating *new nodes* in downstream levels and transferring capacity to them (we call this method *node generation*). Most of this paper deals with the latter *node generation* method that has received little attention in the literature.

The motivation for considering the above-mentioned approaches stems from many real-life settings. The original capacity reallocation problem can be found in the classic communication network literature. For example, when designing grid networks, servers or CPUs can be moved to existing or to newly generated nodes in order to minimize the system response time. Similar applications were related to manufacturing systems. One can envision a manufacturing system where the two types of capacity reallocation can be applied. Consider a factory with hierarchical tree-like production stations in which work is performed task by task with stations dedicated to specific tasks in the process. Some stations might have extra workers or machines (e.g., extra capacity) that can be moved either to existing downstream stations (i.e., *capacity redistribution*), or to empty downstream areas to generate new stations (i.e., *node generation*). Later examples can be taken from supply chain networks where a fixed budget can be allocated either to increase the distribution and storage capacity of existing distributors (e.g., by adding vehicles or warehouse space) or to add a new distribution subcontractor to the chain. Below we provide references to the above mentioned areas, although the paper focuses on the theoretical concepts of capacity reallocation without relating to a specific application. Note that regardless of the specific network application, it is well known (e.g., Bitran and Triputi, 1989A), and further confirmed in this paper, that capacity reallocation can vastly improve the performance of a queueing network.

There is a large body of literature that addresses optimization problems related to optimal service and arrival rates in queueing models. For the class of queueing-network models which is mentioned here, the service rates at the stations typically represent capacity variables, while the arrival rates typically represent the flow of jobs/units of product, or rate of demand.

The classic network-design literature is associated with communication networks and considers various optimization problems, often for a given networks' topology. Kleinrock (1964) modeled packet transmission as a Jackson network and used the model to optimize communication networks performance. Kleinrock and Gerla (1977) considered a design problem in communication networks. They looked for a minimum network cost associated with line capacities and flow assignments, such that an upper limit on transmission delay is satisfied. Ng and Hoang (1987) determined the optimal capacity and flow assignments in a special communication network that lead to a convex optimization problem. Kleinrock et al. (1973)

modeled a computer network as a Jackson network of queues and proposed several approaches for determining capacities and flows, leading to locally optimal solutions. Kleinrock (1976) provided further references and ideas for the use of queueing theory to model computer communication networks, and Crabill et al. (1977) provided further references on related optimization problems of queueing networks. Bretthauer (2000) used a branch and bound algorithm to find optimal service and arrival rates in Jackson queueing networks. Most of the above network-design papers assume that the network topology is fixed, whereas the approach in this paper supports modifications of the network structure.

The use of queueing networks for modeling manufacturing systems became popular in the mid 80's. A typical network-design problem of a manufacturing system is to select service rates that represent workstation capacities such that profits are maximized, while other queueing measures (such as an upper limit on work-in-process inventory) are satisfied. Buzacott and Yao (1986) reviewed the developments of closed queueing network (CQN) models and classified various modeling approaches. Yao and Shanthikumar (1987) presented a method for determining the optimal arrival rates into a manufacturing system modeled as a network of queues. Bitran and Tirupati (1989B) and Bretthauer and Shetty (1995) considered the problem of selecting service rates (capacity) from continuous and discrete sets of choices for a manufacturing queueing network. Buss et al. (1994) treated both the arrival and service rates as decision variables in a single $M/M/1$ station in a manufacturing system, but did not address the case of network of queues. Dewan and Mendelson (1990) and Stidham (1992) considered service facilities modeled as a single station queue where the arrival and service rates are defined as decision variables, but again, they did not analyze the case of a network of queues. Suri et al. (1993) examined performance evaluation models for different manufacturing systems, such as single stage systems (single queues), production lines (tandem queues), assembly lines (arborescent queues), job shops (open queueing network) and flexible manufacturing systems (FMS). Buzacott and Shanthikumar (1992, 1993), Hsu et al. (1993), Kouvelis and Tirupati (1991), Bitran and Dasu (1992) and Bitran and Sarkar (1994) considered performance evaluation models and analyzed optimization models for queueing networks. Buzacott and Shanthikumar presented an extensive analysis oriented toward the design of different manufacturing systems such as flow lines, automated transfer lines, job shops, FMSs, and multi cellular systems. They analyzed optimal design problems and, in particular, considered some optimization models in job shops, such as optimal allocation of workers to stations, optimal number of operators in the system, optimal allocation of jobs to stations and analysis of routing and time diversity effects in job processing.

Hsu et al. (1993) examined optimization models for FMS based on CQN; they also suggested the use of alternative techniques like algebra max-plus, fuzzy sets and expert systems. For further information on the use of queueing theory to analyze and design flexible manufacturing systems, see Gershwin (1994). Queueing models have been also used to analyze service systems. Bretthauer and Cote (1998) presented a method for determining discrete service rates in a health care queueing network. Modern call centers are often analyzed by queueing measures, such as the time customers are waiting for service (e.g., Mandelbaum et. al (2003) and Armony (2005)). The *capacity reallocation* in general and the *node generation* in particular are relevant to this literature branch. Note that most of the above references focus on selecting proper service and arrival rates of the system, unlike the considered problem in this paper that focuses on capacity reallocation in the system. Moreover, to the best of our knowledge, the proposed node generation approach has not been suggested in the literature, and expands the applicability of the approaches mentioned above.

Supply chain networks have been modeled by queueing networks in recent years (Li and Ying 2009, Ha (1997) and Suri, 1998). Reallocation of capacity in such networks can be associated with budget distribution among various service providers in the chain, such as raw material vendors, original equipment manufacturers (OEMs), logistics operators, warehouse operators, distributors and retailers. For example, Srinivasa Raghavan and Viswanadham (2001), Dong and Chen (2005), Jane and Srinivasa Raghavan (2009) and Bhaskar and Lallement (2010) presented analytical models for evaluating the average response time of supply chains by using various queueing networks models. Jane and Srinivasa Raghavan (2009) determined the optimal inventory level in a warehouse chain that minimizes total expected cost of carrying inventory, back order cost associated with serving orders in the backlog queue, and ordering cost. They extended their model to a three-echelon inventory model which explicitly considers the involved logistics processes. Bhaskar and Lallement (2011) considered a queueing network model of uniformly distributed arrivals in a distributed supply chain using subcontractors. Their objective was to compute the minimum response time, and the average number of items (optimum capacity) that can be delivered with this response time. Note that when considering the overall supply chain performance, it is often unclear whether to increase the budget of an existing service provider or to outsource some services to a new subcontractor. The proposed node generation approach aims to model such a question under specific assumptions.

The rest of the paper is organized as follows. Section 2 presents the node generation approach for a 2-level (n+1)-node network and compares it to the more-conventional *capacity redistribution* scheme. Section 3 expands the node generation method to multi-level networks and lists the necessary conditions for optimality. These conditions are then used to propose a heuristic for larger networks that are studied next. In Section 4, a numerical study of the proposed heuristic is presented along with sensitivity analysis of some network parameters. Section 5 summarizes the paper and suggests a few future research directions.

2. Capacity Reallocation in Small Jackson Networks

This chapter provides notation for a general network topology. It then explores the *node generation* procedure and the structure of its optimal solution for a 2-level (n+1)-node network, and compares it to the *capacity redistribution* solution.

2.1 Notation and Definitions

We start by describing the general structure of the networks we are using throughout this paper. The networks considered are multi-level sequential Jackson-type networks comprised of M/M/1 queues. Each node feeds its immediate successors and there are no feedbacks. Examples are given in Figure 1a for a 2-level network and in Figure 3 for an M-level network. Denote the external Poisson arrival rate to the network by λ. Let μ_i^j be the capacity (service rate) of node *i* in level *j*, that is, the service time of each job in node *i* in level *j* is exponentially distributed with mean $1/\mu_i^j$. Let λ_i^j be the Poisson arrival rate to this node. Such a node can be defined as a $M\left(\lambda_i^j\right)/M\left(\mu_i^j\right)/1$ queue. In this work the value of a term *Z* raised to the *R*-th power will be denoted as $\left(Z\right)^R$, instead of the usual notation Z^R.

It is well known (Kleinrock, 1975) that the mean number of jobs in such a queue is given by $E[L_i^j]=\dfrac{\lambda_i^j}{\mu_i^j-\lambda_i^j}$ (provided that $\mu_i^j>\lambda_i^j$). It is clear that, for each level *j*, $\sum_i \lambda_i^j=\lambda$.

The *mean work in process* (WIP) in the entire system is given by

$$E[L]=\sum_{i,j}E[L_i^j] \tag{1}$$

As stated above, the paper focuses on studying a node generation approach for minimizing the total WIP in the network, while comparing it with the more commonly-used capacity redistribution approach. Under node generation, capacity can be transferred from a node in echelon (level) j to a new generated node in echelon j+1. In each echelon we limit the node generation process to the next level only. We do so to limit the complexity of the proposed approach, since the alternative requires extensive real-time knowledge about flow states in the entire network that is often unknown in real life scenarios. Focusing on local decision making is usually an acceptable paradigm in the design and control of supply chains (see Lee and Whang, 1999). Moreover, we do not address dynamic scheduling issues that might exist in real life settings, possibly modeled as multi-server queues, but rather focus on the average long-lasting performance of the system. Relying on such an approximation enables us to model the system by a network of M/M/1 queues, as commonly done in relevant network literature.

2.2 Node Generation in a 2-Level (n+1)-Node Network

Consider a two-level, (n+1)-node network as depicted in Figure 1(a) and let capacity x_1^1 be transferred from the origin node to generate a new node in level 2. The result is depicted in Figure 1(b). Define p_{1k}^1 as the fraction of jobs directed from node 1 in level 1 to node k in level 2 $k=1,2,....,n$. The fraction of jobs directed to the new node is $\left(1-\sum_{k=1}^{n} p_{1k}^1\right)$. Thus, $\lambda_k^2 = \lambda p_{1k}^1$ for $k=1,2,....,n,$ and the arrival rate to the new generated node is $\lambda\left(1-\sum_{k=1}^{n} p_{1k}^1\right)$, which is denoted for simplicity in the general case by $\tilde{\lambda}$. Clearly, we must obtain $\mu_k^2 > \lambda_k^2$ for all k=1,2,…, $n,$ and $x_1^1 > \lambda\left(1-\sum_{k=1}^{n} p_{1k}^1\right)$.

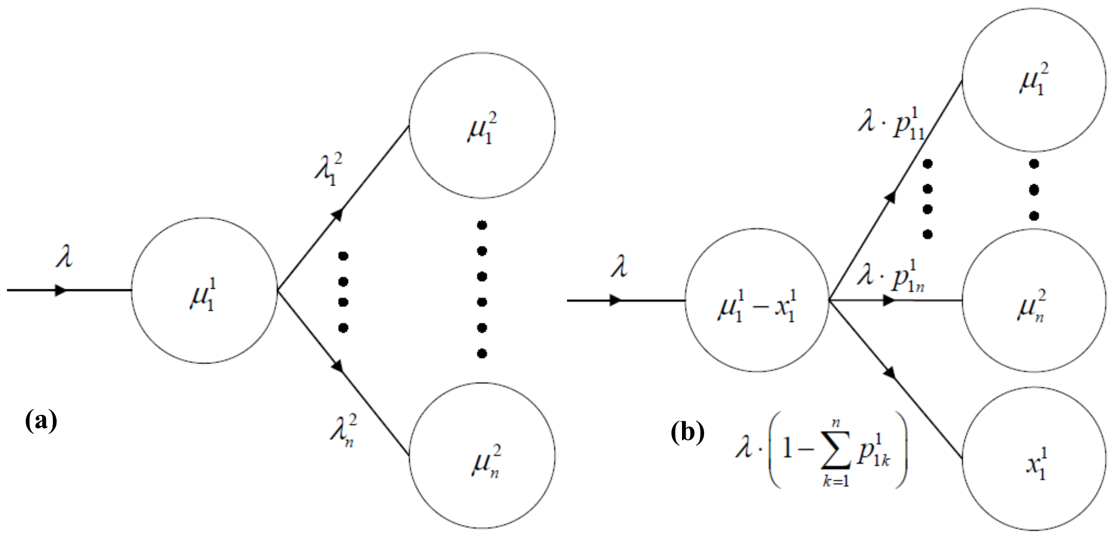

Figure 1: (a): The initial network.

(b): The modified network after applying *node generation.*

Mathematical Formulation

Our objective is to minimize the systems WIP under the *node generation* procedure. This objective can be expressed by the following mathematical program:

$$Min \left[\frac{\lambda}{\mu_1^1 - x_1^1 - \lambda} + \sum_{k=1}^{n} \frac{\lambda p_{1k}^1}{\mu_k^2 - \lambda p_{1k}^1} + \frac{\lambda\left(1 - \sum_{k=1}^{n} p_{1k}^1\right)}{x_1^1 - \lambda\left(1 - \sum_{k=1}^{n} p_{1k}^1\right)} \right] \tag{2}$$

Such that, for $k = 1, 2, ..., n,$

$$x_1^1 \geq 0,\ p_{1k}^1 \geq 0\ , \sum_{k=1}^{n} p_{1k}^1 \leq 1, \tag{3}$$

$$\mu_1^1 - x_1^1 > \lambda,\ \mu_k^2 > \lambda p_{1k}^1$$

$$x_1^1 > \lambda\left(1 - \sum_{k=1}^{n} p_{1k}^1\right)$$

Solution

Differentiating the objective function (2) with respect to p_{1k}^1 k=1,2,…,n, as well as with respect to x_1^1, and then equating each derivative to zero, lead to the following optimality conditions:

$$\frac{\mu_i^2}{\left(\mu_i^2-\lambda p_{1i}^1\right)^2}=\frac{\mu_j^2}{\left(\mu_j^2-\lambda p_{1j}^1\right)^2}\quad \forall i,j$$

$$\frac{\mu_1^2}{\left(\mu_1^2-\lambda p_{11}^1\right)^2}=\frac{x_1^1}{\left(x_1^1-\lambda\left(1-\sum_{k=1}^{n}p_{1k}^1\right)\right)^2} \tag{4}$$

$$\frac{\lambda}{\left(\mu_1^1-x_1^1-\lambda\right)^2}=\frac{\lambda\left(1-\sum_{k=1}^{n}p_{1k}^1\right)}{\left(x_1^1-\lambda\left(1-\sum_{k=1}^{n}p_{1k}^1\right)\right)^2}$$

Equations (4) do not yield a closed form solution. However, rewriting these equations in terms of 'excess capacity' can illuminate the structure of the optimal solution. Define the 'excess capacity' of nodes as follows

$$\Delta_1^1=\mu_1^1-x_1^1-\lambda;\;\; \Delta_i^2=\mu_i^2-\lambda p_{1i}^1 \quad (i=1,2,..,n),$$

$$\Delta_{x_1^1}^2=x_1^1-\lambda\left(1-\sum_{i=1}^{n}p_{1i}^1\right). \tag{5}$$

Then, (4) can be written as

$$\frac{\mu_i^2}{\left(\Delta_i^2\right)^2}=\frac{x_1^1}{\left(\Delta_{x_1^1}^2\right)^2}\quad \forall i \tag{5}$$

$$\frac{\lambda}{\left(\Delta_1^1\right)^2}=\frac{\lambda\left(1-\sum_{k=1}^{n}p_{1k}^1\right)}{\left(\Delta_{x_1^1}^2\right)^2}$$

That is, in the case where $x_1^1 > 0$, the necessary conditions imply: (i) balance between the ratios [capacity/(excess capacity)^2] for all nodes in level 2, including the newly generated node, and (ii) equality of [inflow/(excess capacity)^2] between the origin node (from which capacity x_1^1 is extracted) and the newly generated node. This solution structure does not have closed form even for *n*=1, thus it is difficult to create simple rule to when to use node generation and when not to.

The above analysis can be extended by taking into account various costs parameters that are often used in supply chain literature (Jain and Srinivasa Raghavan, 2009). In particular, denote the holding cost rate of one unit of WIP in node *i* in level *j* by c_i^j and the transfer cost of moving one unit of capacity to the newly generated node by *d*. Then, the corresponding mathematical program to (2) is the following:

$$Min\left[\frac{c_1^1\lambda}{\mu_1^1 - x_1^1 - \lambda} + \sum_{k=1}^{n}\frac{c_k^2\lambda p_{1k}^1}{\mu_k^2 - \lambda p_{1k}^1} + \frac{c_{x_1^1}^2\lambda\left(1-\sum_{k=1}^{n}p_{1k}^1\right)}{x_1^1 - \lambda\left(1-\sum_{k=1}^{n}p_{1k}^1\right)} + dx_1^1\right] \tag{6}$$

such that, for $k = 1,2,...,n,$ all the original constraints hold. The solution procedure to (6) follows exactly the solution procedure to (2). The obtained result in terms of 'excess capacity' is

$$\frac{c_i^2\mu_i^2}{\left(\Delta_i^2\right)^2} = \frac{c_{x_1^1}^2 x_1^1}{\left(\Delta_{x_1^1}^2\right)^2} \quad \forall i \tag{7}$$

$$\frac{c_1^1\lambda}{\left(\Delta_1^1\right)^2} + d = \frac{c_{x_1^1}^2\lambda\left(1-\sum_{k=1}^{n}p_{1k}^1\right)}{\left(\Delta_{x_1^1}^2\right)^2}$$

Thus, one can see that adding costs to the node generation problem has no structural effect on the optimal solution. Yet, the generalized solution takes into account both holding and transfer costs in a manner that can affect the value of the transferred capacity.

Another relevant observation is that, if the excess capacity in the parent node is allocated to the descendent node, the proposed node generation scheme is convex. Let us investigate the convexity of the *node generation* scheme by using a superposition of two networks: Network A is composed of the n original nodes 1,…,n in the second level, whose WIP is dictated by the flow the generated node receives ($\tilde{\lambda}$), while network B is composed of the node in the first level and the generated node, whose combined WIP is a function of $\tilde{\lambda}$ and x_1^1. Note that the WIP function of a single node is convex with respect to its incoming flow, as the second derivative is positive. Thus, WIP of network A is a convex function of λ, being the sum of convex functions.

The Hessian of the WIP of network B is derived as follows:

$$\begin{pmatrix} \frac{2\left(x_1^1\right)}{\left(x_1^1-\tilde{\lambda}\right)^3} & \frac{-\left(x_1^1+\tilde{\lambda}\right)}{\left(x_1^1-\tilde{\lambda}\right)^3} \\ \frac{-\left(x_1^1+\tilde{\lambda}\right)}{\left(x_1^1-\tilde{\lambda}\right)^3} & \frac{2(\lambda)}{\left(\mu_1^1-x_1^1-\lambda\right)^3}+\frac{2\tilde{\lambda}}{\left(x_1^1-\tilde{\lambda}\right)^3} \end{pmatrix}$$

For this function to be convex, the Hessian has to be positive, i.e., all of its principal minors have to be non-negative. The first minor is clearly non-negative in the feasible region, while the second minor equals $\frac{4x_1^1\lambda}{\left(\mu_1^1-x_1^1-\lambda\right)^3}-\frac{1}{\left(x_1^1-\tilde{\lambda}\right)}$. Note that for $x_1^1 \to \mu_1^1-\lambda$, the condition for non-negativity simplifies to $4\lambda\left(\lambda-\mu_1^1\right)\left(\tilde{\lambda}-x_1^1\right)\geq 0$ which is clearly satisfied, as the capacity in each node is not smaller from its inflow. Thus, there is a feasible global solution to the problem if the transferred capacity approaches the excess capacity in the parent node.

2.3 Redistribution in a 2-Level (n+1)-Node Network

In this section we compare the node generation procedure with the capacity redistribution scheme. Consider now the same 2-level (n+1)-node basic network depicted in Figure 1(a) and copied in Figure 2(a). The redistribution procedure is shown in Figure 2(b), where y_{1k}^1 $(k=1,2,....n)$ is the capacity transferred from node 1 in level 1 to node k in level 2. The remaining capacity of node 1 in level 1 is $\mu_1^1-\sum_{k=1}^{n} y_{1k}^1$. Again, $\lambda_k^2=\lambda p_{1k}^1$, where p_{1k}^1 is the fraction of arrivals directed from the origin node to node k in level 2. Clearly, $\lambda=\sum_{k=1}^{n}\lambda_k^2$.

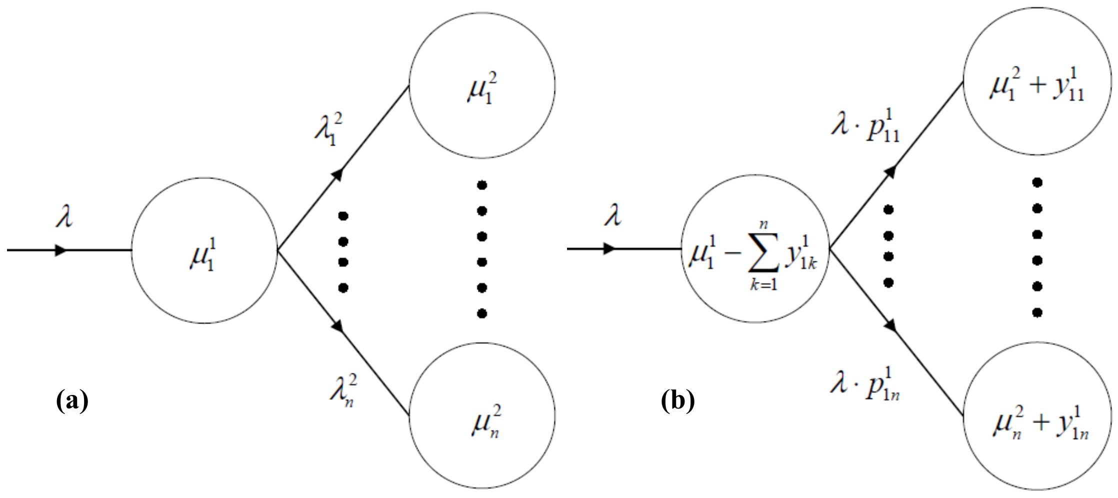

Figure 2: (a): The initial 2-level (n+1)-node network.
(b): The modified network after applying *redistribution.*

Mathematical Formulation

As stated, the objective is to *redistribute* capacities so as to minimize WIP in the system. This objective can be expressed by the following mathematical program:

$$Min\left[\frac{\lambda}{(\mu_1^1 - \sum_{k=1}^{n} y_{1k}^1) - \lambda} + \sum_{k=1}^{n} \frac{\lambda p_{1k}^1}{\left(\mu_k^2 + y_{1k}^1\right) - \lambda p_{1k}^1}\right] \tag{8}$$

Such that, for $k = 1, 2, ..., n,$

$$y_{1k}^1 \geq 0,\ p_{1k}^1 \geq 0,\ \sum_{k=1}^{n} p_{1k}^1 = 1,\ \ \sum_{k=1}^{n} y_k^1 < \mu_1^1,$$

$$\mu_1^1 - \sum_{k=1}^{n} y_{1k}^1 > \lambda,\ \mu_k^2 + y_{1k}^1 > \lambda p_{1k}^1$$

Solution

In Appendix A it is shown that the structure of the optimal solution of the above mathematical program implies that if (some) capacity is transferred from the origin node, it is transferred to the node in level 2 having the largest initial capacity. That is, assuming without loss of generality that $\mu_1^2 = \max\limits_{1\le i\le n}\left\{\mu_i^2\right\}$, the optimal solution satisfies $y_{1k}^1 = 0$ for $k \ge 2$, so that the optimal values of y_{11}^1 and $\left\{p_{1k}^1\right\}$ satisfy the optimality conditions (9) and (10) below:

$$\frac{\lambda}{\left(\mu_1^1 - y_{11}^1 - \lambda\right)^2} = \frac{\lambda p_{11}^1}{\left(\mu_1^2 + y_{11}^1 - \lambda p_{11}^1\right)^2} \tag{9}$$

This follows by differentiating the corresponding Lagrangian with respect to y_{11}^1, and setting $y_{1k}^1 = 0$ for $k \ge 2$. Similarly, by differentiating the corresponding Lagrangian with respect to p_{1k}^1, and setting $y_{1k}^1 = 0$ for $k \ge 2$, one obtains:

$$\frac{\mu_k^2}{\left(\mu_k^2 - \lambda p_{1k}^1\right)^2} = \frac{\mu_1^2 + y_{11}^1}{\left(\mu_1^2 + y_{11}^1 - \lambda p_{11}^1\right)^2} \quad k = 2,\dots,n \tag{10}$$

Appendix A demonstrates and proofs the above claimed general structure of the solution for an (*n*+1)-node network. It is easy to see that the optimal solution for *n*=1 is to equate the capacity of both levels. For the case *n*=2, as depicted in Figure 2(b), and under the assumption that both $\lambda_1^2 > 0$ and $\lambda_2^2 > 0$, the optimal value of the total WIP in Figure 2(a) is

$$\mathrm{WIP} = \frac{\lambda}{\mu_1^1 - \lambda} + \frac{\lambda_1^2}{\mu_1^2 - \lambda_1^2} + \frac{\lambda - \lambda_1^2}{\mu_2^2 - \left(\lambda - \lambda_1^2\right)} \tag{11}$$

Substituting the optimal value of $\lambda_1^2 = \lambda p_{11}^1$ from (9) (with $0 < \lambda_1^2 < \lambda$) results in

$$\mathrm{WIP} = \frac{\lambda}{\mu_1^1 - \lambda} + \frac{2\lambda - \left(\sqrt{\mu_1^2} - \sqrt{\mu_2^2}\right)^2}{\mu_1^2 + \mu_2^2 - \lambda} \tag{12}$$

Now, if the optimal allocation is to transfer y capacity units from the (single) node in level 1 to the nodes in level 2 such that αy units $\left(0 \le \alpha < 1\right)$ are allocated to node 1 in level 2 and $\left(1-\alpha\right)y$ units to node 2 in level 2, then the value of the work-in-process according to (12) will be

$$WIP_\alpha = \frac{\lambda_1^1}{\mu_1^1 - y - \lambda_1^1} + \frac{2\lambda - \left(\sqrt{\mu_1^2 + \alpha y} - \sqrt{\mu_2^2 + \left(1-\alpha\right)y}\right)^2}{\mu_1^2 + \mu_2^2 + y - \lambda} \tag{13}$$

This is obtained from equation (12) by replacing μ_1^1 by $\left(\mu_1^1 - y\right)$, μ_1^2 by $\left(\mu_1^2 + \alpha y\right)$ and μ_2^2 by $\left(\mu_2^2 + \left(1-\alpha\right) y\right)$ following the policy of capacity redistribution.

The other considered alternative is to allocate all y units only to the node with the initial largest capacity. Without loss of generality, we assume that it is node 1 and denote the resulting WIP in the network with WIP_1 (*i.e.* α=1). Calculating the difference in *WIP* values between the two alternatives yields

$$WIP_1 - WIP_\alpha = \frac{2\lambda - \left(\sqrt{\mu_1^2 + y} - \sqrt{\mu_2^2}\right)^2 - \left(2\lambda - \left(\sqrt{\mu_2^2 + \left(1-a\right) y} - \sqrt{\mu_1^2 + \alpha y}\right)^2\right)}{\mu_1^2 + \mu_2^2 + y - \lambda} \tag{14}$$

$$= \frac{2\sqrt{\mu_1^2 + y}\sqrt{\mu_2^2} - 2\sqrt{\mu_2^2 + \left(1-a\right) y}\sqrt{\mu_1^2 + \alpha y}}{\mu_1^2 + \mu_2^2 + y - \lambda}$$

It can readily be seen that if $\mu_1^2 > \mu_2^2$ then $WIP_1 \le WIP_\alpha$. Accordingly, in order to minimize the total WIP in the network, we allocate additional capacity only to the node with the largest capacity.

Remark: It can be intuitively explained that optimal redistribution outperforms optimal node generation. As pointed out by an anonymous referee, the node generation approach can be represented as redistribution of capacity to a node with a zero capacity. Such redistribution is clearly suboptimal, as the optimal approach requires transferring capacity to the node with the largest initial capacity, as indicated in Appendix A. In the following sections, however, we focus on the node generation approach. The reason is twofold. First, there are situations where capacity redistribution is infeasible in real settings, as discussed below, and second, to the best of our knowledge, the node generation approach has not been addressed before in the queueing networks literature.

Finally, let us note that the above observation is consistent with known results in queueing theory (e.g., Yechiali (1977)), where

$$E\left[L\left(M(c\cdot\lambda)/M(c\cdot\mu)/1\right)\right] < E\left[L\left(M(c\cdot\lambda)/M(\mu)/c\right)\right] < c\cdot E\left[L\left(M(\lambda)/M(\mu)/1\right)\right].$$

That is, in terms of reducing mean queue size, a single-server M/M/1 queue with arrival rate $c\cdot\lambda$ and service rate $c\cdot\mu$ is better than a multi-server M/M/c queue with same arrival rate and service rate μ for each of the c parallel servers. The latter configuration is better than c parallel M/M/1 queues, each with arrival rate λ and service rate μ.

3 Node Generation in an M-Level Network

In many real life situations, it is more applicable to perform reallocation of capacities via *node generation* than via *redistribution.* For example, consider again a service system composed of a transfer-line of stations. Some stations might have extra workers that may be better utilized by moving them to empty areas as a new station dedicated to a service stage in a lower level, or by using the budget to hire new subcontractors in a lower stage. The goal remains to minimize the system WIP or response time.

3.1 Mathematical Formulation

We consider now a sequential M-level tree-type network as depicted in Figure 3.

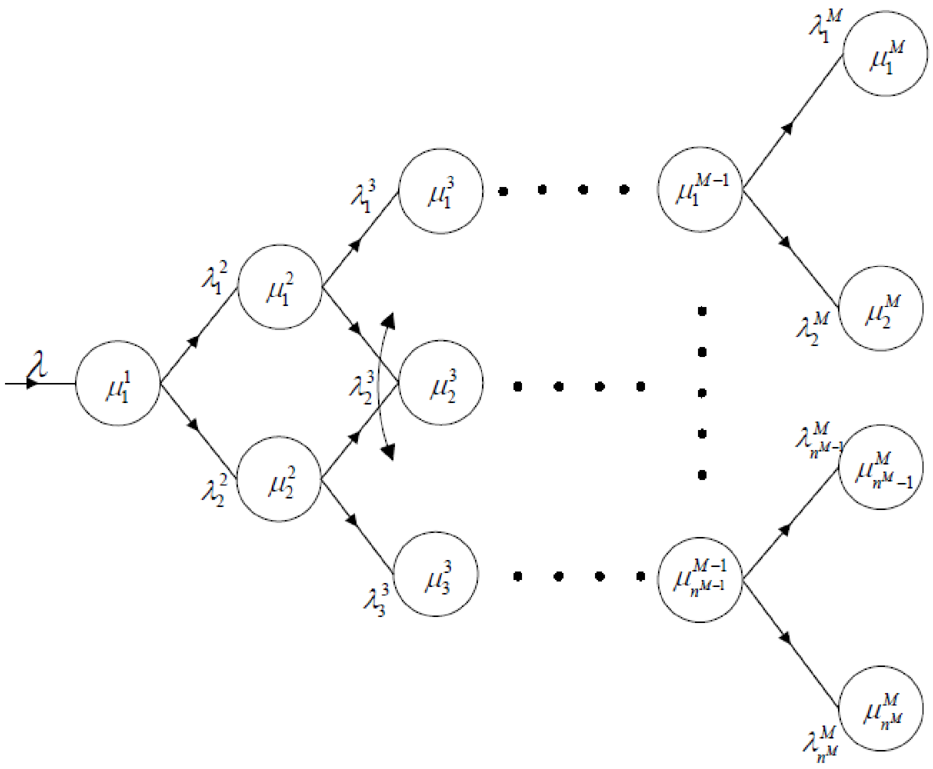

Figure 3. Sequential M-level open Jackson network

In order to formulate the problem as a mathematical programming problem we define the following terms:

A_i^j - the index set of all nodes in level j-1 that are connected to node i in level j (including the newly generated nodes in level j-1).

S_i^j - the set of all successor nodes (down the stream) of node i in level j.

W_i^j - the set of immediate successor nodes of node i in level j , clearly, $W_i^j \subseteq S_i^j$.

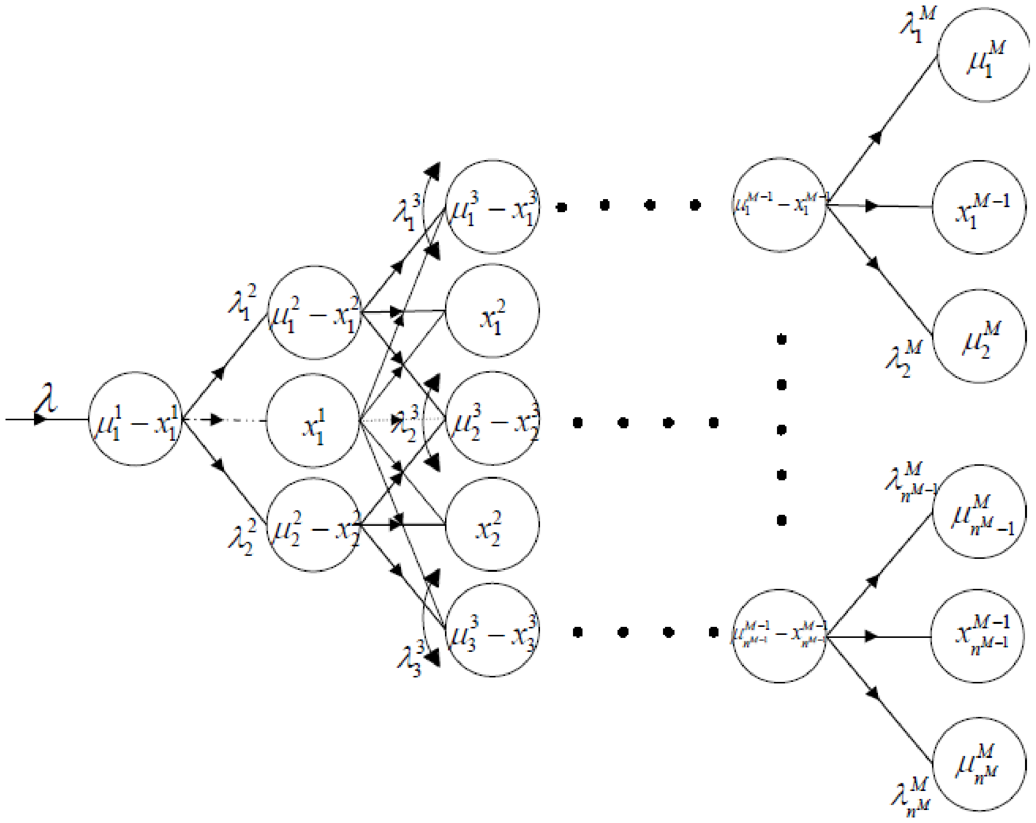

Figure 4. Sequential M-level open Jackson network with generated nodes

Note: We index a generated node that is allocated capacity x_i^j by the index x_i^j . We assume that only existing nodes can generate a new node. The new constructed network is depicted in Figure 4. The configuration is such that every original node with capacity μ_i^j may generate a new node in level j+1 contributing to it x_i^j capacity units. A new x_i^j-type generated node cannot generate further new nodes. Thus, in level j+1 there will be at most n^j new x_i^j-type nodes with positive capacity. Furthermore, the newly generated nodes may direct their job flow to anyone of their sibling node successors. We do so to avoid flow schemes that would be infeasible in the original structure. p_{ik}^j is the fraction of arrivals directed from node i in level j to node k in the next level. Our decision variables are x_i^j and p_{ik}^j. We denote

λ_i^j - the total flow rate into (and out of) node i in level j.

$\lambda_{x_i^j}^{j+1}$ - the total flow rate into (and out of) node x_i^j in level j+1.

n^j - number of initial nodes in level j.

$\tilde{\mu}_i^j$ - the capacity of node i in level j <u>after</u> applying the *node generation* approach.

The corresponding mathematical program is:

$$Min\left[\sum_{j=1}^{M}\sum_{i=1}^{n^j}\frac{\lambda_i^j}{\tilde{\mu}_i^j-\lambda_i^j}+\sum_{j=1}^{M-1}\sum_{i=1}^{n^j}\frac{\lambda_{x_i^j}^{j+1}}{x_i^j-\lambda_{x_i^j}^{j+1}}\right] \tag{15}$$

Such that

$$\tilde{\mu}_i^j=\mu_i^j-x_i^j \qquad j=1,....,M;\ i=1,......,n^j$$

$$\tilde{\mu}_{x_i^j}^{j+1}=x_i^j \qquad j=1,....,M-1;\ i=1,......,n^j$$

$$\tilde{\mu}_i^j>\lambda_i^j \quad \forall i,j$$

$$\sum_{i=1}^{n^1}\lambda_i^1=\lambda$$

$$\lambda_i^j=\sum_{a\in A_i^j}p_{a,i}^{j-1}\lambda_a^{j-1} \quad j>1;\ \text{all nodes } i \text{ in level } j \text{ (including } x_i^j\text{)}$$

$\sum_{w \in W_i^j} p_{i,w}^j = 1 \quad 1 \le j < M$; all nodes i in level j (including x_i^j)

$x_i^j, p_{ik}^j, \lambda_i^j \ge 0 \quad \forall i, j, k$

Solution

Forming the Lagrangian, substituting in the objective function $\tilde{\mu}_i^j = \mu_i^j - x_i^j$, differentiating the objective function (15) with respect to x_i^j, and equating to zero lead to the following necessary conditions:

$$\frac{\lambda_i^j}{\left(\tilde{\mu}_i^j - \lambda_i^j\right)^2} = \frac{\lambda_{x_i^j}^{j+1}}{\left(x_i^j - \lambda_{x_i^j}^{j+1}\right)^2} \quad \text{for every } 1 \le i \le n^j;\ 1 \le j \le M-1 \tag{16}$$

Similarly, differentiating the objective function with respect to p_{ik}^j, where ϕ_i^j is the Lagrange multiplier of the constraint $\sum_{w \in W_i^j} p_{i,w}^j = 1$, results in

$$\frac{\tilde{\mu}_k^{j+1}}{\left(\tilde{\mu}_k^{j+1} - \lambda_k^{j+1}\right)^2} + \sum_{q,l \in s_i^j} \frac{\tilde{\mu}_q^l \cdot \hbar_{k,q}^{j+1,l}}{\left(\tilde{\mu}_q^l - \lambda_q^l\right)^2} = \phi_i^j \qquad k \in W_i^j \tag{17}$$

where $\hbar_{i,q}^{j,l} = \sum_{a \in A_q^l \cap S_i^j} \hbar_{i,a}^{j,l-1} \cdot p_{a,q}^{l-1}$ represents the summation of all path probabilities from node i in level j to node q in level l, where $\hbar_{i,k}^{j,j+1} = p_{ik}^j \quad \forall k \in W_i^j$.

Note that the above formulation allows for several nodes in the first level.

Apparently there is no simple structural solution to the sets (16) and (17). We thus present a heuristic procedure which is based on the optimal properties of the solution of the 2-level problem in Section 2.1.

3.2 Approximating the Optimal Solution via Cuts

We propose a *cut type* procedure and exhibit its effectiveness on a sequential *M*-level tree-type network, as shown in Figure 3. We create cuts that are orthogonal to the direction of the flow, as depicted in Figure 5. A cut fixes the flow to the successor nodes of each origin node, where the total flow via the cut remains unchanged before and after the *node generation* action. Such an approach enables to ignore the possible effects of a node generation on downstream nodes and by that avoiding a complex circular formulation. The effect is neutralized by the fact that no matter which decisions are made in previous levels, the flow remains the same. Therefore, one does not have to take into consideration any changes that have been made to the incoming flow of each family of sibling nodes. That is, by fixing the flow in each cut, it is possible to optimize capacity reallocation (via the node generation approach) in each of the cuts independently. The 'greediness' impact of such a heuristic can be then handled by iteratively repeating this procedure, which can lead to a near-optimal solution, as will be seen below.

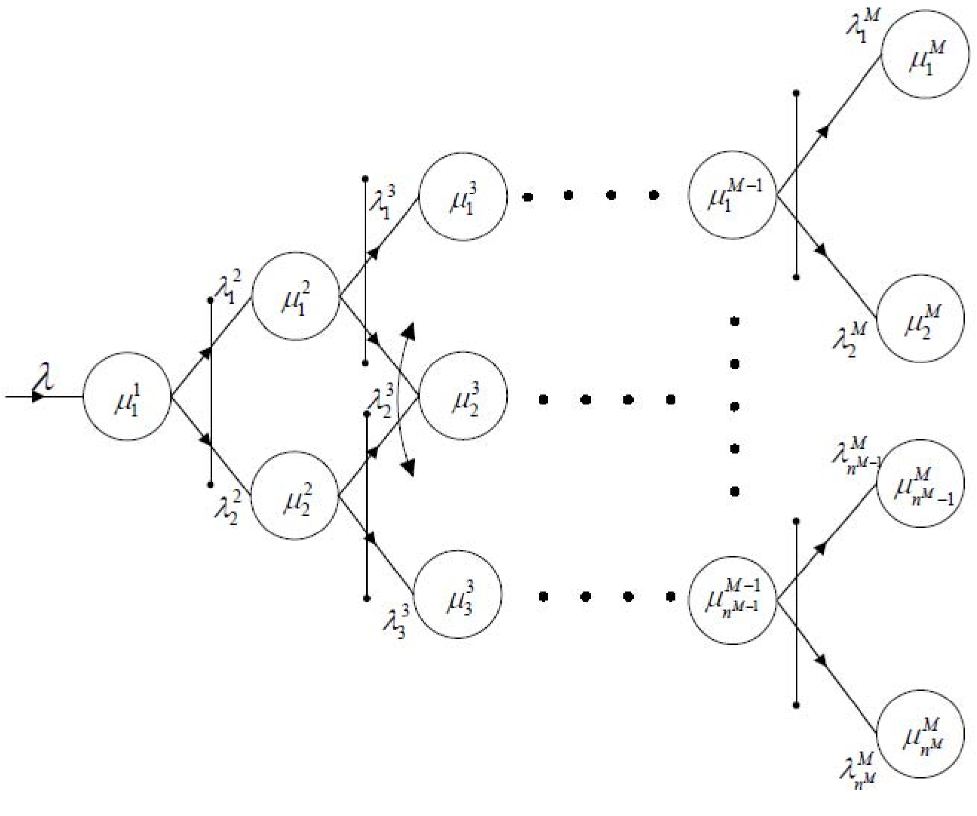

Figure 5. Sequential M-level open Jackson network with orthogonal cuts

We define two new terms and modify the mathematical program as follows.

Λ_i^j - the flow through the cut between node *i* in level *j* to its successor nodes (Λ_i^j is fixed before *node generation* and remains so).

B_i^j - the index set of the nodes in level *j*-1 that are connected to node *i* in level *j* (not including the generated nodes from level *j*-1). Note that $B_i^j \subset A_i^j$.

Mathematical formulation

$$Min \left[\sum_{j=1}^{M} \sum_{i=1}^{n^j} \frac{\lambda_i^j}{\tilde{\mu}_i^j - \lambda_i^j} + \sum_{j=1}^{M-1} \sum_{i=1}^{n^j} \frac{\lambda_{x_i^j}^{j+1}}{x_i^j - \lambda_{x_i^j}^{j+1}} \right]$$

Such that

$$\tilde{\mu}_i^j = \mu_i^j - x_i^j \qquad j = 1, ..., M;\; i = 1, ..., n^j$$

$$\tilde{\mu}_{x_i^j}^{j+1} = x_i^j \qquad j = 1, ..., M-1;\; i = 1, ..., n^j$$

$$\tilde{\mu}_i^j > \lambda_i^j \quad \forall i, j \tag{18}$$

$$\sum_{i=1}^{n^1} \lambda_i^1 = \lambda$$

$$\lambda_i^j = \sum_{a \in A_i^j} p_{a,i}^{j-1} \lambda_a^{j-1} \quad j > 1;\ \text{all nodes } i \text{ in level } j \text{ (including } x_i^j)$$

$$\sum_{w \in W_i^j} p_{i,w}^j = 1 \quad 1 \le j < M;\ \text{all nodes } i \text{ in level } j \text{ (including } x_i^j)$$

$$x_i^j, p_{ik}^j, \lambda_i^j \ge 0 \quad \forall i, j, k$$

$$\sum_{w \in W_i^j} p_{i,w}^j \lambda_i^j + \sum_{b \in B_i^j} \sum_{w \in W_i^j} p_{x_b^{j-1},w}^j \lambda_{x_b^{j-1}}^j = \Lambda_i^j \qquad j = 1, ..., m-1;\; i = 1, ..., n^j$$

It is important to emphasize that the set of equations (18) differs from the set of equations (15) only by the addition of the last constraint in (18) that computes the flow through the cuts in the network.

<u>Optimal Solution Structure to the Cut-Based Approach</u>

Differentiating the objective function (18) with respect to x_i^j and following the first condition of optimality lead to equations (16). Similarly, differentiating the objective function (18) with respect to p_{ik}^j and equating to zero lead to equations (19).

$$\frac{\tilde{\mu}_k^{j+1}}{\left(\tilde{\mu}_k^{j+1}-\lambda_k^{j+1}\right)^2}=\phi_i^j \quad k\in A_i^j \tag{19}$$

Clearly, the new set of equations (19) simplifies matters as we do not have to take into consideration the decisions in previous levels. Equations (19) no longer have the second term that appears on the right hand side of equations (17). This term aggregates all the actions that were taken in previous levels with regard to the flows. Thus, by using the cut-approximation technique one reduces the computation complexity to a polynomial order in the network size.

To further reduce the computation effort, we propose solving each cut independently, where the total incoming flow into a cut is considered fixed. The cuts can then be solved iteratively top to bottom, until one obtains convergence to a solution, or some stopping criterion is satisfied. To illustrate the efficiency of the cut-approximation method, in the following chapter we apply it to various networks and investigate it numerically.

4. Numerical Study of the Node Generation Iterative Heuristic

The effectiveness of the node generation method is shown in this section through several numerical examples. Section 4.1 presents two examples that motivate the use of the cut-approximation method. Namely, we apply this method to both a 4-level network, and to a 16-level network. We show that only few iterations are required for the WIP level to approach its minimal value. Section 4.2 presents a numerical analysis of the efficiency of the *node generation* method in 8-level networks as a function of several topological and operational parameters.

These networks are also used to compare the cut-approximation method versus a simple naïve heuristic, where a fixed percentage of the excess capacity in a parent node is transferred to the newly generated node in the downstream level. The results further illustrate the advantage of the node generation approach over the naïve approach.

4.1 Examples

Example 4.1. Consider a 4-level binary network depicted in Figure 8. Note that, in this example, level 1 is comprised of two nodes, rather than a single node, as used in previous figures. The values of the initial parameters are as follows: an arrival rate of $\lambda = 100$, service rates of $\mu_1^1 = \mu_2^1 = 145$; $\mu_1^2 = \mu_2^2 = \mu_3^2 = \mu_4^2 = 65$; $\mu_1^3 = \mu_2^3 = \mu_3^3 = \mu_4^3 = \mu_5^3 = 50$; $\mu_1^4 = \mu_2^4 = \mu_3^4 = \mu_4^4 = \mu_5^4 = \mu_6^4 = 35$; and an initial WIP of 158.6.

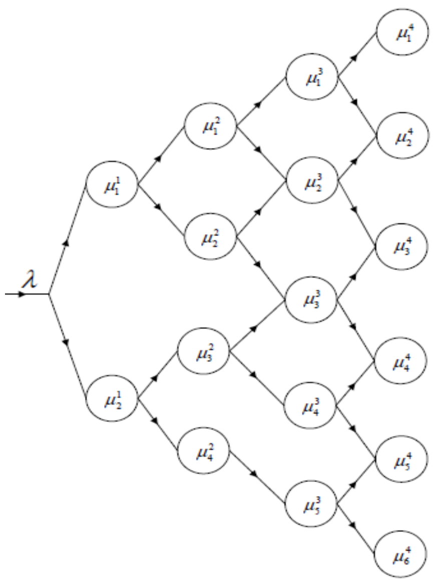

Figure 8. Sequential 4-level binary open Jackson network.

Applying the mathematical program (15) in Section 3.1, one obtains a new network with a minimal WIP of 93.6 units, an improvement of 41% with respect to the initial WIP value. The computational time of the above procedure on a PC with a 1.6 Ghz CPU is 3.3 seconds.

Using the cut-approximation formulation (18) in Section 3.2 and solving each cut iteratively, we obtain, after four iterations only, a new network with a total WIP of 94.1 units, while the computation time on the same computer is 1.2 seconds.

Next we present how to use the *node generation* method after designing and constructing a new network. The procedure for generating networks is composed of three stages (see Appendix B). The first stage generates the topology of the network. To each node two successor nodes are allocated, either by choosing them from the already available nodes in the downstream level or by generating a new node (or nodes) in the next level. In the second stage, capacities and flows are allocated to the nodes in the network: the flow of each node is randomly distributed among its successor nodes, to which capacity are then allocated proportionally to their incoming flow. In the third stage, the node generation method is implemented by using the iterative cut-approximation top to bottom. Overall, the procedure is based on several parameters that determine the network structure and its properties, namely: λ the arrival rate of jobs into the system; θ the ratio between capacity and arrival rate in the first node; ϕ the probability of generating a new successor node; η the system excess-capacity decrease factor that represents difference in the capacities between consecutive levels; T the number of maximum iterations allowed by the cut-approximation method; and M the number of levels in the network. The pseudo-code of the networks generating algorithm is presented in Appendix B.

<u>*Example 4.2.*</u> Based on the above, we now consider 16-level binary networks that were generated randomly by the procedure described above. This procedure was replicated ten times to obtain statistical measurements on WIP reduction via these sampled networks. The resulting networks contained at least 13,000 nodes each. The stopping criterion for the cut-approximation heuristic was defined to be a decrease in the total WIP of less than 0.01% between consecutive stages. This criterion was satisfied on the average after 10 iterations with an average computation time of 7 minutes. Figure 6 shows the decrease in the total WIP (where the initial WIP is defined as 100%) as a function of the number of iterations for each generated network. As can be seen, an improvement of 12.7% with respect to the initial WIP level is obtained on the average. Moreover, note that after five iterations, most of the improvement is already achieved in most of the considered networks. The generation parameters of the networks are $\eta = 0.75, \phi = 0.8, \theta = 2$ and $\lambda = 10{,}000$. In comparison, the average computation time for the optimal solution in the same networks was 91 minutes with an improvement of 14.6%.

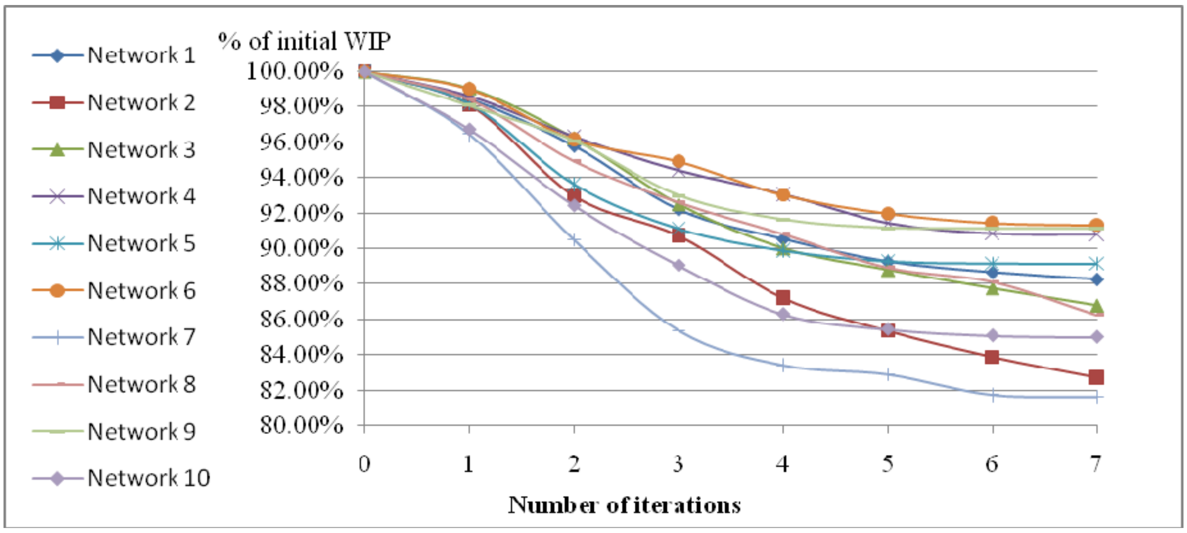

Figure 6. WIP Reduction for each of the ten 16-level networks

4.2. Numerical Study of Parameters Effects

In this section we study the effects of various network parameters on the WIP reduction where we test 8-level networks. The considered numerical study is composed of three parts. In the first part we present several examples for the convergence rate of the algorithm. Next, we analyze the effect of: *i*) capacity reduction; *ii*) number of levels; and *iii*) the ratio of capacity to flow in the first network level on the performance of the algorithm relative to different breadths of the network (defined by the probability to generate new successor nodes). For each of the mentioned parameters we perform 40 different runs based on sampled networks. A comparison to a naïve decision algorithm is given in the last part of the analysis.

Figure 7 demonstrates the percentage decrease in the total WIP as a function of the iteration number, where we fix $\theta = 2$ and $\lambda = 10{,}000$. Note that the effect of the capacity difference parameter η on the WIP value is more significant than that of ϕ (the probability of generating a new successor node). As seen, the optimization process stabilizes quite rapidly – after five to six iterations on the average.

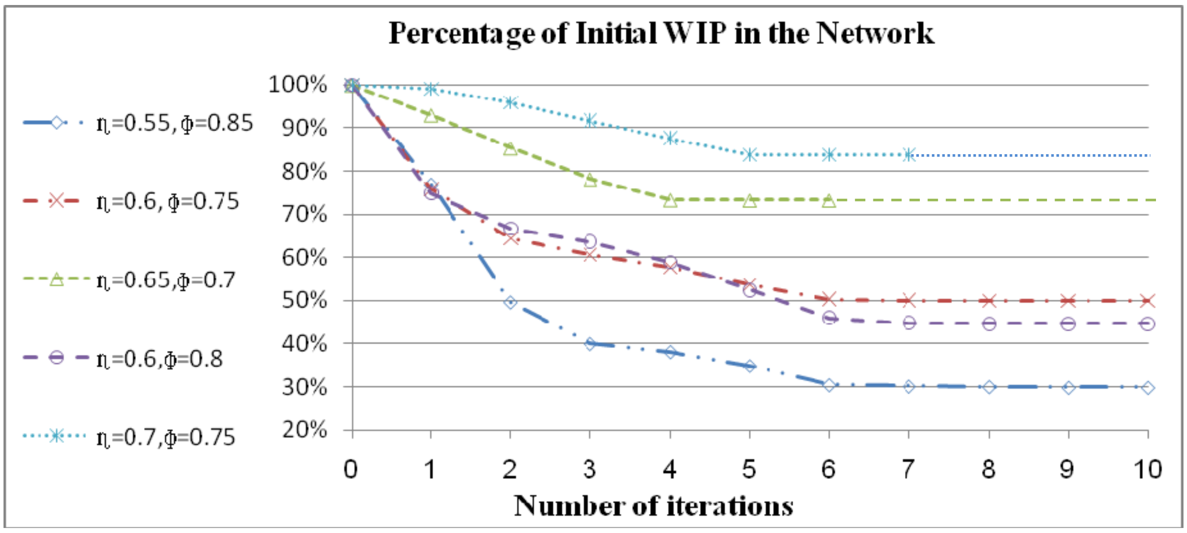

Figure 7. Examples of WIP decrease through iterations of the algorithm.

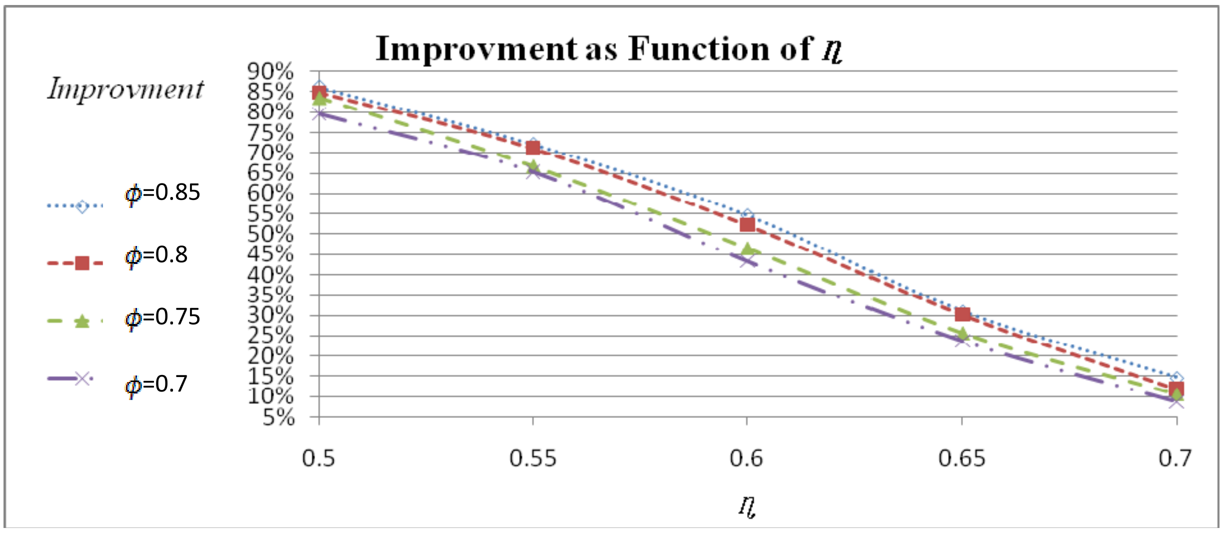

Figure 8. Performance of *Cut* Method *as a Function* of η, ϕ

Figure 8 presents the reduction of the total WIP via the *node generation* algorithm in relation to the input parameters η,ϕ. Every point in the figure represents the average WIP percentage over the 40 generated networks with the following parameters: *M*=8 levels, an initial capacity ratio of $\theta = 2$ and an arrival rate of $\lambda = 10{,}000$. Note that the average improvement of WIP is 48.11% and varies in single cases between 5% to 95%. An intuitive result that can be derived here is that as η (representing the difference in capacities between levels) increases, the reduction in the WIP level (i.e., the efficiency of the algorithm) increases. With a lesser significance one can notice that as ϕ increases (leading to a wider network) a lower level of WIP is reached. Thus, as seen by other examples, the node generation method is more efficient in wider unregulated networks.

The effect of the initial capacity ratio θ is presented in Figure 9. Each point in the figure represents the average percentage of WIP decrease over 40 generated networks. Each network was generated by using the following parameters: M=8 levels, a capacity difference parameter $\eta = 0.6$ and an arrival rate of $\lambda = 10{,}000$. Note that the improvement in the WIP level reduces as the value of θ increases. This phenomenon is attributed to the fact that the network overall capacity is determined by η and θ. Once again, for a fixed value of θ, M, λ and η, a lower level of WIP is reached in wider networks (i.e., as ϕ increases).

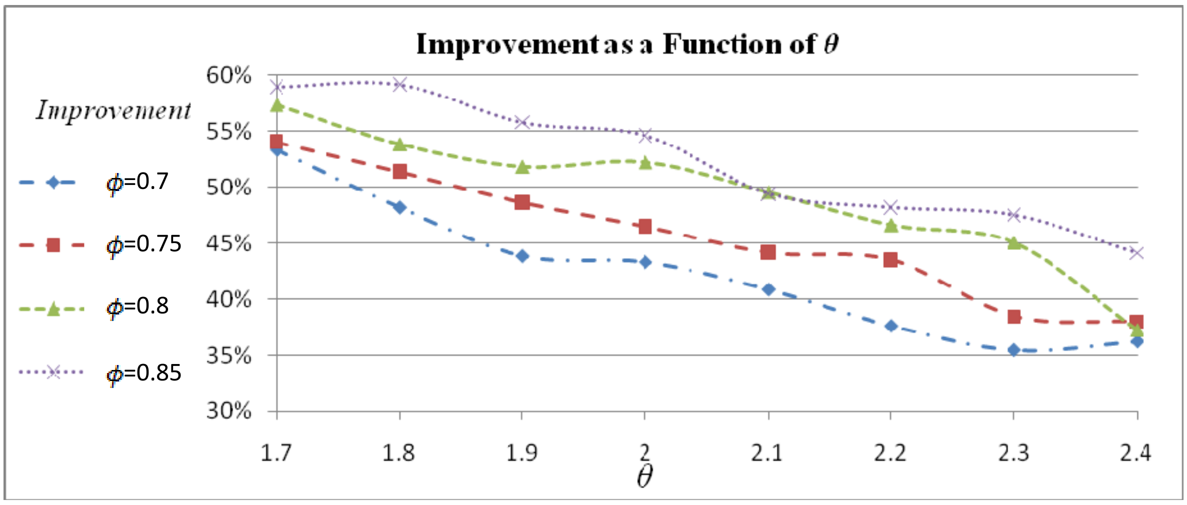

Figure 9. Improvement relative to first level ratio of capacity to flow

The effect of the number of levels in the network on the WIP decrease is presented in Figure 10. Each point represents the reduction in the WIP level with respect to the initial WIP level, as averaged over 40 networks. Each network was generated by using the following parameters: a capacity difference parameter $\eta = 0.6$; an initial capacity ration of $\theta = 2$ and an arrival rate of $\lambda = 10{,}000$. Note that as the number of levels increases, the node generated method performs better (leading to lower WIP values), as one might expect. Thus, when adding levels to the network, these new levels have lower capacities, and therefore can benefit more from the *node generation* approach. As indicated above, the improvement increases also with the width of the network (larger ϕ parameter).

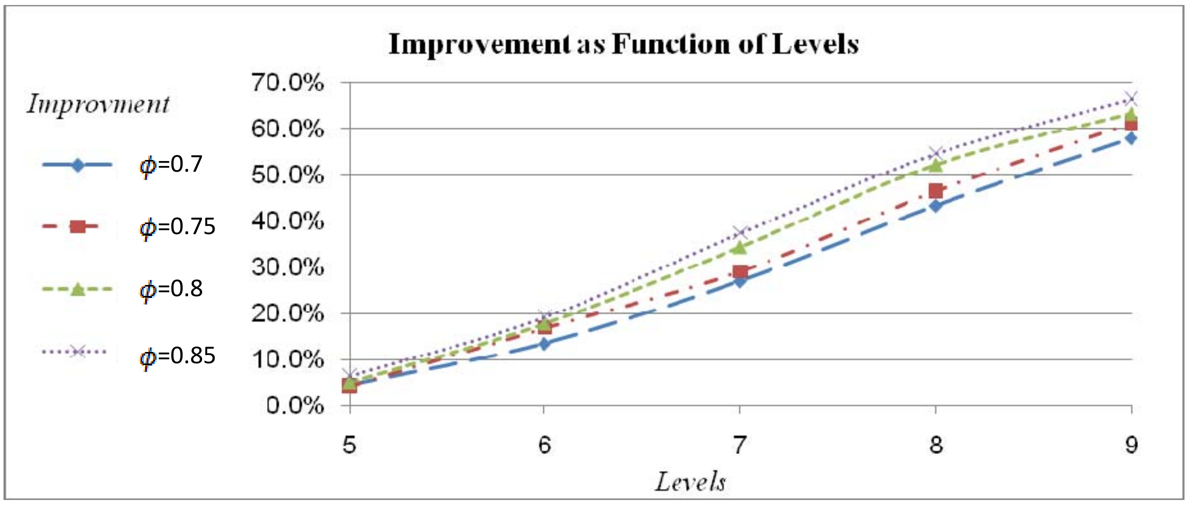

Figure 10. Improvement as a function of the number of network levels

The above numerical studies provide some insights about the factors and network parameters that influence the node generation performance. Namely, reducing the overall capacity in the system (by reducing either θ or η) leads to an improved performance of the proposed method, as well as increasing the size of the network (either by increasing its width ϕ or the number of levels *M*), as well as varying the capacities among nodes.

A Comparison of the cut-approximation versus a naïve approach

The above analysis provides insight into the affecting factors in the cut-approximation method. In this last part of the analysis we compare the cut-approximation approach versus a naïve and 'intuitive' algorithm that computes the excess capacity in a parent node and transfers a predetermined percentage of it, denoted by ε, to the newly generated downstream nodes. It searches for the minimal WIP solution within a predetermined region of excess capacity (Δ) for each cut. Thus, given the parameter ε $(0 \le \varepsilon \le 1)$ the naïve algorithm searches for the best solution (minimal WIP) in a region defined by $\Delta \cdot (\varepsilon - 0.01) \le x \le \Delta \cdot (\varepsilon + 0.01)$, where x is the amount of capacity transferred from a parent node to a new successor node. The naïve algorithm was analyzed by the following numerical study. Each iteration is generated by using the following parameters: $\phi = 0.8$; $\theta = 2$ and $\lambda = 10,000$. Table 1 shows the average improvement

(WIP reduction) achieved by both the cut-approximation and the naïve algorithm over 30 replications for different values of η, M and ε. Note that in general the improvement of the cut-approximation approach is significantly higher than the naïve algorithm. Yet, for some values of ε, the naïve algorithm can approach the WIP reduction, as obtained by the cut approximation. This attribute is partially due to the systematic creation of the analyzed networks, while less organized networks do not exhibit such phenomenon. Predicting the best value of ε in each cut and in each level requires an extensive combinatorial search on a huge number of candidates, which is computationally intractable, and avoided in this paper. Moreover, adding limitations on how much capacity can be moved to a new node can be implemented in the original model formulation of the node generation approach.

	$\eta = 0.5$				$\eta = 0.6$				$\eta = 0.7$			
Algorithm Network parameters	***M*=5**	***M*=6**	***M*=8**	***M*=10**	***M*=5**	***M*=6**	***M*=8**	***M*=10**	***M*=5**	***M*=6**	***M*=8**	***M*=10**
Naive (ε =10%)	**0.3%**	**0.9%**	**2.6%**	**2.8%**	**0.3%**	**1.1%**	**3.1%**	**4.4%**	**0.6%**	**0.7%**	**0.8%**	**0.7%**
Naive (ε =30%)	**3.2%**	**11.0%**	**32.0%**	**34.4%**	**2.4%**	**8.3%**	**24.3%**	**34.7%**	**0.1%**	**0.2%**	**0.1%**	**0.1%**
Naive (ε =40%)	**5.3%**	**18.3%**	**53.4%**	**57.3%**	**3.5%**	**12.1%**	**35.2%**	**50.4%**	**0.1%**	**0.1%**	**0.2%**	**0.2%**
Naive (ε =50%)	**6.7%**	**23.1%**	**67.4%**	**72.4%**	**4.6%**	**15.9%**	**46.3%**	**66.2%**	**0.1%**	**0.1%**	**0.1%**	**0.2%**
Naive (ε =70%)	**1.4%**	**1.8%**	**1.5%**	**1.5%**	**0.0%**	**0.1%**	**0.3%**	**0.5%**	**0.1%**	**0.2%**	**0.2%**	**0.1%**
Node Generation	**8.3%**	**28.6%**	**83.5%**	**89.6%**	**5.2%**	**17.9%**	**52.2%**	**74.6%**	**1.2%**	**4.1%**	**11.8%**	**16.9%**

Table 1. Naïve algorithm vs. node generation. The numbers in the table represent the average WIP reduction with respect to its initial value.

5. Conclusions and Future Research

We have studied optimal capacity reallocation in open Jackson networks in order to minimize total WIP, or alternatively, system response time. We start by comparing two capacity reallocation methods: *capacity redistribution* among existing nodes and *node generation*, in which capacity is allocated to newly generated downstream nodes. Although the former approach is superior in terms of overall WIP and response time, we focus on the latter approach. This approach has not been studied before, in spite of its applicability to real life settings in industrial, service and communication networks. We show that the optimal solution for the node generation approach in a 2-level network yields two necessary conditions of optimality. These conditions dictate some balance equations after capacity transfer between each parent node and its successor nodes, as well as among the successor nodes themselves. These optimality conditions and their implementations in larger multi-level networks have been studied and inspired a construction of a single ancestor *cut-approximation* method. The basic structure that we propose in this approximation can be further expanded in an attempt to tackle even more complex problems, as shown in the last section. The effectiveness of the proposed technique increases with the density of the network (both its depth and width), with irregular capacities spread within the network and with the decrease in the network overall capacity with respect to the incoming stream of jobs. The effectiveness of the algorithm has been studied numerically and found to be significant in many cases. For example, when referring to the above given numerical examples, we obtained an average decrease level of WIP between 10% to 85%.

The proposed cut-approximation approach can be used to improve the performance of ad-hoc supply chain segments or flexible manufacturing systems. One might consider the implementation of the cut-approximation techniques not only in the node generation approach, but also for capacity redistribution. The cut technique is a heavily constrained mathematical program. Relaxation or modification of these constraints has a potential for even greater improvement of network performance. This can be done by using the available capacities and without requiring a global topological optimization - a requirement that often leads to computational limitations. Modeling the system by multi-server queues can improve the optimal result for addressing reallocation of capacities with discrete resources (e.g., workers) or scheduling considerations.

Another direction that can be investigated is by using local heuristics. For example, each node in the network can use local information of neighborhood nodes to maintain the balance constraints by giving or getting capacity in a fashion similar to Conway's *game of life* (Gardner (1970)). It might be of practical importance to study if these ad-hoc networks, such as peer-to-peer segments that exist in the internet nowadays, can improve their performance locally to a satisfactory degree.

References

M. Armony, “Dynamic Routing in Large-Scale Service Systems with Heterogeneous Servers”, Queueing Systems: Theory and Applications, Vol. 51, No 3-4, Pg. 287 - 329, 2005.

V. Bhaskar and P. Lallement, “Modeling a supply chain using a network of queues” Applied Mathematical Modelling 34, Pg. 2074–2088, (2010).

V. Bhaskar and P. Lallement, “Queuing network model of uniformly distributed arrivals in a distributed supply chain using subcontracting” Decision Support Systems 51, Pg. 65–76, 2011.

G. R Bitran. and S. Dasu, “A Review of Open Queueing Network Models of Manufacturing Systems.” Queuing Systems. Vol. 12, Numbers 1-2, Pg. 95-133, 1992.

G.R. Bitran and D. Sarkar, “Targeting Problems in Manufacturing Queueing Networks – An Iterative Scheme and Convergence, European Journal of Operation Research Vol.76, Pg. 501-506, 1994.

G.R. Bitran and D. Tirupati, “Capacity planning in manufacturing networks with discrete options”, Annals of Operations Research, Vol. 17, No. 1, Pg. 119-135,1989-B.

G.R. Bitran and D. Tirupati, “Tradeoff Curves, Targeting and Balancing in Manufacturing Queueing Networks”, Operation Research, Vol. 37, No 4, Pg. 547-564,1989-A.

K.M. Bretthauer and B. Shetty, “The nonlinear resource allocation problem, Operations Research”, Vol. 43, No.4, Pg. 670-683, 1995.

K.M. Bretthauer and M.J. Cote, “A model for planning resource requirements in health care organizations”, Decision Science, Vol. 29, No.1, Pg. 243-270, 1998.

K.M. Bretthauer, “Optimal Service and Arrival Rates in Jackson Queueing Networks”, Naval Research Logistics, Vol. 47, No. 1, Pg. 1-17, 2000.

A.H. Buss, S.R. Lawrence, and D.H. Kropp, “Volume and capacity interaction in facility design”, IIE Transactions, Vol. 26, No 4, Pg. 36-49, 1994.

J. Buzacott and J. G. Shanthikumar ,“Design of Manufacturing Systems Using Queueing Models” Queuing Systems, Vol. 12, No. 1-2, Pg. 135-214.,1992.

J.A. Buzacott and J.G. Shanthikumar, “Stochastic models of manufacturing systems”, Prentice Hall, Englewood Cliffs, NJ, 1993.

J. Buzacott and D. Yao, “On queuing network models of flexible manufacturing systems”, Queuing Systems, Vol. 1, No.1, Pg. 5-27, 1986.

T.B. Crabill, D. Gross and M.J. Magazine, “A classified bibliography of research on optimal design and control of queues”, Operations Research, Vol. 25, No. 2, Pg. 219-232, 1977.

S. Dewan and H. Mendelson, “User delay costs and internal pricing for a service facility, Manage Science”, Vol. 36, No. 12, Pg. 1502-1517, 1990.

M. Dong and F.F. Chen, “Performance modeling and analysis of integrated logistic chains: An analytic framework”, European Journal of Operational Research 162, Pg. 83–98 (2005)

M. Gardner, “Mathematical games: the fantastic combinations of John Conway's new solitaire game "life"”, Scientific American Vol.223, Pg.120-123, 1970.

S.B. Gershwin, "Manufacturing Systems Engineering", TPR Prentice Hall, 1994.
A.Y. Ha, “Inventory rationing in a make-to-stock production system with two priority classes and backordering” Management Science, Vol. 43, Pg. 1093–103, 1997.

L. F. Hsu, C. S. Tapiero, and C. Lin “Network of Queues Modeling in Flexible Manufacturing Systems: A Survey”, Operations Research, Vol. 27, No. 2, Pg. 201-248, 1993.

J. S. Jane, N. R. Srinivasa Raghavan, “A queuing approach for inventory planning with batch ordering in multi-echelon supply chains, Central European Journal of Operations Research, Vol. 17, No. 1, Pg. 95-110, 2009.

L. Kleinrock, Communication Nets; Stochastic Message Flow and Delay, McGraw-Hill Book Company, New York, 1964.

L. Kleinrock, L. Fratta and M. Gerla, “The flow deviation method: An approach to store-and-forward communication network design”, Networks, Vol. 3, No. 2, Pg. 97-133, 1973.

L. Kleinrock, “Queueing systems”, Volume I: Theory, Wiley, New York, 1975.

L. Kleinrock, “Queueing systems”, Volume II: Computer applications, Wiley, New York, 1976.

L. Kleinrock, and M. Gerla “On the topological design of distributed computer networks”, IEEE Transactions: Communication, Vol. 25, No.1, Pg. 48-60, 1977.

P. Kouvelis and D. Tirupati, “Approximate Performance Modeling and Decision Making for Manufacturing Systems: A Queueing Network Optimization Framework,” Journal of Intelligent Manufacturing, Vol. 2, No.2, Pg. 107-l34, 1991.

H. Lee and S. Huang, “Decentralized Multi-Level Supply Chains: Incentives and Information”, Management Science, Vol. 45, No 5, Pg. 633-640,1999.

Y. Li and B. Ying,” A Performance Assessment on Manufacturing Nodes in robust design of Supply Chain Networks (SCNs)”,IE&EM 09, Pg. 1442 – 1446, 2009.

A. Mandelbaum, N. Gans and G. Koole," Telephone Call Centers: Tutorial, Review, and Research Prospects", Manufacturing & Service Operations Management, Vol. 5, No. 2, Pg. 79 – 141, 2003 .

T.M. Ng and D.B Hoang, "Joint optimization of capacity and flow assignment in a packet-switched communications network", IEEE transactions on communications, Vol. 35, No.2, Pg. 202-209, 1987.

N. R. Srinivasa Raghavan, N. Viswanadham, "Generalized queueing network analysis of integrated supply chains, Int. J. Prod. Res., 2001, Vol. 39(2), Pages 205-224, 2001.

S. Stidham, "Pricing and capacity decisions for a service facility: Stability and multiple local optima", Management Science, Vol. 38, No.8, Pg. 1121-1139, 1992.

R. Suri, J. L. Sanders, and M. Kamath ,"Performance Evaluation of Production Networks," Handbooks in Operations Research and Management Science, Volume 4, Pages 199-286, 1993.

R. Suri, Quick Response Manufacturing, Productivity Press, Portland, OR, USA, 1998.

D. D. Yao and J.G. Shanthikumar, "The optimal input rates to a system of manufacturing cells", INFOR, Vol. 25, No. 7, Pg. 57-65, 1987.

U. Yechiali, "On the Relative Waiting Times in the GI/M/s and the GI/M/1 Queueing Systems", Operational Research Quarterly, Vol. 28, No. 2, Pg. 325-337, 1977.

Appendix A: Optimal Distribution of Extra Capacity among *n* Parallel Servers

Consider a system with *n* nodes as depicted in Figure 11. Let λ be the overall external arrival rate, to be distributed among the *n* nodes. Let the existing capacity of node *i* be μ_i^1 i=1,2,…,n. Assume that there is additional capacity Γ to be distributed among the n nodes such that node *i* gets y_i additional units of capacity, $\sum_{i=1}^{n} y_i = \Gamma$. Let p_i be the fraction of flow directed to node *i* and let $\tilde{\mu}_i^1 = \mu_i^1 + y_i$.

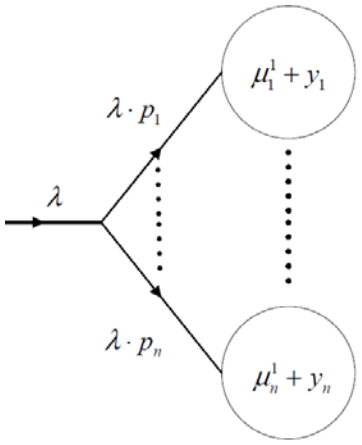

Figure 11. Capacity Distribution for *n* Parallel Servers

The problem A1 below aims to obtain the minimal WIP in the network, where y_i, p_i are considered as decision variables, and μ_i^1 and λ are treated as the problem parameters.

Problem A1

$$Min\left\{\sum_{i=1}^{n} \frac{\lambda_i^1}{\tilde{\mu}_i^1 - \lambda_i^1}\right\} \equiv Min\ F\left(\vec{y}, \vec{p}\right)$$

Subject to

$$\tilde{\mu}_i^1 > \lambda_i^1 \quad i = 1, \ldots, n$$

$$\sum_{i=1}^{n} p_i = 1$$

$$\sum_{i=1}^{n} y_i = \Gamma$$

$$\tilde{\mu}_i^1 = \mu_i^1 + y_i \quad i = 1,\dots,n$$

$$\lambda_i^1 = p_i \cdot \lambda \quad i = 1,\dots\dots,n$$

$$y_i \geq 0 \quad k = 1,\dots,n$$

$$0 \leq p_i \leq 1 \qquad i = 1,\dots,n$$

Theorem A1

When transferring extra capacity Γ to one server in a system composed of n parallel $M/M/1$ servers, the optimal solution in terms of minimal WIP is obtained by transferring Γ to the server with the largest initial capacity.

Proof

Consider first two $M/M/1$ parallel servers with service intensities μ_1 and μ_2, respectively, and combined arrival rate λ, where the objective is to minimize the WIP in the network. Accordingly, we solve Problem A2 below, in which the goal is to reduce the WIP of the parallel servers by choosing the proper arrival rate λ_1 to node 1.

Problem A2

$$Min\left\{\frac{\lambda_1}{\mu_1-\lambda_1}+\frac{\lambda-\lambda_1}{\mu_2-\lambda+\lambda_1}\right\}$$

Subject to

$$\mu_1 - \lambda_1 > 0$$
$$\mu_2 - \lambda + \lambda_1 > 0$$
$$\lambda - \lambda_1 \geq 0$$
$$\lambda_1 \geq 0$$

Since the optimal value of λ_1 is $\frac{\mu_1\sqrt{\mu_2}-\mu_2\sqrt{\mu_1}+\lambda\sqrt{\mu_1}}{\left(\sqrt{\mu_2^1}+\sqrt{\mu_1^1}\right)}$, the total WIP is given by $WIP = WIP_1 + WIP_2 = \frac{2\lambda-\left(\sqrt{\mu_1}-\sqrt{\mu_2}\right)^2}{\mu_1+\mu_2-\lambda}$. As stated before, if λ_1 is negative we route the flow only to the node with the largest capacity.

Now consider the system with *n* queues, and look at a solution in which we transfer $\alpha\cdot\Gamma$ ($0\le\alpha\le1$) capacity units to server *k* and $(1-\alpha)\Gamma$ capacity units to server *j*, while rerouting the arrival rates only for those two servers. Suppose the two servers have a combined arrival rate of $\lambda_j+\lambda_k\equiv\Upsilon$. Thus, the minimal WIP of the two servers as a function of α is $WIP(\alpha)=\frac{2\Upsilon-\left(\sqrt{\mu_k+\alpha\cdot\Gamma}-\sqrt{\mu_j+(1-\alpha)\cdot\Gamma}\right)^2}{\mu_k+\mu_j+\Gamma-\Upsilon}$. Since $WIP(\alpha)$ is a concave function, the minimum is obtained either in the point $\alpha=1$ or $\alpha=0$. Thus, consider

$$WIP(1)-WIP(0)=\frac{\left(\sqrt{\mu_j+\Gamma}-\sqrt{\mu_k}\right)^2-\left(\sqrt{\mu_k+\Gamma}-\sqrt{\mu_j}\right)^2}{\mu_k+\mu_j+\Gamma-\Upsilon}=\frac{-2\sqrt{\mu_j+\Gamma}\sqrt{\mu_k}+2\sqrt{\mu_k+\Gamma}\sqrt{\mu_j}}{\mu_k+\mu_j+\Gamma-\Upsilon}.$$

It follows that $WIP(1)>WIP(0)$ if and only if $\mu_k<\mu_j$, implying that it is optimal to allocate Γ to the node with the largest capacity between the two nodes. Therefore, if one applies this notion repeatedly between each pair of nodes, the optimal solution is to transfer Γ capacity units to the server with the largest service intensity out of all existing nodes.

Appendix B: Networks-Generating Algorithm

Our algorithm is composed of three parts: *i*) topology; *ii*) capacity; and *iii*) flow allocation and optimization. The algorithm requires several parameters for its operation, namely: λ the arrival rate of jobs to the system, θ the capacity factor of the first node (the ratio between the initial node capacity and arrival rate), ϕ the probability for generating a new child node, η system excess capacity decrease factor, *T* number of maximum iterations before the optimization process stops and *M* number of levels in the network.

Let us now present the pseudo-code of the algorithm:

I. Topology: The first part of the algorithm creates the topology of the network. For each node we ensure two sons either from the available nodes in the next level or a new node that is created and added to the next level.

```
    1.1 $j$=0, $i$=1.
    1.2 Draw a random number $r(0,1)$. If $r > \phi$, choose a node on level $j$+1 and set it as the right son; else create a new node and set it as the right son.
    1.3 Draw a random number $r(0,1)$. If $r > \phi$, choose a node on level $j$+1 that is not the right son and set it as the left son, or else create a new node and set it as the left son.
    1.4 If $i < n^j$ then $i = i+1$ and return to stage 1.2, else continue.
    1.5 If $j < M$ then $j = j+1$, $i = i+1$ and return to stage 1.2 else end.
```

II. Capacity and Flow Allocation: The second part of the algorithm allocates flow and capacity to the nodes in the network. Each node flow is randomly distributed between his successors, which are then allocated with capacity proportional to their flow.

```
    2.1 $j$=0, $i$=1, $\mu_1^0 = \theta \cdot \lambda, \lambda_1^0 = \lambda$.
    2.2 Draw a random number $r(0,1)$ set $\lambda_1^1 = r \cdot \lambda_1^0$ and $\lambda_2^1 = (1-r) \cdot \lambda_1^0$
    2.3 $j = j+1$
    2.4 $\mu_i^j = \frac{\lambda_i^j}{\lambda} \cdot \left(\lambda + (1-\theta) \cdot \eta^j\right)$
```

2.5 If $j < M-1$, draw a random number $r(0,1)$, set $\lambda_{right\ \ son}^{j+1} = r \cdot \lambda_i^j + \lambda_{right\ \ son}^{j+1}$ and $\lambda_{left\ \ son}^{j+1} = (1-r) \cdot \lambda_i^j + \lambda_{left\ \ son}^{j+1}$

2.6 If $i < n^j$ then $i = i+1$ and return to stage 2.4

2.7 If $j < M-1$ then $i = 1$ and return to stage 2.3 else end.

III. Optimization: The last part of the algorithm implements the *cut* approach for each cut iteratively, top to bottom.

3.1 *t*=1,*j*=0, *i*=1.

3.2 Set ϑ as the initial λ_i^j, $p = i$, $S = \{right\ son, left\ son\}$ solve the *cut* and update the network accordingly.

3.3 If $i < n^j$ then $i = i+1$ and return to stage 3.2

3.4 If $j < M-1$ then $i = 1, j = j+1$ and return to stage 3.2.

3.5 If $t = T$ or there is no improvement in the iteration then end, else $t = t+1, j = 0, i = 1$ and return to stage 3.2