MASTER

Joint buffer-server optimization in general queueing networks using the generalized expansion method

Andriansyah, R.

Award date:
2007

Joint Buffer-Server Optimization in General Queueing Networks Using the Generalized Expansion Method

R. Andriansyah

June 14, 2007

Acknowledgement

This thesis brings an end to one of the greatest chapters in my life as a master student of Operations Management and Logistics at the Technische Universiteit Eindhoven. The whole program, particularly the making of this thesis, was an intense learning experience for me. I would like to express my deepest gratitude to the following people that have supported me in my thesis.

1. My first supervisor, Dr. Tom van Woensel, for all discussions and help in solving any kind of problems that I encountered in the thesis. Thanks for making everything seems easy and doable (even when it's actually not).

2. My second supervisor, Prof.Dr.ir. J.W.M. Bertrand, who gave me critical comments and insights during the intermediate presentation of the thesis. Your insight is invaluable for this thesis.

3. My family back in Jakarta, Indonesia: my moms, dad, brothers, and sisters, who continuously supported me with their transcontinental prayers.

4. My friends, particularly from the Operations Management & Logistics, and the Innovation Management master programs, for all the things that we did together from taking courses, doing sports, to having dinners. Some names to mention include: Edward, Gokhan, Said, Peter, Gaus, Chun, and Andrew.

5. Finally my lovely wife, Dina Wiyasti, for her infinite buffers of love and support throughout the years. It's never enough for me to say this: thank you for everything, sweetheart.

And for all of you who have made my two-year study at TU Eindhoven an unforgettable experience, *bedankt allemaal!*

Eindhoven, June 2007

Ricky Andriansyah

Summary

In this thesis, we consider a joint buffer-server optimization in general $M/G/c/K$ queueing networks. The research question posted here is: "*given a specific system configuration and a limited availability of budget and space in a manufacturing environment, how should one decide upon the number of buffers and servers to maximize productivity?*". This question can be formulated mathematically into:

"Given a queueing network structure with N nodes, an external arrival rate of λ, and service rates of μ_i, what is the optimal number of servers (x_i) and buffers (y_i) at each node i so as to maximize the corresponding throughput rate $\theta(x,y)$?

The Generalized Expansion Method (GEM) is used as an approximation technique to calculate the throughput of such networks. This performance evaluation tool will be used in combination with the optimization tool (search method), namely the *Powell algorithm* and *Coggin algorithm* to search for the optimal allocation of buffers and servers, given three objectives as follows:

1. Minimize the total number of servers.
2. Minimize the total number of buffers.
3. Maximize the resulting throughput.

Two optimization methodologies are developed, namely the *BAP-CAP interaction* and the *single objective BCAP*. The results from these two methodologies will be compared with the results from another methodology, the *complete enumeration*. This comparison is done to see the quality of the two optimization methodologies for a small, simple queueing network structure.

The first methodology, namely the BAP-CAP interaction, is based on the two previous studies of Buffer Allocation Problem (BAP) and Server Allocation Problem (CAP) by Smith *et.al* (2006). In the BAP, the optimal buffer allocation is identified while assuming a fixed server allocation. In the CAP, on the contrary, the optimal server allocation is identified while assuming a fixed buffer allocation. We will use the output of optimal buffer allocation (K^*) from the BAP as the input for the CAP. The resulting optimal server allocation (c^*) from the CAP will be then used as the input for the BAP. This interaction is carried on continuously until all solutions are generated for a given queueing network structure. *Non-dominated* solutions are then identified from the other *dominated* solution.

The second methodology, namely the single objective BCAP, combines all the three objectives mentioned previously into one objective function. This is done using a *Lagrangian relaxation*, in which all complicating constraints are included to the objective function by giving a penalty for each of the constraint. The resulting objective function (referred to as the *relaxed objective*) is used in the powell and coggin algorithm to search for the optimal allocation of servers and buffers. Two scenarios are developed, namely the case where the prices of servers and buffers

are (1) equal, and (2) different.

Three queueing network structures, namely the *series, split*, and *merge* topologies are used for each of the methodology. The network size (N) for these three topologies are set to 3, 5, and 7 nodes. The processing rate (μ_i) is set to 10 at each node i in the network. Finally, three arrival rates (λ) of 3, 7, and 15 are used and three squared coefficient of variation (CV) of 0.5, 1.0, and 1.5 are used to represent the servers' variability. The combination of these parameters are used as the experimental settings.

The results from the first methodology (BAP-CAP interaction) show that there are several *non-dominated* solutions for a given queueing network structure. More non-dominated solutions are obtained for networks with high arrivals. In the series topology, the BAP-CAP interaction identifies two server-buffer allocation pattern, namely (1) equal number of servers and buffers are allocated at each node, and (2) extra allocation of servers and buffers to either start or end node of the network. In the split/merge topology, there is a tendency to add more servers and buffers to the first splitting/last merging node in the network, which is the node with highest arrival. The buffer allocation is also more extreme than the server allocation. We also note that the non-dominated solutions from the BAP-CAP are not necessarily optimal allocation of buffers and servers.

The results from the second methodology (single objective BCAP) show that the is only *one* optimal buffer-server allocation for a given queueing network structure and a set of starting search point. The starting search points have a significant effect on the optimal solution, where different starting search points will lead to different server-buffer allocations. In the series topology, an equal allocation of servers to each node is apparent for low and medium arrivals. For high arrivals, the optimal allocation shows extra servers are given to either the start or the end node. Buffer allocation, however, is more unpredictable. In the split/merge topology, there is a tendency to add more servers and buffers to the first splitting/last merging node in the network, which is the node with highest arrival. We observe that nodes that already have extra servers are not allocated with extra buffers. In the case where servers are more expensive than buffers, there will be two possible changes in the optimal allocation as compared to the case of equal price. It can be that the number of servers is hold similar while the number of buffers is increased slightly. Another possibility is that the number of servers is reduced slightly and results in a significant addition to the number of buffers. In the case where buffers are more expensive that servers, the change in the optimal buffer-server allocation shows a small addition of servers and a relatively large reduction of buffers. As we compare the single objective BCAP with the BAP and CAP methodologies, we see that (1) the single objective BCAP does not generate excessive buffer allocation for networks with high arrivals, and (2) the single objective BCAP does not generate allocations where the number of servers are larger than the number of buffers. These two findings reflect the quality of the single objective BCAP.

Compared to the results of the complete enumeration, we see that the optimal allocation from the single objective BCAP has lower number of buffers and servers than that of the BAP-CAP interaction. Both of the optimization methodologies are able to find an optimal allocation according to the complete enumeration strategy. The two methodologies also work well with complex topologies, where series, split and merge topologies are combined in one queueing network structure. In general, however, we argue that the single objective BCAP is more preferable to the BAP-CAP interaction.

Chapter 1

Introduction and motivation

The optimization of manufacturing systems and production lines has been the focus of numerous studies for decades. Queueing networks are commonly used to model such systems because they account for dynamics, interactions, and uncertainty in an aggregate way (Suri (1985)). Of particular interest here is the joint optimization of the number of buffers and servers, since both of them represent a significant amount of investment in a company.

Throughout the years, there has been a vast literature regarding optimization of queueing networks. Optimization problems in this field can be classified into three main groups according to the decision variable of interest, which are referred to as:

1. Buffer allocation problem, *BAP* (see Smith and Cruz (2005) for a comprehensive classification of the methodologies)
2. Server allocation problem, *CAP* (see *e.g.* Hillier and So (1989), Magazine and Stecke (1996))
3. Workload allocation problem, *WAP* (see *e.g.* Hillier and Boling (1979)).

Despite the numerous studies in optimization of queueing networks, it is interesting to note that there is scant literature about simultaneous optimization of the buffers and servers.

One of the most comprehensive studies of optimization in queueing system was done by Hillier and So (1995), in which a total of seven optimization settings were studied. In this study, they considered a serial production line with up to eight stations. The underlying assumption that they used was that there is always a customer available to begin service at the first station, hence, overlooking the arrival rate. This assumption causes no jobs to be lost due to the blocking, and blocking will only slow down the entire process within the production line. They started with the three independent (buffer, server, and workload) optimization problems. Specifically, they put the values of two variables constant to search for the optimal setting for the other one decision variable (*i.e.* holding the number of servers and buffers constant to determine the optimal workload). In all of the three independent settings, they empirically found that the optimal setting that maximizes throughput follows the "bowl phenomenon" (Hillier and Boling (1979)). That is, the interior stations should be given preference (*i.e.* more servers, more buffers, or less workload) than the other stations. In the case of joint optimization of server and workload, they found that the maximum throughput is obtained using the setting that they refer to as the "L-phenomenon" (Hillier and So (1995)). Under this setting, each station is allocated with a single server and a decreasing workload towards the end of the line. The end station is then allocated with all of the extra servers and the highest workload. The biggest limitation of their

study was the optimization methodology itself. To optimize the number of servers and buffers, they used enumeration strategy, where the throughput from all possible combinations of buffers and servers are evaluated to search for the optimal setting. As such, their results are limited only to a relatively small-sized queueing networks. Despite the limitations, the results of this research serve as the building block for subsequent studies regarding optimization of queueing networks.

Spinellis *et.al* (2000) extended the work of Hillier and So (1995) by combining Simulated Annealing and the Generalized Expansion Method (GEM) to optimize the performance of production lines. They used the same queueing settings with that of Hillier and So (1995) for comparison purposes. They also assumed the first station in the line is never starved (hence, infinite buffers for the first node). While some similarities in the results existed, striking differences in allocation of buffers, servers, and workload were found as well (*i.e.* no bowl phenomenon was encountered in either workload or buffer allocation, and no L-phenomenon was found in joint server-workload allocation). With regards to the joint buffer-server optimization, they found that the buffer and server allocation results are roughly similar to the results by Hillier and So (1995). That is, servers tend to accumulate towards the beginning and the middle of the line, while buffers tend to accumulate towards the end of the line. The most valuable contribution of their research was that they were able to optimize a large production line (up to 70 stations with 140 buffers and servers) in reasonable time.

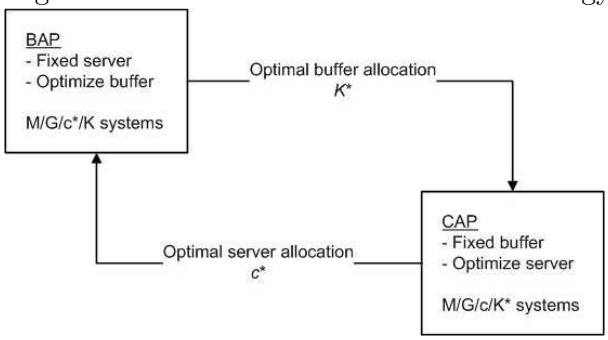
Literature about joint buffer-server optimization have been dominated by the serial production line setting. Other topologies such as merging and splitting have not been subject to joint buffer-server optimization. These topologies are mainly discussed for buffer allocation problem only (see for instance Smith and Cruz (2005) and Smith *et.al* (2006)). A possible explanation for this limitation is the fact that joint buffer-server optimization is a nonlinear, multi-objective optimization problem with integer decision variables, which is hard to solve. As such, complex network topologies would make the problem even more difficult.

In this report, we discuss joint buffer-server optimization for networks with finite queues and general service times, thus $M/G/c/K$ queueing networks. This particular network is characterized by blocking that eventually degrades the performance, or the *throughput*, of the network. Three objectives are considered in this joint buffer-server optimization problem, namely *maximizing* the throughput, *minimizing* the total number of servers, and *minimizing* the total number of buffers. We approach this problem by using two optimization methodologies, both of which are based on a search method (*generative model*) in conjunction with the Generalized Expansion Method (GEM), which was developed by Kerbache and Smith (1987), as an *evaluation tool*. Using GEM, it is possible to evaluate the performance of complex queueing networks including serial, merge, and split topologies, as well as the combination of these topologies (see *i.e.* Jain and Smith (1994), Smith and Cruz (2005), Andriansyah *et.al* (2006)).

The first methodology is referred to as *Buffer Allocation Problem (BAP) - Server Allocation Problem (CAP) interaction*. In this method, we interact the methods from two independent optimization studies, namely the BAP and CAP. In BAP, the number of servers are fixed and the optimal allocation of buffers are identified. The CAP, on the contrary, assumed a fixed number of buffers and optimize the allocation of servers. The idea behind BAP-CAP interaction is to use the optimal buffer allocation from the BAP as the input buffer for the CAP. In the same way, the optimal server allocation from CAP will be then used as the input server for the BAP. Figure 1.1 depicts the interaction of the two independent optimization methods to be used in

this first methodology.

Figure 1.1: BAP-CAP interaction methodology



The second methodology is referred to as the *single objective buffer-server allocation problem (BCAP)*. That is, we simultaneously optimize the server and buffer allocations by using a single objective function. The original objective function to be optimized is as follows.

$$Z = min(f(\mathbf{x},\mathbf{y}) = \sum_{i=1}^{N} x_i + \sum_{i=1}^{N} y_i) \tag{1.1}$$

subject to:

$$\theta(\mathbf{x},\mathbf{y}) \geq \theta^{\tau} \tag{1.2}$$

$$x_i \leq y_i \tag{1.3}$$

$$x_i, y_i \in 1, 2, 3..., \forall i \tag{1.4}$$

where:

x_i = number of servers at node i
y_i = number of buffers at node i
$\theta(\mathbf{x},\mathbf{y})$ = resulting throughput
θ^{τ} = threshold throughput

This objective function has two complicating constraints, namely constraint 1.2 and constraint 1.3. One can argue that without this two constraints, the objective function is fairly simple and thus easy to solve. To "simplify" the objective function, we will use the *Lagrangian relaxation* to incorporate the complicating constraints into the objective function. Subsequently, we will use the GEM in combination with the Powell algorithm and the Coggin algorithm to optimize the buffer and server allocations based on the "relaxed" objective function. More details on both of the above methodologies will be further discussed in the next chapters.

The main contributions of this report are:

1. We consider a joint buffer-server optimization for general servers, thus $M/G/c/K$ systems. According to our knowledge, this type of system has never been a subject for joint buffer-server optimization in the existing literature.

2. The network configuration is not restricted only to a serial production line, as in all existing literatures about joint buffer-server optimization. We consider arbitrary settings of queueing networks, including a pure series, merge, and split topology. A combined topology that consists of all these three topologies is also discussed.

3. Two optimization methodologies are used, namely the BAP-CAP interaction and the single objective BCAP. Both of which have never been used for joint buffer-server optimization. From these two methodologies, we are able to compare the performance of them both and gain insight about the optimal buffer and server allocation patterns.

4. We compare the results of the two optimization methodologies with the results from the complete enumeration technique for a small series network with 3 nodes, a maximum number of servers of 6, and a maximum number of buffers of 9. From this comparison, we are able to further assess the quality of solutions from the two optimization methodologies.

5. The combination of all settings for the two optimization methodologies used in this report sums up to 533 experiments. In addition, the complete enumeration covers a total of 662 settings. These bulk of data can be used for comparison purposes for the future research on the similar topic of joint buffer-server optimization.

The remainder of the paper will be organized as follows. First, we will review the literature on the GEM and the search methods that we used for the joint buffer-server optimization. Next, we elaborate the queueing network structure that will be used throughout this paper and create a simple network structure that we optimize using the complete enumeration technique. Afterwards, we discuss in separate sections the joint buffer-server optimization using the two methodologies as mentioned previously. We then discuss the performance of the optimization methodologies for a combined network of series, split and merge topologies. The last section concludes the research, reflects on managerial implication of the research, and elaborates on possible topics for future research.

Chapter 2

Literature review

The joint buffer-server optimization combines the use of the *evaluation tool* and *optimization tool*. Evaluation tool is the method that we use to evaluate the performance of a queueing network. Throughout this research, we used the Generalized Expansion Method as the evaluation tool to approximate the throughput of the finite $M/G/c/K$ queueing networks. Subsequently, we use the optimization tool to search for the optimal allocation of buffers and servers based on the pre-defined objective function. In this section, we will elaborate both evaluation and optimization tools that are used for the joint buffer-server optimization.

2.1 Evaluation tool: The Generalized Expansion Method (GEM)

The GEM is an effective and robust approximation technique to measure the performance of open finite queueing networks. Developed by Kerbache and Smith (1987), the GEM has become an appealing approximation technique for performance evaluation of queueing networks due to its accuracy and relative simplicity. Not to mention, exact solutions to performance measurement are restricted only to very simple networks and simulation requires considerable amount of time. The effectiveness of GEM as a performance evaluation tool has been presented in many papers, including Kerbache and Smith (1988), Kerbache and Smith (1987), Jain and Smith (1994), Smith (2003), and Andriansyah *et.al* (2006).

The GEM is basically a combination of two approximation methods, namely repeated trials and node-by-node decomposition. To evaluate the performance of a queueing network, the GEM first divides the network into single nodes with revised service and arrival parameters. Blocked customers are registered into an artificial "holding node" and are repeatedly sent to this node until they are serviced. The addition of the holding node *expands* the network and transforms the network into an equivalent Jackson network, where each node can be solved independently. Generally, the GEM assumes a *type I blocking* that is commonly referred to as transfer blocking. This type of blocking occurs when the service of a job is completed at a certain node but it cannot proceed to the next node because the queue is full.

The following notation for the GEM will be used throughout the paper (Kerbache and Smith (1988)).
Λ = external Poisson arrival rate to the network
λ_j = Poisson arrival rate to node j
$\tilde{\lambda_j}$ = Effective arrival rate to node j
μ_j = exponential mean service rate at finite node j
$\tilde{\mu_j}$ = effective service rate at finite node j due to blocking

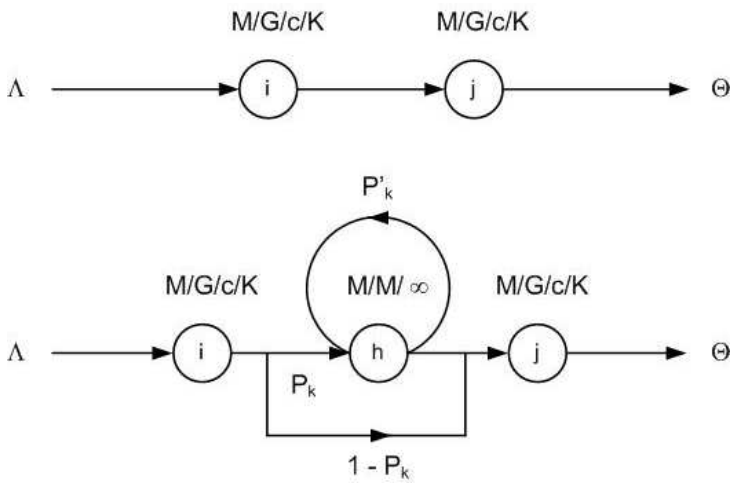


Figure 2.1: Expanded finite node j

p_K = blocking probability of finite queue of size K
p'_K = feedback blocking probability in the expansion method
h = the artificial holding node created in the GEM
c = number of servers
B_j = buffer capacity at node j *excluding* those in service
K_j = buffer capacity at node j *including* those in service
N = number of nodes in the network
$\rho = \lambda/(\mu c)$ = traffic intensity
θ = mean throughput rate

There are three main stages in the GEM:

1. Stage I: network reconfiguration
2. Stage II: parameter estimation
3. Stage III: feedback elimination.

After conducting these stages, the network is decomposed and each node can be evaluated separately.

2.1.1 Stage I: Network reconfiguration

For each finite node in the queueing network, an artificial node is created to register the blocked jobs. By introducing such artificial node, we also create new routing probabilities in the network. The result of network reconfiguration can be seen from figure 2.1. There are two possible states of the finite node, namely *saturated* and *unsaturated*. Arriving jobs will try to access the finite node j. With a probability of $(1 - p_K)$, the job will find the the finite node unsaturated, where it will enter the queue and eventually get serviced. However, if the finite node is saturated (with a probability of p_K), then the job will be directed to the artificial holding node h where it will get a delay. The delay at the artificial node is modeled using a $M/G/\infty$ queue, representing delay time without queueing. Afterwards, the blocked job will try to re-enter the finite queue with a success probability of $(1 - p'_K)$. There is a probability of p'_K that the blocked job still finds the finite node saturated and thus it will be directed again to the artificial node h. This process repeats until the blocked job is able to enter the finite node.

2.1.2 Stage II: Parameter estimation

At this stage, the values for parameters p_K, p'_K, and μ_h are determined.

- To determine p_K, exact analytical formulas should be used whenever possible (Kerbache and Smith (2000)). For cases where exact p_K formula is unavailable, approximations for p_K in $M/G/c/K$ setting provided by Smith (2003) can be used. These approximations are based on a closed-form expression derivable from the finite capacity exponential queue ($M/M/c/K$) using Kimura's two-moment approximation (Kimura (1996)). The following p_K formula for $M/G/2/K$ is presented as an example.

$$p_K = \frac{2\rho^{2((2+\sqrt{\rho/e}s^2-\sqrt{\rho/e}+B)/(2+\sqrt{\rho/e}s^2-\sqrt{\rho/e}))}(2\mu-\lambda)}{-2\rho^{(2(2+\sqrt{\rho/e}s^2-\sqrt{\rho/e}+B)/(2+\sqrt{\rho/e}s^2-\sqrt{\rho/e}))}\lambda+2\mu+\lambda}$$

- Since no exact method is available to calculate p'_K, the approximation based on diffusion technique by Labetuolle and Pujolle is used (Labetoulle and Pujolle (1980).

$$p'_K = [\frac{\mu_j+\mu_h}{\mu_h} - \frac{\lambda[(r_2^K-r_1^K)-(r_2^{K-1}-r_1^{K-1})]}{\mu_h[(r_2^{K+1}-r_1^{K+1})-(r_2^K-r_1^K)]}]^{-1}$$

  where r_1 and r_2 are the roots of the polynomial:

$$\lambda - (\lambda + \mu_h + \mu_j)x + \mu_h x^2 = 0,$$

  where, $\lambda = \lambda_j - \lambda_h(1-p'_K)$, and λ_j and λ_h are the actual arrival rates to the finite and artificial holding notes, respectively. Furthermore, it can be argued that:

$$\lambda_j = \tilde{\lambda_i}(1-p_K) = \tilde{\lambda_i} - \lambda_h.$$

- The delay distribution at the holding node h is actually nothing but the remaining service time of the finite node j. Based on the renewal theory, one can formulate the remaining service time distribution as the following rate μ_h:

$$\mu_h = \frac{2\mu_j}{1+\sigma_j^2\mu_j^2},$$

  where σ_j^2 is the service time variance of the finite node. At this point, one should notice that if the service time of the finite node is exponentially distributed with rate μ_j, then the memoryless property of exponential distribution will hold such that:

$$\mu_h = \mu_j$$

2.1.3 Stage III: Feedback elimination

As a result of the feedback loop at the holding node, strong dependency on the arrival process is created. In order to eliminate such dependency, the service rate at the holding node must be adjusted as follows:

$$\mu'_h = (1-p'_K)\mu_h$$

As a consequence, the service rate at node i preceding the finite node j is affected as well. One can see that the mean service time at node i is μ_i^{-1} when the finite node is unsaturated, and $\mu_i^{-1} + {\mu'_h}^{-1}$ when the finite node is saturated. Thus, on average, the mean service time of node i preceding the finite node j is:

$$\mu_i^{-1} = \mu_i^{-1} + p_K {\mu'_h}^{-1}$$

The above equations apply to all finite nodes in the queueing network. To sum up, all performance measures of the network can be obtained by solving the following equations simultaneously.

$$\lambda = \lambda_j - \lambda_h(1 - p'_K) \tag{2.1}$$

$$\lambda_j = \tilde{\lambda_i}(1 - p_K) \tag{2.2}$$

$$\lambda_j = \tilde{\lambda_i} - \lambda_h \tag{2.3}$$

$$\lambda_j = \tilde{\lambda_i} - \lambda_h \tag{2.4}$$

$$p'_K = [\frac{\mu_j + \mu_h}{\mu_h} - \frac{\lambda[(r_2^K - r_1^K) - (r_2^{K-1} - r_1^{K-1})]}{\mu_h[(r_2^{K+1} - r_1^{K+1}) - (r_2^K - r_1^K)]}]^{-1} \tag{2.5}$$

$$z = (\lambda + 2\mu_h)^2 - 4\lambda\mu_h \tag{2.6}$$

$$r_1 = \frac{[(\lambda + 2\mu_h) - z^{\frac{1}{2}}]}{2\mu_h} \tag{2.7}$$

$$r_2 = \frac{[(\lambda + 2\mu_h) + z^{\frac{1}{2}}]}{2\mu_h} \tag{2.8}$$

$$p_K = \frac{2\rho^{2((2+\sqrt{\rho/e}s^2-\sqrt{\rho/e}+B)/(2+\sqrt{\rho/e}s^2-\sqrt{\rho/e}))}(2\mu - \lambda)}{-2\rho^{(2(2+\sqrt{\rho/e}s^2-\sqrt{\rho/e}+B)/(2+\sqrt{\rho/e}s^2-\sqrt{\rho/e}))}\lambda + 2\mu + \lambda} \tag{2.9}$$

Note that the above expression of p_K only applies in an $M/G/2/K$ setting. Other expressions of p_K for $M/G/c/K$ queues with $c = 3$ to $c = 10$ have been developed by Smith (2003) and can be used in the above set of equations.

In short, there are three stages in the GEM: network configuration, parameter estimation, and feedback elimination. In all of these stages, no physical changes are made to the network. The expansion processes are embedded within the GEM algorithm.

2.2 Optimization tool: search methods

Now that we are able to approximate the throughput of a queueing network using the GEM, we use a search method to find the optimal allocation of buffers and servers. The following are some search methods that we use in combination with the GEM.

2.2.1 Complete enumeration

Perhaps the most classical search method for buffer and server optimization is the *complete enumeration*. As can be found in Hillier and So (1995), this search method basically enumerates all feasible buffer and server allocations. The throughput can then be calculated analytically or obtained using simulation. Eventually the optimal allocation that maximizes throughput is determined. One requirement of using this method is that the maximum number of buffers and servers must be given beforehand. As such, the main question is how many servers and buffers to allocate to each node in the network given a fixed number of total buffers and servers. This approach clearly has a serious limitation, namely the number of possible assignments of buffer and server increases significantly with increasing number of buffers, servers, and nodes. In particular, Hillier and So (1995) showed that the total number of possible way to allocate S servers and Q buffers to the $N-1$ nodes within a queueing network with N nodes is given by: $\begin{pmatrix} Q+N-2 \\ N-2 \end{pmatrix} \times \begin{pmatrix} S-1 \\ N-1 \end{pmatrix}$, assuming that the queue at the first node is infinite (that is, the first node is never starved). Within their assumptions, a relatively simple serial queueing network with 5 nodes, 4 buffer slots, and 10 servers, will incur an astonishing 4410 total number of possible buffer and server allocation. Given this condition, it is practically impossible to use complete enumeration to search the optimal allocation of buffers and servers for large, complex queueing networks (which is ultimately an NP-hard problem).

2.2.2 Powell and Coggin algorithm

Another search method that is often used in queueing network optimization is the *Powell algorithm*. This method has been successfully used in combination with the GEM particularly for buffer allocation problems (see *i.e.* Smith *et.al* (2006), Smith and Cruz (2005) and Smith and Daskalaki (1988)). As explained by Powell (1964), the Powell algorithm can be described as an unconstrained optimization procedure that does not require the calculation of first derivatives of the function. Numerical examples has shown that the method is capable of minimizing a function with up to twenty variables. The minimum of a non-linear function $f(\mathbf{x})$ can be found using this method by successive uni-dimensional searches starting from the best known approximation to the minimum $\mathbf{x}^k$ along a set of conjugate directions. The procedure generates these sets of conjugate directions and ensures that the rate of convergence to the minimum is satisfactory, even when the selected starting point is very poor. The set of procedures involved in the Powell algorithm is as follows:

1. Select a starting point and an initial set of direction vectors parallel to the coordinate axes $M_{i,j}$ $(i = 1, 2, ..., M;\ j = 1, 2, ..., M)$, where:

$$\begin{aligned} M_1 &= (1, 0, 0, ..., 0) \\ M_2 &= (0, 1, 0, ..., 0) \\ M_3 &= (0, 0, 1, ..., 0) \\ &... \\ M_M &= (0, 0, 0, ..., 1) \end{aligned}$$

2. Set up the Gauss-Newton equations:

$$(A^t A)\Delta\hat{A} = A^t(Y - \hat{Y}^*)$$

3. Solve the Gauss-Newton equations for $\Delta\hat{A}$.

4. Set up the new normalized direction vector from $\Delta\hat{A}$, using the following components.

$$M_{i,new} = \frac{\Delta\hat{A}_i}{[\sum_{j=1}^{M} \Delta\hat{A}_j^2]^{\frac{1}{2}}}$$

5. Perform a uni-dimensional search in the new direction.

6. Perform an overall convergence test when the one dimensional minimum has been found:

   (a) Convergence obtained: stop the procedure

   (b) Convergence not obtained: replace one of the previous direction vector with a new direction vector, replace elements in the Gauss Newton equation, and repeat the above procedure from 3.

To conduct the uni-dimensional search in Powell algorithm (as in step 5 above), the *Coggin algorithm* is often used as an option. The Coggin algorithm finds the unconstrained maximum of a single variable, non-linear function (Kuester and Mize (1973)). In a nutshell, this algorithm works as follows:

1. Select a starting point and evaluate the objective function.

2. Increment the independent variable with a distance ΔX and evaluate the objective function again. This is the first step in the search procedure. If the objective value is:

   (a) Improved: then double the step size for the next function evaluation.

   (b) Worsen: then reverse the direction of search and locate the next point with a distance of $-\Delta X$ from the initial starting point.

3. After the first step, the step size is doubled if the objective function is still improved, and halved if a worse objective function is obtained.

4. When a local optimum is obtained, evaluate three points (X_k, X_{k-1}, X_{k-2}) surrounding the optimum. An additional point $X_{k+1} = X_{k-1} + \frac{\Delta X}{2}$ is also located, where ΔX is the current step size. Keep the best three points as (X_1, X_2, X_3).

5. Fit a quadratic equation to the three points, where the optimum location X^* is obtained by setting $\frac{\partial f}{\partial X} = 0$

$$X^* = \frac{1}{2}[\frac{(X_2^2 - X_3^2)F(X_1) + (X_3^2 - X_1^2)F(X_2) + (X_1^2 - X_2^2)F(X_3)}{(X_2 - X_3)F(X_1) + (X_3 - X_1)F(X_2) + (X_1 - X_2)F(X_3)}] \tag{2.10}$$

6. Compare the objective function at X^* with the best of previous points subject to a convergence limit,

$$|X^* - X_i(\text{best})| \leq \text{limit}$$

   The procedure stops if the above criterion is satisfied. If not, replace the worst point with X^* and fit a new quadratic surface to obtain another local optimum. Repeat this procedure until convergence is obtained.

From the above elaboration, we can see how the Powell algorithm is coupled with the Coggin algorithm. The Powell algorithm is mainly used to choose the conjugate directions of the search by solving the Gauss-Newton equation, while the Coggin algorithm operationalizes the uni-dimensional search in the conjugate directions defined by the Powell Algorithm. It should be noted, however, that the Powell algorithm may find a local minimum instead of a global minimum. To overcome this problem, it is suggested that for a function with many variables, different initial starting points are used and see whether better extremes can be found. For more details on the two algorithms and their programming into FORTRAN, the readers are referred to Kuester and Mize (1973).

For more sources on heuristic search methods, the readers are referred to Glover and Greenberg (1989).

Chapter 3

Queueing network structure

3.1 Variables and topologies

As mentioned earlier, the existing studies about joint buffer-server optimization have been focusing mainly on serial production lines. In this study, we consider a more general queueing network structure by incorporating not only series, but split and merge topology as well. For each of these topologies, we use queueing networks with the number of nodes N of 3, 7, and 15.

In general, jobs arrive at an arrival rate of λ into the network. The value of λ in all experimental settings are restricted to 3, 7, and 15. It is possible to have more than one arrival node in the network. The jobs then flow according to the predefined routes. At each node i, there are c_i available servers that process the jobs, each with a processing rate of μ_i. For all settings of serial, split, and merge topologies, we will use a processing rate $\mu_i = 10,\ i = 1, 2, 3..., N$. The variability of the servers is reflected from their *coefficient of variation* (CV), which we set to either 0.5, 1.0, or 1.5 for each node. This way, we are able to evaluate general queueing networks $(M/G/c/K)$. There can also be K_i buffers at each node. Due to the finite buffer capacity, blocking may occur and some arrivals will be lost. In such cases, the resulting throughput θ will be lower than the arrival rate λ.

Note that we will use the following notations for buffers:
K_i = the number of buffers at each node i *including* the servers.
B_i = the number of buffers at each node i *excluding* the servers $(B_i = K_i - c_i)$.
These notations are important due to the fact that the servers themselves act at the same time as buffers. As such, we will treat B_i as the number of *pure* buffers in the network.

The series topology can be seen from figure 3.1 to 3.3, which show networks of 3, 7, and 15 nodes respectively. This topology has a simple flow structure where the finished jobs from one node are moved to the next node downstream. The routing probability from one node to another is simply 1.0.

The split topology can be seen from figure 3.4 to 3.6, which show networks of 3, 7, and 15

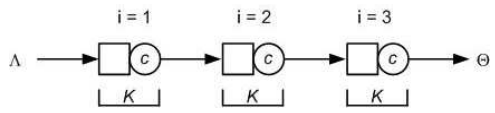


Figure 3.1: Series topology, $N = 3$

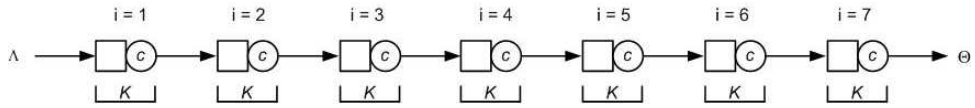


Figure 3.2: Series topology, $N = 7$

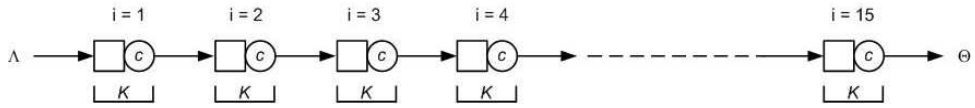


Figure 3.3: Series topology, $N = 15$

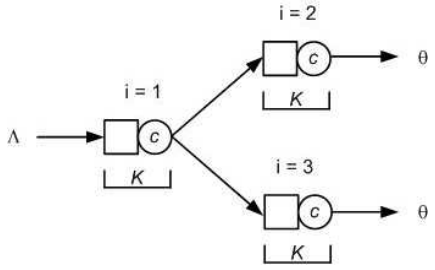


Figure 3.4: Split topology, $N = 3$

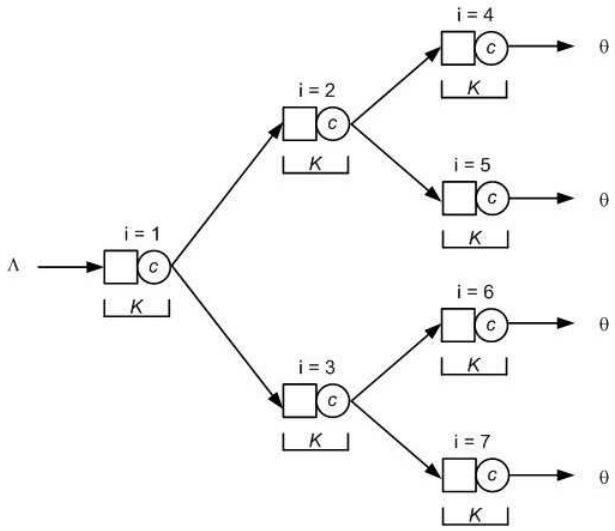


Figure 3.5: Split topology, $N = 7$

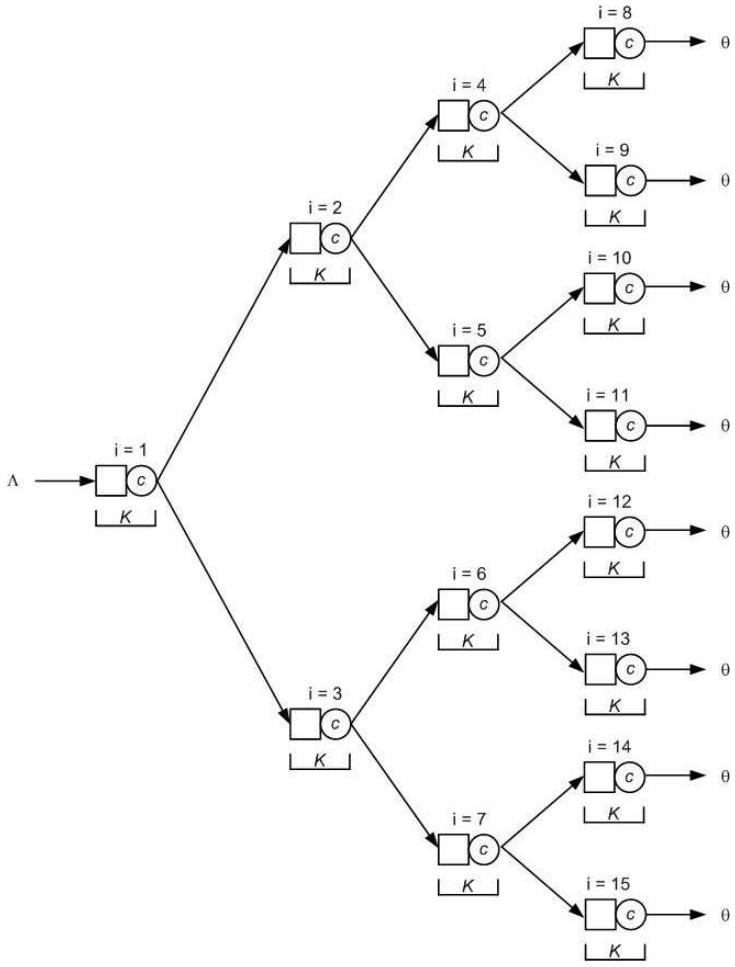


Figure 3.6: Split topology, $N = 15$

nodes respectively. This topology represents alternative routings to be chosen by the incoming arrival after being processed at a node. The likelihood of choosing a route is represented by the routing probability.

The merge topology can be seen from figure 3.7 to 3.9, which show networks of 3, 7, and 15 nodes respectively. This topology combines more than one incoming sources of arrival stream into a finite node.

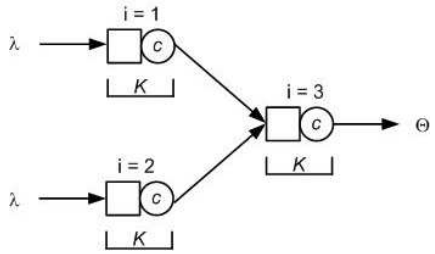


Figure 3.7: Merge topology, $N = 3$

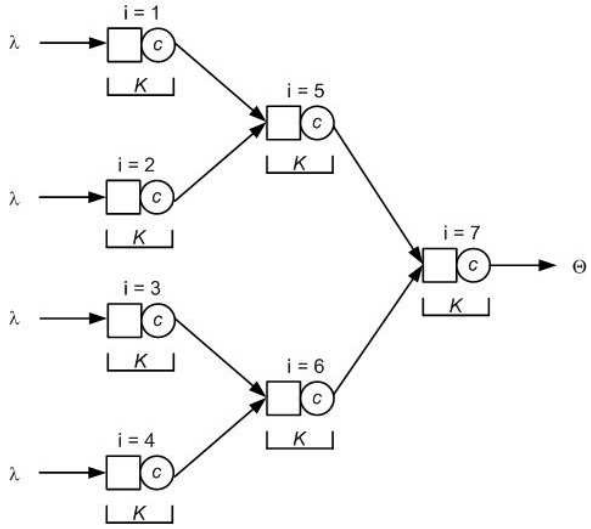


Figure 3.8: Merge topology, $N = 7$

In all of the figures from the three topologies, the position of each node i in the network is also depicted. These positions will be referred to in all experiments and analysis that involve such topologies.

3.2 Complete enumeration

As the structure of the queueing network becomes more complex (*e.g* with increasing number of nodes), the number of possible allocations of buffers and servers increases exponentially. As such, optimizing a complex queueing network by using complete enumeration is impossible. In this section, we will use complete enumeration to optimize a simple, small queueing network. The results will be used later to reflect on the quality of optimization using the BAP-CAP interaction and the single objective BCAP.

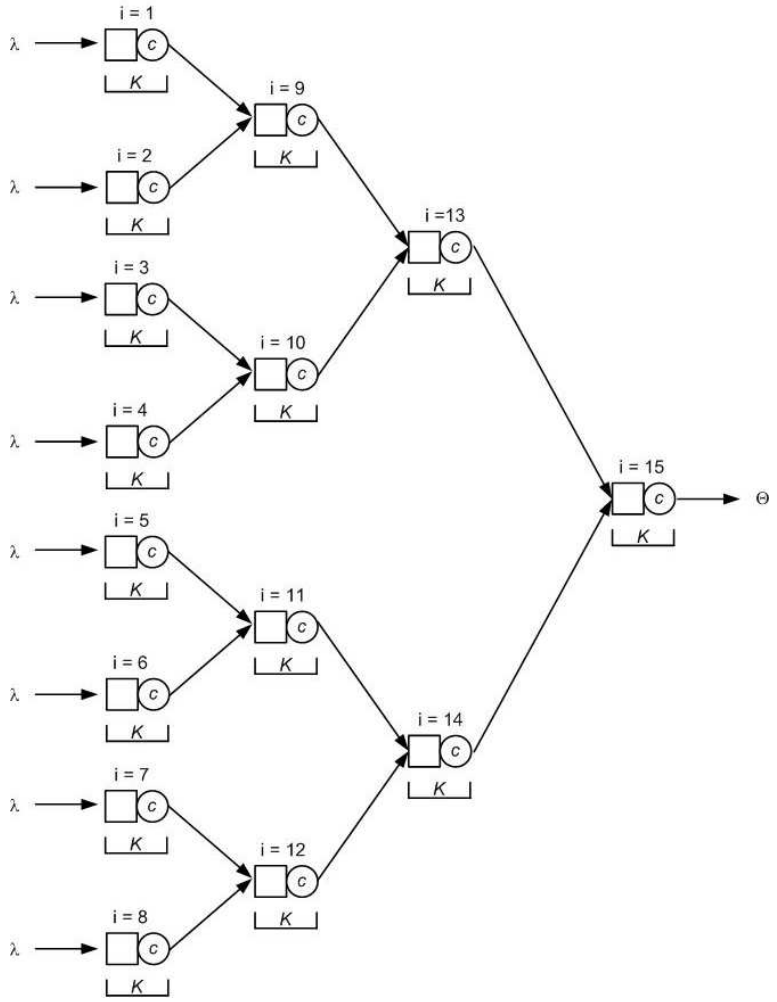


Figure 3.9: Merge topology, $N = 15$

The network that we will optimize is a 3-node series topology (see figure 3.10). We set the maximum number of servers as $\sum_i^N c = 6$ and the maximum number of buffers as $\sum_i^N K = 9$. The arrival rate λ is fixed at 3, the processing rate at each node μ_i is fixed at 10, and a CV of 1.0 is used. Though the network is very small, we identified 622 possible combinations of buffers and servers. Each of the possible combination is then evaluated using the GEM. The resulting throughput from settings with the same number of total buffers and servers are compared to seek for the optimal allocation, which gives the maximum throughput. The optimal allocations of buffers and servers are given in the table 3.1. Table C.1 to table C.7 in appendix C provides all enumeration results. This case example simply shows how tedious it might be to use the complete enumeration method to search for the optimal allocation of buffers and servers, even for a very simple queueing network structure.

Figure 3.10: Queueing network structure for the complete enumeration case

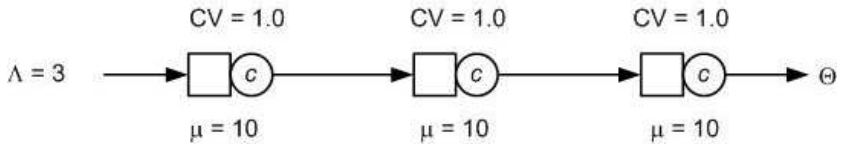


Table 3.1: Optimal buffer-server allocation using complete enumeration

c	K	θ	$\sum c$	$\sum K$
(1 1 1)	(1 1 1)	2.6350	3	3
(1 1 1)	(2 1 1)	2.7485	3	4
(1 1 1)	(2 1 2)	2.8156	3	5
(1 1 1)	(2 2 2)	2.8862	3	6
(1 1 1)	(3 2 2)	2.9234	3	7
(1 1 1)	(3 2 3)	2.9441	3	8
(1 1 1)	(3 3 3)	2.9652	3	9
(2 1 1)	(2 1 1)	2.7935	4	4
(2 1 1)	(2 1 2)	2.8635	4	5
(2 1 1)	(2 2 2)	2.9372	4	6
(2 1 1)	(2 2 3)	2.9582	4	7
(2 1 1)	(2 3 3)	2.9796	4	8
(2 1 1)	(2 3 4)	2.9859	4	9
(2 2 1)	(2 2 1)	2.8894	5	5
(2 1 2)	(2 2 2)	2.9657	5	6
(2 1 2)	(2 3 2)	2.9874	5	7
(2 2 1)	(2 2 4)	2.9938	5	8
(1 2 2)	(5 2 2)	2.9961	5	9
(2 2 2)	(2 2 2)	2.9954	6	6
(2 2 2)	(3 2 2)	2.9973	6	7
(2 2 2)	(3 2 3)	2.9983	6	8
(2 2 2)	(3 3 3)	2.9993	6	9

Chapter 4

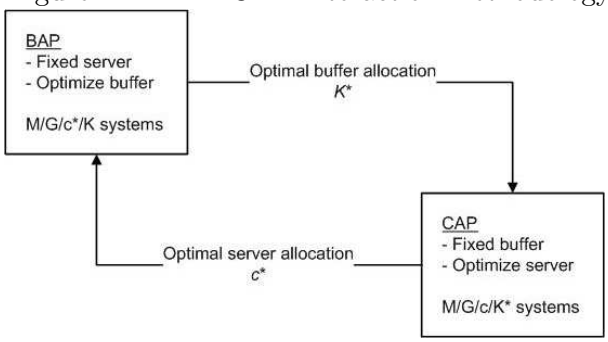
Method 1: BAP-CAP interaction

4.1 Methodology

In this first methodology, we interact two optimization studies that have been developed in the previous studies by Smith *et.al* (2006), namely BAP and CAP. Both of these studies used the GEM as the evaluation tool and the Powell algorithm in combination with Coggin algorithm as the optimization tool.

In the Buffer Allocation Problem (BAP), the number of servers is assumed to be fixed at each node in the queueing network. The BAP then concerns with finding optimal number and allocation of buffers for each node. On the other hand, the Server Allocation Problem (CAP) assumes a fixed number of buffers at each node and searches for the optimum number and allocation of servers that results in a throughput higher than a predefined threshold. We will use the output from the BAP (that is, the optimal buffer configuration) as the input for the CAP. As we set the number and allocation of buffers to the optimal buffer configuration according to BAP, we run the CAP to get the optimal server configuration. This process of using the output of BAP as input for the CAP and the other way around is run continuously until all possible combination of buffer and server allocation is obtained. Non-dominated solutions are then identified out of the dominated solutions. In both BAP and CAP, factors such as the queueing

Figure 4.1: BAP-CAP interaction methodology



network structure, arrival rate, processing rates of servers, and the coefficient of variation (CV) are used as inputs for the optimization software purposes. The following settings will be used in the experiments for BAP-CAP interaction.

1. Three topologies: series, split, and merge (refer to figure 3.1 to figure 3.9 for the structure of each topology and the position of each finite node in the network, respectively).

2. Three network structures: 3, 7, and 15 nodes

3. Three arrival rates: 3, 7, and 15 for CV of 1.0 (exponential arrivals)

4. Three CV's: hypoexponential (0.5), exponential (1.0), and hyperexponential (1.5) for an arrival rate of 15.

Note that we do not use all three possible CV values for each arrival rate, since the BAP-CAP interaction method is rather tedious to use. Using these settings, however, we are still able to capture a vast combination of buffer and server allocation from BAP-CAP interaction.

4.2 Experimental results

The complete experimental results from the BAP-CAP interaction are provided in the appendix table A.1 to table A.3. For the ease of analysis, we will reproduce here table A.1 into table 4.1 that represents the results from series, split and merge topologies for queueing networks with 3 nodes.

For a given starting point, the BAP-CAP interaction found several *non-dominated solutions*. The concept of non-dominated solutions can be seen from table 4.1. In the first row of the 3-node series topology, we see a solution of (2 2 2) for the server allocation and (3 3 3) for the buffer allocation with a resulting throughput of 2.9993. In the next row, the server allocation is (2 2 3) and the buffer allocation is (3 3 3) with a throughput of 3.000. Comparing these two solutions, we cannot say that the first solution is better than the second solution, or the other way around, because each solution has an advantage in terms of either the number of servers, buffers, or throughput. In the first solution, the total number of server is 6, which is one less than the second solution (7). This makes the first solution better in terms of number of servers, as it has fewer servers. However, the second solution has a higher throughput due to the fact that the number of server in this solution is higher. As such, the second solution is better in terms of throughput. These two results are called non-dominated. If one scrutinizes the results in table 4.1, one will see that all results are non-dominated. Defined formally, *non-dominated* solutions are solutions that are better than other solutions with regards to at least one objective, while not being worse in any other objectives than that other solutions. Recall from the introduction part that our objectives are to *maximize* the throughput, *minimize* the total number of servers, and *minimize* the total number of buffers. In general, it can be seen from the results in table 4.1 that settings with a high arrival rate ($\lambda = 15$) generate more non-dominated solutions than settings with a low arrival rate (*i.e.* $\lambda = 3$). As such, it is tedious to use this method to identify non-dominated solutions particularly for networks with relatively large arrival rates as compared to the processing rates. This is due to the fact that one has to manually sort the non-dominated solutions out of other dominated solutions.

In the 3-node series topology of table 4.1, we can see *i.e.* in rows 1, 2, 3, 6, 8, 9 and 10, that each node will be allocated equally with n buffers if the total number of available buffer K is a multiple integer (n) of the number of nodes N in the queueing network. In a 3-node series network, for example, when total number of buffers is 9, then each node is allocated with 3 buffers; when the total number of buffers is 27, then each node is allocated equally with 9 buffers, *etc.* However, if buffers cannot be divided equally throughout the nodes, then there will likely be an extreme allocation of buffers towards either the end or the beginning of the line. This extreme buffer allocation also depends on the server allocation, where in most cases the extreme buffer allocation is given to the single node where more servers are present than

Table 4.1: BAP-CAP interaction results, N=3

SERIES							
λ	CV	c	K	$\sum c$	$\sum K$	$\sum B$	θ
3	1.0	(2 2 2)	(3 3 3)	6	9	3	2.9993
		(2 2 3)	(3 3 3)	7	9	2	3.0000
7	1.0	(2 2 2)	(6 6 6)	6	18	12	7.0000
15	1.0	(3 2 2)	(23 10 10)	7	43	36	15.000
		(4 2 2)	(11 8 8)	8	27	19	15.000
		(3 4 4)	(7 7 7)	11	21	10	14.998
	0.5	(2 2 2)	(9 9 20)	6	38	32	15.000
		(2 2 3)	(9 9 9)	7	27	20	14.995
		(2 2 4)	(9 9 9)	8	27	19	15.000
	1.5	(2 2 3)	(11 11 11)	7	33	26	14.997
		(2 3 2)	(11 25 11)	7	47	40	15.000
SPLIT							
λ	CV	c	K	$\sum c$	$\sum K$	$\sum B$	θ
3	1.0	(2 2 1)	(3 2 3)	5	8	3	2.9990
		(2 2 2)	(3 2 2)	6	7	1	2.9998
		(2 2 2)	(3 2 3)	6	8	2	3.0000
		(2 2 3)	(3 2 2)	7	7	0	3.0000
7	1	(2 2 2)	(6 3 3)	6	12	6	7.0000
15	1	(2 2 2)	(7 4 7)	6	18	12	14.997
		(2 2 2)	(9 8 7)	6	24	18	14.999
		(2 2 3)	(7 4 4)	7	15	8	14.997
		(3 2 2)	(7 4 7)	7	18	11	15.000
		(4 4 2)	(7 4 4)	10	15	5	14.999
	0.5	(2 2 2)	(9 7 7)	6	23	17	14.998
		(2 2 3)	(6 6 4)	7	16	9	14.998
		(3 2 2)	(20 6 4)	7	30	23	15.000
		(2 4 2)	(6 6 4)	8	16	8	15.000
	1.5	(2 2 2)	(13 9 9)	6	31	25	14.999
		(3 2 2)	(7 5 5)	7	17	10	15.000
MERGE							
λ	CV	c	K	$\sum c$	$\sum K$	$\sum B$	θ
3	1	(2 2 2)	(2 2 3)	6	7	1	2.9998
		(2 2 3)	(2 2 3)	7	7	0	3.0000
7	1	(2 2 2)	(3 3 6)	6	12	6	6.9979
		(2 2 3)	(3 3 4)	7	10	3	6.9992
		(2 2 3)	(3 3 6)	7	12	5	7.0000
15	1	(2 2 2)	(7 7 23)	6	37	31	14.994
		(2 2 3)	(8 8 11)	7	27	20	14.996
		(2 2 3)	(7 7 23)	7	37	30	15.000
		(2 2 4)	(7 4 10)	8	21	13	15.000
	0.5	(2 2 2)	(6 6 20)	6	32	26	14.994
		(2 3 2)	(7 7 9)	7	23	16	15.000
	1.5	(2 2 2)	(8 8 25)	6	41	35	14.994
		(2 3 2)	(8 8 5)	7	21	14	15.000

any other nodes. One can see this allocation by looking at *i.e.* rows 4, 5, and 7. In row 4, for example, the total number of buffers ($\sum K$) is 43 and (23 10 10) buffers are allocated to the first, second, and third node, respectively. The first node is allocated with 23 buffers, and this first node also received the most servers, given that the server allocation is (3 2 2). This server and buffer allocation pattern is apparent regardless of the CV and the number of nodes in the queueing network (see also table A.2 and table A.3 in appendix A). For example, for a 7-node series network with $\lambda = 15$, $CV = 0.5$ (series topology of table A.2 in appendix A), one solution is a server allocation of (2 2 2 2 2 2 3) and a buffer allocation of (5 5 5 5 5 5 20). An exception of

this pattern exists in the last row of the 3-node series topology in table 4.1. The buffer allocation is (11 25 11), which means that the middle node receives the most buffers. However, the middle node is also allocated with more servers than the other two nodes, given the server allocation of (2 3 2). Despite this minor exception, it can be seen that generally either one end of the serial line is allocated significantly with the rest of available buffers and servers, if the number of allocated buffers and servers are not a multiple integer of the number of nodes in the series topology.

In the split topology, there is a tendency to imbalance the server and buffer allocation by giving more servers and buffers to the first splitting node in the network. This is due to the fact that the first splitting node has a significantly higher traffic as compared to the other nodes, as it processes the whole arrivals to the network. This tendency is particularly clear for settings with a relatively high arrival rate. In table 4.1 split topology, when $\lambda = 15$ and $CV = 1.0$, we get a buffer allocation of (7 4 7), (9 8 7), (7 4 4), all of which indicates a large buffer allocation to the first node. Higher CV also leads to a more obvious pattern, as we can see when $\lambda = 15$, $CV = 1.5$, the buffer allocation is (13 9 9) and (7 5 5). Both of which give preference to the first node (the splitting node). While we can see an extreme allocation of buffers to the splitting node, the server allocation is more homogenous. Only a few extra servers are allocated to the splitting node with the highest utilization and the rest of the nodes are allocated with an equal number of servers. See for example the server allocation of (3 2 2) when $\lambda = 15$ and $CV = 1.5$. This server and buffer allocation pattern is more apparent for larger networks, *i.e.* networks with $N = 7$ and $N = 15$. An example from a larger network (7 nodes) can be seen in table A.2 in appendix A, the split topology with $\lambda = 7$, $CV = 1.0$, which results in a server allocation of (2 2 2 1 1 1 1) and a buffer allocation of (6 3 3 3 3 3 3). Also, in table A.3 in appendix A, the split topology with 15 nodes, $\lambda = 15$, $CV = 1.0$ results in a server allocation of (2 2 2 2 2 2 2 1 1 1 1 1 1 1 2) and a buffer allocation of (10 7 7 3 3 3 3 4 4 4 4 4 4 4 2). These examples and other results in table A.2 and table A.3 of appendix A clearly indicates that the allocation pattern, namely more servers and buffers are allocated to the first splitting node in the network, is present mainly in large networks of 7 and 15 nodes. At this point, it is important to note that the results from the split topologies are obtained from setting a routing probability at each splitting node fixed at 0.5. As such, each route following the split node has the same arrival. One can intuitively argue that when the routing probability is higher to one route than another, then there will be more extra buffers and servers allocated to the route that has the highest routing probability because that particular route will also receive more arrivals (refer to the Routing Allocation Problem by *i.e.* Daskalaki and Smith (2004)).

In the merge topology, there is a tendency to imbalance the server and buffer allocation by giving more servers and buffers to the last merging node in the network. With a similar reasoning as in the split topology, this happens mainly due to the fact that the last merging node in the network receives all incoming arrivals in the network and thus has a significantly higher traffic. This imbalance of utilization along the nodes is likely to cause blocking to occur. To overcome this problem, the preference for allocating servers and buffers is given to the station that acts as the merging node with the highest utilization, that is, the last node in the network. This can be seen for example in table 4.1, the merge topology with $\lambda = 7$ and $CV = 1.0$, which gives server allocations of (2 2 2) and (2 2 3), with buffer allocations of (3 3 6) and (3 3 4). As the arrival rate is increased, the pattern becomes more apparent. See for example the case in table 4.1, the merge topology with $\lambda = 15$, $CV = 1.0$, results in server allocations of (2 2 2), (2 2 3), and (2 2 4), with buffer allocations of (7 7 23), (8 8 11), and (7 4 10). All of these results reflects the above mentioned pattern of server and buffer allocation. This pattern gets even stronger for larger networks (7-node and 15-node network). As can be seen from table A.2

in appendix A, a 7-node merge network with $\lambda = 15$ and $CV = 1.0$ results in a server allocation of (2 2 2 2 2 2 2) and a buffer allocation of (3 3 3 3 7 7 23). In a 15-node network with $\lambda = 15$ and $CV = 1.0$ (see table A.3 in appendix A), one solution is a server allocation of (2 2 2 2 2 2 2 2 2 2 2 2 2 2 2) and a buffer allocation of (2 2 2 2 2 2 2 2 3 3 3 3 7 7 23). All of these examples and other solutions indicates the tendency of giving more servers and buffers to the last merging node in the network. The buffer allocation, however, is significantly more extreme: large difference of allocated buffer is present between the last merging node in the network and the other nodes. This can partly be explained by the fact that the extra servers adds processing rate to the network and thus a slight increase in the number of servers is sufficient to reduce blocking, hence increasing the throughput. Additional buffers, however, do not have any effect on the processing rate of the overall network as they only provide waiting spaces for the finished jobs at one node before they are moved to the next node. As such, a large amount of buffers is needed before the same effect as from slightly adding the number of server can be obtained.

Finally, some solutions have less servers and/or buffers, but the resulting throughput is also smaller than that of solutions with more buffers and/or servers. Although these are clearly non-dominated solutions, one may argue that they are not necessarily optimal server and buffer allocations due to the nature of the optimization methodology used to generate these solutions. No literature in optimization can be found that justifies the use of interacting two separate optimization methods (in this case, the buffer allocation problem and the server allocation problem) to generate a combined optimization method (buffer-server allocation problem).

Chapter 5

Method 2: Single objective BCAP

5.1 Methodology

5.1.1 BCAP Experiments

In this second methodology, we optimize both servers and buffers simultaneously. The following optimization problem is used as the objective function.

$$Z = min(f(\mathbf{x},\mathbf{y}) = \sum_{i=1}^{N} x_i + \sum_{i=1}^{N} y_i) \tag{5.1}$$

subject to:

$$\theta(\mathbf{x},\mathbf{y}) \geq \theta^{\tau} \tag{5.2}$$

$$x_i \leq y_i \tag{5.3}$$

$$x_i, y_i \in 1, 2, 3..., \forall i \tag{5.4}$$

where:

x_i = number of servers at node i
y_i = number of buffers at node i
$\theta(\mathbf{x},\mathbf{y})$ = resulting throughput
θ^{τ} = threshold throughput

Using the above objective for the optimization problem, we basically seek for a minimum number of buffers and servers such that the resulting throughput is larger than a predefined threshold. Another constraint is that the number of buffers must be equal or larger than the number of servers. By buffer y_i we mean the buffers *including* servers (hence, $y_i \equiv K_i$). Therefore, it is impossible to have buffers less than servers and hence constraint 5.3 is added. We will denote the pure buffer as B_i, that is, the buffers *excluding* servers.

There are two hard constraints in the above formulation of optimization problem, namely the throughput constraint and the server-buffer constraint. These two constraints are the complicating constraints to the optimization problem. One way to ease the complexity is by incorporating the complicating constraints into the objective function via *Lagrangian relaxation*. A comprehensive overview about this method can be found in Lemarechal (2001) and Lemarechal (2003). The relaxed objective function can be formulated as follows.

$$Z_{\alpha} = min[\sum_{i=1}^{N} x_i + \sum_{i=1}^{N} y_i + \beta(\theta^{\tau} - \theta(\mathbf{x},\mathbf{y})) + \gamma \sum_{i=1}^{N}(x_i - y_i)] \tag{5.5}$$

subject to:

$$x_i, y_i \in 1, 2, 3..., \forall i \tag{5.6}$$

Note that in the above relaxed formulation of the objective function, the terms $\beta(\theta^\tau - \theta(\mathbf{x},\mathbf{y}))$ and $\gamma \sum(x_i - y_i)$ are always non-positive, if the constraints 5.2 and 5.3 are to be met. Since $Z_\alpha \leq Z$, we will use Z_α as a lower bound on the optimal objective value of Z and therefore we would like to have this lower bound as high as possible.

To solve this simultaneous server-buffer allocation problem, we will set the threshold throughput θ^τ equal to the incoming arrival rate, λ. The GEM will then approximate the resulting throughput, $\theta(\mathbf{x},\mathbf{y})$ given the particular server and buffer configuration. By setting the threshold throughput θ^τ equal to the arrival rate λ, the term $(\theta^\tau - \theta(\mathbf{x},\mathbf{y}))$ will become non-negative, as no server and buffer allocation will result in a throughput larger than the arrival rate into the network. Using this value, it can be seen from the equation 5.5 that the best solution would be to set servers and buffers to infinity to get the term:

$$(\theta^\tau - \theta(\mathbf{x},\mathbf{y})) = 0$$

Since this solution is not practical, we assume that a small difference is acceptable, such that $(\theta^\tau - \theta(\mathbf{x},\mathbf{y})) = \varepsilon$. An acceptable value of ε lies between [0,1] because otherwise it will be better to add one extra server or buffer to increase the throughput (refer to Cruz *et.al* (2006)). We note that $\theta(\mathbf{x},\mathbf{y})$ is a non-decreasing function of x and y since additional servers (x) will increase the processing rate and additional buffers (y) will provide waiting space, both of which reduce blocking probability and eventually increase throughput. If we assume an acceptable $(\theta^\tau - \theta(\mathbf{x},\mathbf{y})) \leq 10^{-3}$, then we can define a corresponding β as:

$$\beta \leq \frac{1}{(\theta^\tau - \theta(\mathbf{x},\mathbf{y}))}$$

which yields $\beta \geq 10^3$ for $(\theta^\tau - \theta(\mathbf{x},\mathbf{y})) \leq 10^{-3}$.

As for the value of γ, we choose a large enough number (*i.e.* 100) relative to the overall objective value to make sure that the Powell and Coggin algorithm will not allocate more servers than buffers at each node in the network. This way, the two algorithms will avoid getting a high penalty of $\gamma[\sum_{i=1}^{N}(x_i - y_i)]$ by fulfilling the constraint 5.3 when searching for the optimal number of buffers and servers to be allocated.

The GEM will be used in conjunction with Powell's algorithm and the Coggin algorithm as the search method. These algorithms have been successfully coupled with the GEM in previous studies, *i.e.* by Cruz *et.al* (2007), Cruz *et.al* (2006), Smith and Cruz (2005). The search method will look for server and buffer allocations that results in the minimum total number of buffers and servers while ensuring the throughput to be as close as possible to the arrival rate and that the number of servers will not exceed the number of buffers.

5.1.2 Different pricing for servers and buffers

As one might notice, in both the original objective function and the relaxed objective function we use an equal weight for server and buffers (see equation 5.1 and 5.5). As such, we treat servers and buffers only as a physical object while disregarding the fact that the price for a server is likely to be different with that of a buffer. To bring the analysis one step further, we consider next a different price for servers and buffers to be used in the relaxed objective

function. The new relaxed objective function with different weights for the server and buffer is formulated as follows.

$$Z_\alpha = min[\alpha \sum_{i=1}^{N} x_i + (2-\alpha) \sum_{i=1}^{N} y_i + \beta(\theta^\tau - \theta(\mathbf{x},\mathbf{y})) + \gamma \sum_{i=1}^{N} (x_i - y_i)] \tag{5.7}$$

subject to:

$$x_i, y_i \in 1, 2, 3..., \forall i \tag{5.8}$$

In this modified objective function, we assign a weight of α to servers and $(2-\alpha)$ to buffers. We will then modify the value of α such that $0 < \alpha < 2$ to reflect the relative weights of servers to buffers. We set $\alpha = 1$ as the default value, which will result in the exact same objective function as in equation 5.5. As α is decreased below 1.0, then the value of servers will become less than that of buffers. That is, using α less than 1.0 we assume that buffers are more expensive than servers. On the contrary, when the value of α is increased above 1.0, then the servers gain more weight than the buffers and therefore the servers become more expensive than the buffers. In this way, we can see whether the different pricing of servers and buffers will result in a significantly different optimal allocations.

5.1.3 Quality of optimization solutions

To evaluate the quality of solutions generated by the single objective BCAP, we will compare the allocation results with the allocations via BAP and CAP codes. Recall that in the single objective BCAP we obtain a simultaneous allocation of servers and buffers given a specific queueing network structure. BAP and CAP, on the contrary, assume a fixed allocation of server to search for the optimum buffer allocation (for BAP), and a fixed allocation of buffer to search for the optimum server allocation (for CAP). As such, we first look at the optimum server and buffer allocation by the single objective BCAP. Afterwards, we use the server allocation from the single objective BCAP as the input for BAP to get the optimum buffer allocation according to this method. In the similar way, we will use the optimum buffer allocation from the single objective BCAP as the input for CAP to get the optimum server allocation according to this method. By comparing the results from BCAP, BAP, and CAP, we can get some insight about the quality of optimal server and buffer allocation using the single objective BCAP.

Next to the above procedures, we will also compare the optimum allocation of buffers and servers from the single objective BCAP with the results from the complete enumeration in chapter 3 (see table 3.1. Our purpose is to see whether the optimal solution obtained using the single objective BCAP is indeed the optimal solution given other possible allocations of buffers and servers.

5.2 Experimental results

5.2.1 BCAP results

The complete experimental results from the single objective BCAP are provided in the appendix table B.1 to table B.3. For presentation, we will reproduce here table B.2 into table 5.1 that represents the results from series, split and merge topologies for queueing networks with 7 nodes. Recall that K represents the number of buffers *including* servers, while B represents the number of pure buffers (that is, buffers *excluding* servers ($B = K - c$)).

The single objective BCAP identifies *one* optimal server and buffer allocations for a given

Table 5.1: Single objective BCAP results, N=7

SERIES							
λ	CV	c	K	$\sum c$	$\sum K$	$\sum B$	θ
3	0.5	(2 2 2 2 2 2 2)	(2 2 2 2 2 3 5)	14	18	4	3.0000
	1	(2 2 2 2 2 2 2)	(4 2 2 2 2 2 2)	14	16	2	2.9998
	1.5	(2 2 2 2 2 2 2)	(4 2 2 2 2 2 2)	14	16	2	2.9997
7	0.5	(2 2 2 2 2 2 2)	(7 2 2 7 2 2 3)	14	25	11	6.9988
	1	(2 2 2 2 2 2 2)	(8 2 2 3 2 2 6)	14	25	11	6.9992
	1.5	(3 3 3 3 3 3 3)	(3 3 3 3 3 3 3)	21	21	0	6.9866
15	0.5	(2 2 2 2 2 2 4)	(2 2 2 2 2 2 5)	16	17	1	14.936
	1	(3 2 2 2 2 2 2)	(3 3 2 2 2 2 3)	15	17	2	15.000
	1.5	(3 2 2 2 2 2 2)	(3 2 2 2 2 2 3)	15	16	1	15.000
SPLIT							
λ	CV	c	K	$\sum c$	$\sum K$	$\sum B$	θ
3	0.5	(2 2 2 1 1 1 1)	(2 2 2 2 2 2 2)	10	14	4	2.9984
	1	(2 2 2 1 1 1 1)	(2 2 2 2 2 2 2)	10	14	4	2.9976
	1.5	(2 2 2 1 1 1 1)	(2 2 2 2 2 2 2)	10	14	4	2.9967
7	0.5	(3 2 2 2 2 2 2)	(3 2 2 2 2 2 2)	15	15	0	6.9989
	1	(3 2 2 2 2 2 2)	(3 2 2 2 2 2 2)	15	15	0	6.9986
	1.5	(3 2 2 2 2 2 2)	(3 2 2 2 2 2 2)	15	15	0	6.9982
15	0.5	(4 2 2 2 2 2 2)	(7 2 2 2 2 2 2)	16	19	3	14.999
	1	(5 2 2 2 2 2 2)	(5 2 2 2 2 2 2)	17	17	0	15.000
	1.5	(4 2 2 2 2 2 2)	(4 2 2 2 2 2 3)	16	17	1	14.996
MERGE							
λ	CV	c	K	$\sum c$	$\sum K$	$\sum B$	θ
3	0.5	(1 1 1 1 2 2 2)	(2 2 2 2 2 2 2)	10	14	4	2.9994
	1	(1 1 1 1 2 2 2)	(2 2 2 2 2 2 2)	10	14	4	2.9991
	1.5	(1 1 1 1 2 2 2)	(2 2 2 2 2 2 2)	10	14	4	2.9988
7	0.5	(2 2 2 2 2 2 3)	(2 2 2 2 2 2 3)	15	15	0	6.9990
	1	(2 2 2 2 2 2 3)	(2 2 2 2 2 2 3)	15	15	0	6.9988
	1.5	(2 2 2 2 2 2 3)	(2 2 2 2 2 2 3)	15	15	0	6.9985
15	0.5	(2 2 2 2 2 2 5)	(3 3 2 4 2 2 5)	17	21	4	14.999
	1	(3 2 2 2 3 3 4)	(3 3 3 5 3 3 4)	19	24	5	15.000
	1.5	(3 2 2 2 2 2 5)	(3 3 3 5 2 2 5)	18	23	5	14.999

queueing network structure and a set of starting search points. Starting search points have a significant effect on the resulting server and buffer allocation, where different starting search points will likely lead to different optimal server and buffer configurations. This is due to the fact that there is no single solution to the optimal server and buffer configuration, as we embedded the three original objectives (minimizing the total number of buffers and servers, and maximizing the throughput) into one objective function. Most of the results in table B.1, B.2, and B.3 are obtained using small starting points for servers and buffers (*i.e.* 2).

In the series topology, servers are in most cases allocated equally throughout the network. We can see from table 5.1, for example, the 7-node series topology with $\lambda = 3$ results in a server allocation of (2 2 2 2 2 2 2) regardless of the CV. Also, when $\lambda = 7$, the server allocations are (2 2 2 2 2 2 2) and (3 3 3 3 3 3 3). These results show a perfectly balanced allocation of servers among the nodes. However, as the arrival rate is increased further to $\lambda = 15$, the optimal allocation of servers is not balanced anymore. Either the start or the end node is given more servers than the other nodes, as can be seen from the server allocation of (2 2 2 2 2 2 4), (3 2 2 2 2 2 2), and (3 2 2 2 2 2 2). This optimal server allocation pattern is consistent regardless of

the network size. We can see that in a 3-node serial network, the servers are equally allocated at each node with (2 2 2) for $\lambda = 3$ and $\lambda = 7$ (refer to table B.1, rows 1-6 in appendix B). In a 15-node serial network, the servers are also equally allocated at each node with (2 2 2 2 2 2 2 2 2 2 2 2 2 2 2) for $\lambda = 3$ and $\lambda = 7$ (refer to table B.3, rows 1-6 in appendix B).
Buffer allocation pattern, on the contrary, is more unpredictable. In most cases, the buffers are not allocated equally to each node. Nodes located near or at the starting/end of the line tend to receive more buffers. In table 5.1, for example, the buffer allocation for series topology with $\lambda = 3$ is imbalanced towards either end of the line: (2 2 2 2 2 3 5) and (4 2 2 2 2 2 2). This buffer allocation pattern is however less consistent than the server allocation pattern.
Note that our setting is different from the setting of the serial production line by Hillier and So (1995) and Spinellis *et.al* (2000)in at least two dimensions. In our series network, all nodes are finite nodes, including the first node in the network. Also, we assume that the arrivals that find a full queue at the first node will be lost. These two assumptions are different with those of the previous studies mentioned earlier. In particular, they assumed that the first node in the serial production line has a infinite buffers and thus no arrivals are lost. These differences eventually lead to different pattern of buffer-server allocation. Our results suggest an extra allocation of servers and/or buffers to either the first or the end nodes, while Hillier and So (1995) and Spinellis *et.al* (2000) suggest to allocate any extra servers and/or buffers to the middle node.

In the split topology, more servers are allocated to the splitting node than the other nodes. The first splitting node is likely to be allocated with more servers as this node handles the most arrivals compared to any other nodes in the network. The results of the 7-node split topology in table 5.1 clearly show this pattern. For $\lambda = 3$, the optimal server allocation is (2 2 2 1 1 1 1) regardless of the CV; for $\lambda = 7$, the optimal server allocation is (3 2 2 2 2 2 2) regardless of CV; and for $\lambda = 15$, the optimal server allocations are (4 2 2 2 2 2 2) for $CV = 0.5$ and $CV = 1.5$, and (5 2 2 2 2 2 2) for $CV = 1.0$. All of these results have the tendency of allocating more servers toward the first node, which is the splitting node that receives the highest arrival rate in the network. This pattern also exists in the large 15-node network (see the 15-node split topology in table B.3, $\lambda = 3,\ 5$, and 7). The pattern is less obvious in the small 3-node split network with small arrival rate (*i.e.* $\lambda = 3$). One explanation would be there is no need to add an extra server to the splitting node if the number of servers are already sufficient to handle the high arrival at the first splitting node. Observe, for example, that the 3-node split topology in table B.1 with $\lambda = 3$ results in an optimal server allocation of (2 2 2) regardless of the CV. However, as the arrival rate is set to $\lambda = 15$, there is a need to add an extra server at the splitting node, which results in an optimal server allocation of (3 2 2), regardless of CV.
Regarding the buffer allocation, it is interesting to see that, in most cases, nodes that already received extra servers are not allocated with extra buffers. For example, in the split topology results of table 5.1 with $\lambda = 3$, the optimal server allocation is (2 2 2 1 1 1 1) and the optimal buffer allocation is (2 2 2 2 2 2 2). As such, the first three nodes in this network are practically not allocated with buffers. That is, the number of servers (c) and buffers (K) in these nodes are equal, indicating a zero-buffer node (where $B_i = K_i - c_i = 0$). Buffers are allocated to the nodes with fewer servers (in the above case, nodes 4, 5, 6, and 7) and the priority is given to splitting nodes. This pattern, however, is more apparent in large networks (*i.e.* $N = 7$ and $N = 15$). If one scrutinizes the 15-node split topology in table B.3 with $\lambda = 7$, one will see that the optimal server allocation is (2 2 2 2 2 2 2 1 1 1 1 1 1 1 2) and the optimal buffer allocation is (2 2 2 2 2 2 2 2 2 2 2 2 2 2 2), regardless of the CV. This indicates that the first 7 splitting nodes in this network (node 1 to node 7) are not given with buffers. Buffers are only allocated to the 8th until the 14th nodes. In the small 3-node network, the number of servers are already sufficient to handle the arrivals and therefore zero-buffer networks are suggested as an optimal

configuration particularly for $\lambda = 3$ and $\lambda = 15$. For the small 3-node network with $\lambda = 7$, however, the optimization algorithm fails to find an optimal zero-buffer configuration. Also, the exact same server and buffer allocations will result in higher throughput for systems with lower CV's, which is intuitively correct. See, for example, the 7-node split topology in table 5.1 with $\lambda = 3$. The resulting throughput for the same buffer and server configuration is decreasing from 2.9984, 2.9976, to 2.9967, for the settings with $CV = 0.5, 1.0$, and 1.5, respectively

In the merge topology, more servers are allocated to the merging node than the other nodes. The last merging node is likely to be allocated with more servers as this node handles the most arrivals compared to any other nodes in the network. For example, in the 7-node merge topologies of table 5.1, the optimal server allocation is (1 1 1 1 2 2 2) for $\lambda = 3$, (2 2 2 2 2 2 3) for $\lambda = 7$, and (2 2 2 2 2 2 5), (3 2 2 2 3 3 4), or (3 2 2 2 2 2 5) for $\lambda = 15$. All of these results consistently suggest a higher number of server to be allocated at the last (merging) node. This pattern also exist in the 15-node large network. For this large network, one can see from table B.3 in appendix B that the server allocation is (1 1 1 1 1 1 1 1 1 1 1 2 2 2) for $\lambda = 3$, (1 1 1 1 1 1 1 1 2 2 2 2 2 2 2) for $\lambda = 7$, and (2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 4) or (2 2 2 2 2 2 2 2 2 2 2 2 2 2 5) for $\lambda = 15$. However, this server allocation pattern is less obvious for the 3-node small network. Only when the arrival rate is set relatively high (*i.e.* $\lambda = 15$) will this pattern emerges. In table B.1 of appendix B, the 3-node merge topology with $\lambda = 15$ results in a server allocation of (2 2 4) and (2 2 5).
We also found a similar buffer allocation pattern with that of split topology, namely nodes that already received extra servers are not allocated with extra buffers. This buffer allocation pattern is particularly apparent for networks with low and medium arrival rate. This can be seen in table 5.1, the 7-node merge topology with $\lambda = 3$ results in a server allocation of (1 1 1 1 2 2 2) and a buffer allocation of (2 2 2 2 2 2 2), regardless of CV. This means that the buffers are only allocated to the first 4 nodes, which have less servers than the last 3 nodes. As we use $\lambda = 7$, we get a zero-buffer network as an optimal configuration, where the server and buffer allocations are both (2 2 2 2 2 2 3). This buffer allocation pattern also applies to the 15-node large network. In table B.3 of appendix B, the 15-node merge topology with $\lambda = 3$ results in an optimal server allocation of (1 1 1 1 1 1 1 1 1 1 1 1 2 2 2) and an optimal buffer allocation of (1 1 1 1 1 1 1 1 2 2 2 2 2 2 2), regardless of CV. For $\lambda = 7$, the optimal server allocation is (1 1 1 1 1 1 1 1 2 2 2 2 2 2 2) and the optimal buffer allocation is (2 2 2 2 2 2 2 2 2 2 2 2 2 2 2). Both of these results indeed suggest that the extra buffers (if any) are allocated to the nodes with no extra servers (fewer number of servers). As a final note, networks with higher arrival rate and CV are in most cases allocated with more servers and/or buffers. This can be seen from the columns $\sum c$ and $\sum K$ in table 5.1, for example.

In general, the single objective BCAP generates many zero-buffer systems, where the only buffers available in the network are the servers themselves. This can be seen from column $\sum B$, where zero-buffer systems are indicated with $\sum B = 0$, that is, no extra buffers other than the servers. Zero-buffer systems are found to be optimal allocations particularly for small networks ($N = 3$) regardless of the network structure. For more insight about the performance evaluation of zero-buffer systems and the optimization of such systems, readers are referred to the paper by Andriansyah *et.al* (2006).

From the previous observations on server and buffer allocation patterns for series, split, and merge topologies, we can argue that the server allocation pattern is more consistent across different network sizes, arrival rates, and coefficient of variations. The buffer allocation pattern is generally less straightforward and often missing in different queueing network structure.

5.2.2 Different pricing for servers and buffers

The complete experimental results from single objective BCAP with different pricing of servers and buffers are provided in the appendix table B.4 to table B.6. For the ease of analysis, we will reproduce here table B.5 into table 5.2 that represents the results from series, split and merge topologies for queueing networks with 7 nodes. The $c - K$ price ratio indicates the relative weights of server as compared to buffer. A price ratio of 10/1, for example, means that servers are 10 times more expensive than buffers.

Table 5.2: Different server-buffer price ratio, N=7

SERIES	$c-K$ **price ratio**		c	K	$\sum c$	$\sum K$	$\sum B$	θ
	c	K						
λ **= 7**	1	1	(2 2 2 2 2 2 2)	(8 2 2 3 2 2 6)	14	25	11	6.9992
CV**=1.0**	2	1	(2 2 2 2 2 2 2)	(8 2 2 3 2 2 6)	14	25	11	6.9992
	3	1	(2 2 2 2 2 2 2)	(8 2 2 3 2 2 7)	14	26	12	6.9997
	4	1	(2 2 2 2 2 2 2)	(9 2 2 3 2 2 7)	14	27	13	6.9997
	10	1	(2 2 2 2 2 2 2)	(9 2 2 3 2 2 8)	14	28	14	6.9997
	1	2	(3 2 2 2 2 2 2)	(5 2 2 2 2 2 2)	15	17	2	6.9994
	1	3	(4 2 2 2 2 2 2)	(5 2 2 2 2 3 3)	16	19	3	6.9998
	1	4	(4 2 2 2 2 2 2)	(4 2 2 2 2 2 2)	16	16	0	6.9998
	1	10	(4 2 2 2 2 2 2)	(4 2 2 2 2 2 2)	16	16	0	6.9998
SPLIT	$c-K$ **price ratio**		c	K	$\sum c$	$\sum K$	$\sum B$	θ
	c	K						
λ**= 15**	1	1	(5 2 2 2 2 2 2)	(5 2 2 2 2 2 2)	17	17	0	14.998
CV**=1.0**	2	1	(4 2 2 2 2 2 2)	(8 2 2 2 2 2 2)	16	20	4	14.999
	3	1	(4 2 2 2 2 2 2)	(8 2 2 2 2 2 2)	16	20	4	14.999
	4	1	(4 2 2 2 2 2 2)	(8 2 2 2 2 2 2)	16	20	4	14.999
	10	1	(4 2 2 2 2 2 2)	(9 2 2 2 2 2 2)	16	21	5	14.946
	1	2	(5 2 2 2 2 2 2)	(5 2 2 2 2 2 2)	17	17	0	14.999
	1	3	(5 2 2 2 2 2 2)	(5 2 2 2 2 2 2)	17	17	0	14.999
	1	4	(5 2 2 2 2 2 2)	(5 2 2 2 2 2 2)	17	17	0	14.999
	1	10	(4 2 2 3 3 3 3)	(5 2 2 3 3 3 3)	20	21	1	15.000
MERGE	$c-K$ **price ratio**		c	K	$\sum c$	$\sum K$	$\sum B$	θ
	c	K						
λ **= 3**	1	1	(1 1 1 1 2 2 2)	(2 2 2 2 2 2 2)	10	14	4	2.9991
CV**=1.0**	2	1	(1 1 1 1 2 2 1)	(2 2 2 2 2 2 5)	9	17	8	2.9984
	3	1	(1 1 1 1 1 1 1)	(2 2 2 2 3 3 6)	7	20	13	2.9984
	4	1	(1 1 1 1 1 1 1)	(2 2 2 2 3 3 6)	7	20	13	2.9984
	10	1	(1 1 1 1 1 1 1)	(2 2 2 2 4 4 6)	7	22	15	2.9992
	1	2	(1 1 1 1 2 2 2)	(2 2 2 2 2 2 2)	10	14	4	2.9991
	1	3	(1 1 1 1 2 2 2)	(2 2 2 2 2 2 2)	10	14	4	2.9991
	1	4	(1 1 1 1 2 2 2)	(2 2 2 2 2 2 2)	10	14	4	2.9991
	1	10	(2 1 1 1 2 2 2)	(2 2 2 2 2 2 2)	11	14	3	2.9994

As the price of server is increased and the price of buffers is reduced, the optimal allocation shows that *there might be a slight decrease in the number of servers and the number of buffers will increase considerably* as compared to the optimal allocation for the case of equal price. See for example the split and merge topologies in table 5.2. When the servers is twice more expensive than buffers (that is, the c/K price ratio is 2/1), the server allocation is reduced from (5 2 2 2 2 2 2) to (4 2 2 2 2 2 2) in the split topology. For the merge topology, the server allocation is reduced from (1 1 1 1 2 2 2) to (1 1 1 1 2 2 1). The reduction of server allocation is apparent in all topologies, network sizes and arrivals, except for the 7-node series topology (see table 5.2) where the server allocation stays at (2 2 2 2 2 2 2) regardless of the server/buffer price ratio. In the small 3-node series network, for example, the server allocation drops from (2 2 2)

to (1 1 1) as the server price is set twice of the buffer price. The reduction of servers happens due to the fact that the price of servers are becoming more expensive and hence it is better to reduce the servers and add more buffers to the network. However, reducing the number of servers affect the throughput significantly due to the large reduction of the processing rates and therefore many buffers are needed to compensate for even a slight decrease in the number of servers. This result indicates that the higher penalty from having a lower throughput (hence higher value of $(\theta^{\tau} - \theta(\mathbf{x},\mathbf{y}))$) is justified by the lower overall objective value even though the total number of buffers is increased considerably.
We also observed that the optimal allocation starts changing when the servers are set to be twice more expensive than the buffers (thus a server/buffer price ratio of 2/1). From the 7-node split and merge networks of table 5.2, we can see that the server allocation does not change from (4 2 2 2 2 2 2) for the split topology, and the server allocation changed slightly to (1 1 1 1 1 1 1) for the merge topology, as the server/buffer price ratio is increased beyond 2/1. In most cases, the optimal allocation does not change much from this point even when the servers are set up to four times more expensive than the buffers (price ratio 4/1). This is particularly true for the small 3-node networks, regardless of the topology. We can see from the 3-node network of table B.4 in appendix B, that the server allocation stays at (1 1 1), (2 1 1), and (2 2 4) for the series, split and merge topologies, respectively, when the server/buffer price ratio is increased up to 10/1. This finding may indicate that there is no significant increase in the throughput when the buffer and server allocations are changed due to the different pricing. The objective value remains similar and therefore the previous optimal allocation is retained. When the servers are extremely more expensive than the buffers (*i.e.* 10 times more expensive), then the optimal allocation shows a further slight change in buffer allocation, but rarely in server allocation. See for example the 7-node network in table 5.2, the buffer allocation increased from (9 2 2 3 2 2 7) to (9 2 2 3 2 2 8) for the series topology, from (8 2 2 2 2 2 2) to (9 2 2 2 2 2 2) for the split topology, and from (2 2 2 2 3 3 6) to (2 2 2 2 4 4 6) for the merge topology, as the servers are 10 times more expensive than the buffers. The same pattern also takes place in the small 3-node network (see table B.4). It is interesting to note that the number of servers is never reduced too much even in cases where servers are set to be extremely expensive. For example, in the 7-node network of table 5.2, the total servers stays at 14 for the series topology even when the servers are 10 times more expensive than buffers. One explanation would be that servers are needed in sufficient quantity to guarantee an acceptable throughput and therefore they cannot be decreased in high extent. The largest reduction of servers under an extreme price ratio happens in the large 15-node merge network (see table B.6). Compared to equal price setting, the total allocated servers is reduced from 22 to 16, while the total allocated buffers is increased from 30 to 62.

As the price of servers is reduced and the price of buffers is increased, the optimal allocation shows that *there is a notable increase in the number of servers and a considerable decrease in the number of buffers* as compared to the optimal allocation for the case of equal price. See for example the 7-node network in table 5.2. The server allocation for the 7-node series topology incresed from (2 2 2 2 2 2 2) to (3 2 2 2 2 2 2) when the server price is set to be half of the buffer price. The server allocation further increases to (4 2 2 2 2 2 2) as the server price is further reduced. However, this pattern is missing for the 7-node split topology, and also less obvious for the 7-node merge topology (see the split and merge results of table 5.2, when the server/buffer price ratio is set to 1/2 to 1/10). Also, for cases where the optimal allocation is already a zero-buffer system, changes in optimal allocation will not likely to happen. This is the case for the 7-node split topology in table 5.2, where the server allocation stay at (5 2 2 2 2 2 2) as the server price is reduced up to 4 times of the buffer price. In general, however, there is no clear pattern about at which price ratio the optimal allocation policy will start changing

as some optimal allocation changes early at server/buffer price ratio of 1/2 (as in the 7-node series topology 5.2) and some other changes later at server/buffer price ratio of 1/10 (as in the 7-node merge topology, table 5.2).
The increase of buffer price relative to the server price also causes the total number of buffers to reduce considerably, as can be seen from the column $\sum B$ in table 5.2. In the 7-node series network of table 5.2, the total pure buffers B is reduced from 11 to 2 when the buffers are twice more expensive than the servers, and the network becomes a zero-buffer system as the buffers are set 4 times more expensive than the servers. This pattern applies particularly for the small 3-node network (see table B.4 in appendix B) and less obvious for the large 15-node network (see table B.6 in appendix B).

5.2.3 Quality of optimization solutions

The complete experimental results from the comparison of single objective BCAP, BAP and CAP are provided in the appendix table B.7 to table B.9. For the ease of analysis, we will reproduce here table B.8 into table 5.3 that represents the results from series, split and merge topologies for queueing networks with 7 nodes.

In general, the single objective BCAP is able to find optimal solutions that allocate less servers and/or buffers to the queueing network as compared to both BAP and CAP. Only in some cases does CAP performs better than the single objective BCAP, that is, allocating less server and getting higher or equal throughput than the single objective BCAP.

The single objective BCAP can be argued to be more robust than the other two optimization methods. In a sense, the single objective BCAP never allocates excessive buffers, which happens for the 7-node series network with the BAP methodology (see table 5.3). The buffer allocation from the BAP is (9 24 24 23 24 24 24), while the BCAP only allocate (3 3 2 2 2 2 3) buffers. Also, BCAP always generate acceptable solutions within the hard constraint that the total number of servers must be less or equal than the total number of buffers (see constraint 5.3). CAP, on the contrary, generates some solutions where the number of allocated servers exceed the number of allocated buffers, particularly for cases where the allocated buffers is relatively small. See for example the CAP for the 7-node series topology with $\lambda = 15$ in table 5.3, where the number of servers are larger than the number of buffers (as shown by the negative value of $\sum B$). This also happens to the 7-node merge topology in table 5.3 with $\lambda = 7$, where $\sum B = -1$. Comparing these results, we argue that the single objective BCAP performs reasonably well.

In comparison with the results from the complete enumeration, we can see that the single objective BCAP is able to find an optimal point as suggested by the complete enumeration. In table B.1 of appendix B, the result for a 3-node series topology with $\lambda = 3$ and $CV = 1.0$ is an optimal zero-buffer network with the number of servers equal to the number of buffers: (2 2 2), and a resulting throughput of 2.9954. This allocation is also optimal according to the complete enumeration method in table 3.1. Since the performance of the single objective BCAP is heavily dependant on the starting search points, we also found another optimal setting with server allocation of (2 2 2) and buffer allocation of (3 2 2) when we changed the starting search points. This allocation is also optimal according to the complete enumeration results in table 3.1. These results reflect the relatively good quality of single objective BCAP in finding optimal server and buffer allocation.

Table 5.3: Different methods, N=7

SERIES							
λ	**Method**	c	K	$\sum c$	$\sum K$	$\sum B$	θ
3	BCAP	(2 2 2 2 2 2 2)	(4 2 2 2 2 2 2)	14	16	2	2.9998
	BAP	(2 2 2 2 2 2 2)	(3 3 3 3 3 3 3)	14	21	7	2.9980
	CAP	(3 2 2 2 2 2 2)	(4 2 2 2 2 2 2)	15	16	1	3.0000
7	BCAP	(2 2 2 2 2 2 2)	(8 2 2 3 2 2 6)	14	25	11	6.9992
	BAP	(2 2 2 2 2 2 2)	(6 6 6 6 6 6 6)	14	42	28	6.9890
	CAP	(4 2 2 2 2 2 2)	(8 2 2 3 2 2 6)	16	25	9	7.0000
15	BCAP	(3 2 2 2 2 2 2)	(3 3 2 2 2 2 3)	15	17	2	15.000
	BAP	(3 2 2 2 2 2 2)	(9 24 24 23 24 24 24)	15	152	137	14.977
	CAP	(3 3 2 2 2 7 2)	(3 3 2 2 2 2 3)	21	17	-4	14.996
SPLIT							
λ	**Method**	c	K	$\sum c$	$\sum K$	$\sum B$	θ
3	BCAP	(2 2 2 1 1 1 1)	(2 2 2 2 2 2 2)	10	14	4	2.9976
	BAP	(2 2 2 1 1 1 1)	(3 2 2 2 2 2 2)	10	15	5	2.9980
	CAP	(2 2 2 1 1 1 1)	(2 2 2 2 2 2 2)	10	14	4	2.9976
7	BCAP	(3 2 2 2 2 2 2)	(3 2 2 2 2 2 2)	15	15	0	6.9986
	BAP	(3 2 2 2 2 2 2)	(4 3 3 2 2 2 2)	15	18	3	6.9970
	CAP	(3 2 2 2 2 2 2)	(3 2 2 2 2 2 2)	15	15	0	7.0000
15	BCAP	(5 2 2 2 2 2 2)	(5 2 2 2 2 2 2)	17	17	0	15.000
	BAP	(5 2 2 2 2 2 2)	(5 7 7 3 3 3 3)	17	31	14	14.993
	CAP	(3 2 2 2 2 2 2)	(5 2 2 2 2 2 2)	15	17	2	14.999
MERGE							
λ	**Method**	c	K	$\sum c$	$\sum K$	$\sum B$	θ
3	BCAP	(1 1 1 1 2 2 2)	(2 2 2 2 2 2 2)	10	14	4	2.9991
	BAP	(1 1 1 1 2 2 2)	(2 2 2 2 2 2 3)	10	15	5	2.9980
	CAP	(1 1 1 1 2 2 3)	(2 2 2 2 2 2 2)	11	14	3	2.9991
7	BCAP	(2 2 2 2 2 2 3)	(2 2 2 2 2 2 3)	15	15	0	6.9988
	BAP	(2 2 2 2 2 2 3)	(3 3 3 3 4 4 5)	15	25	10	6.9970
	CAP	(2 2 2 2 3 3 2)	(2 2 2 2 2 2 3)	16	15	-1	6.9992
15	BCAP	(3 2 2 2 3 3 4)	(3 3 3 5 3 3 4)	19	24	5	15.000
	BAP	(3 2 2 2 3 3 4)	(3 3 3 3 4 4 7)	19	27	8	14.994
	CAP	(2 2 2 2 2 2 3)	(3 3 3 5 3 3 4)	15	24	9	15.000

Figure 5.1 shows the plot of optimal buffer and server allocation from the complete enumeration in table 3.1. The positions of the two optimal allocations that are found by the single objective BCAP can be seen in this figure, both of which lie on the optimal sphere of complete enumeration. We also observed that the non-dominated solution from the BAP-CAP interaction is also present in the optimal complete enumeration result for the 3-node series network structure with $\lambda = 3$ and $CV = 1.0$. The solution is a server allocation of (2 2 2), a buffer allocation of (3 3 3), and a throughput of 2.9993 (see table A.1 in appendix A). The position of this non-dominated allocation by the BAP-CAP interaction can be seen in figure 5.1, which is also on the optimal sphere of complete enumeration. We can see that the optimal solutions from the single objective BCAP have lower total number of servers and buffers than that of BAP-CAP interaction. However, it might still be possible to find an optimal allocation with higher number of servers and buffers, if we set a high starting point for the single objective BCAP. Note that even though the result of the BAP-CAP interaction is optimal for the 3-node series network with $\lambda = 3$ and $CV = 1.0$, it does not guarantee that all allocations from this method is optimal, due to the lack of literature support for this method.

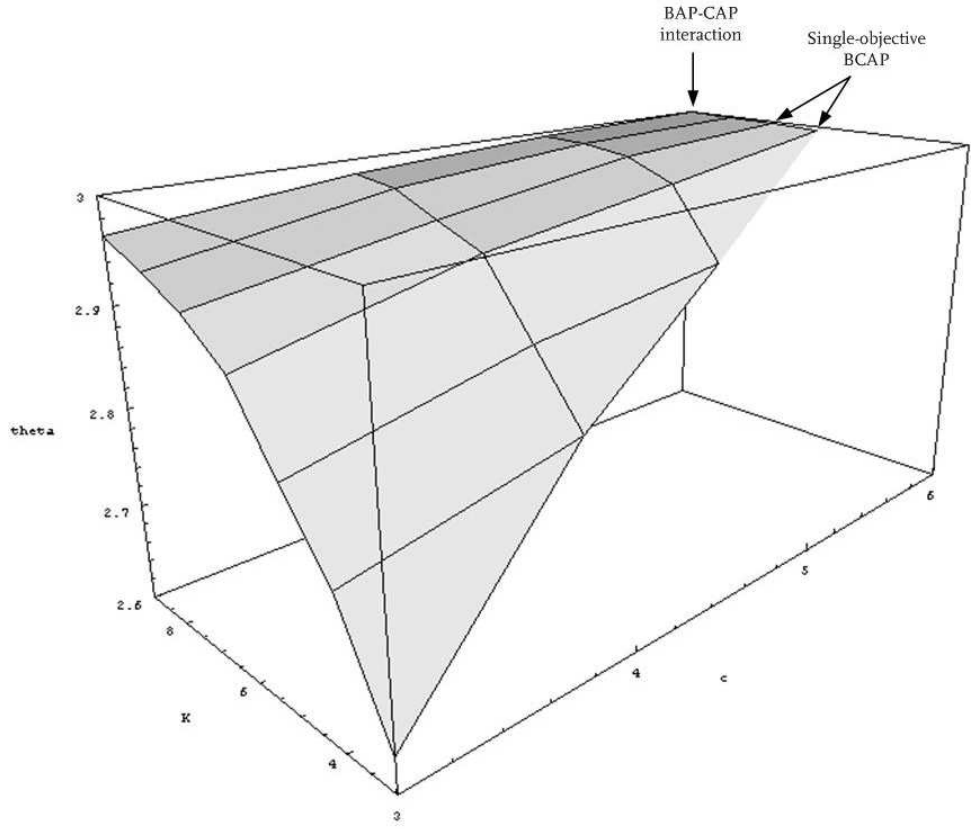


Figure 5.1: Optimal allocations of servers and buffers (complete enumeration)

Chapter 6

Comparison: combined topology

In the previous chapters, we have seen how each methodology performs in a pure series, split, and merge topology. We will next consider a combined setting of all three topologies, as can be seen in figure 6.1. This network consist of 16 nodes with the processing rate of servers in each node given in the figure. The network is adopted from the previous research of Buffer Allocation Problem (BAP) by Smith and Cruz (2005). We use here the exact same values for λ, μ, CV, and routing probabilities for the splitting node (#1 and #2). Note that the routing probability #1 refers to the upper tier of the node, while #2 refers to the lower tier of the node. Refer to figure 6.1 for the position of each node in the network.

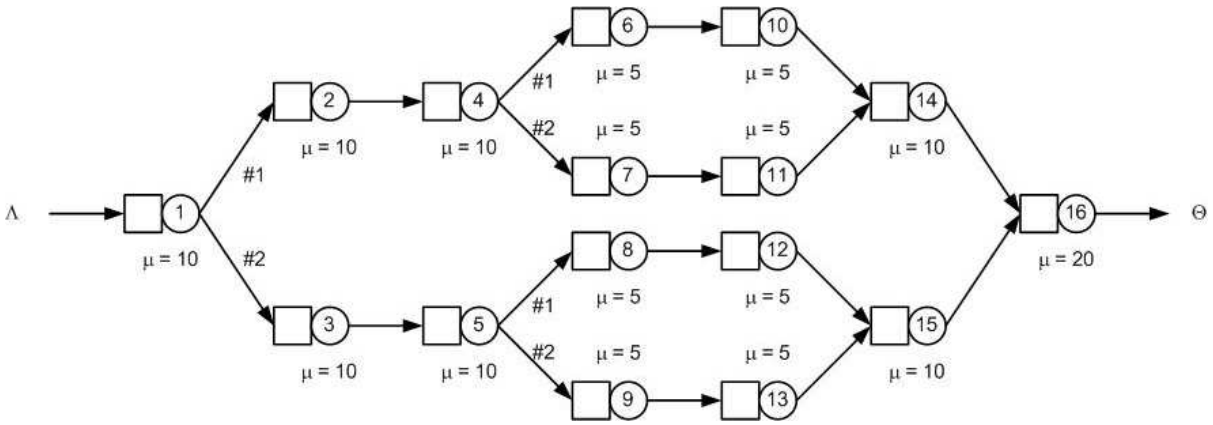


Figure 6.1: Combined topology

For the purpose of comparison, we reproduce in table 6.1 the results from Smith and Cruz (2005) for this network structure (table 29 in their paper). Note that they considered an $M/G/1/K$ setting and therefore the number of servers in all nodes is set to 1 while optimizing the buffer allocation.

Table 6.1: Results for complex topology from Smith and Cruz (2005) (BAP)

#1	**#2**	λ	CV	c	B	$\sum c$	$\sum B$	θ
0.5	0.5	5	0.5	(1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1)	(8 4 4 4 4 3 3 3 3 3 3 3 3 4 4 5)	16	61	4.9899
0.7	0.3	5	0.5	(1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1)	(8 6 3 6 3 5 5 3 3 5 5 3 3 6 3 5)	16	72	4.9916
0.5	0.5	5	1	(1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1)	(10 5 5 5 5 4 4 4 4 4 4 4 4 5 5 5)	16	77	4.9879
0.5	0.5	5	1.5	(1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1)	(11 5 5 5 5 5 5 5 5 5 5 5 5 5 5 6)	16	86	4.9877
0.6	0.4	5	1.5	(1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1)	(11 6 4 6 4 6 6 4 4 6 6 4 4 6 4 6)	16	87	4.9877
0.7	0.3	5	1.5	(1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1)	(11 7 3 7 3 6 6 3 3 6 6 3 3 7 3 6)	16	83	4.9818

6.1 Method 1: BAP-CAP interaction

The BAP-CAP interaction is applied to all settings in table 6.1. The results are given in table 6.2.

Table 6.2: Results for complex topology (BAP-CAP interaction)

#1	**#2**	λ	CV	c	K	$\sum c$	$\sum K$	$\sum B$	θ
0.5	0.5	5	0.5	(2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2)	(4 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3)	32	35	3	4.9993
0.7	0.3	5	0.5	(2 2 2 2 2 2 2 2 1 2 2 2 1 2 2 2)	(4 3 2 3 2 4 2 2 2 4 2 2 2 3 2 3)	30	42	12	4.9985
0.5	0.5	5	1	(2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2)	(5 2 2 2 2 2 2 2 2 2 2 2 2 2 3)	32	36	4	4.9990
0.5	0.5	5	1.5	(2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2)	(5 2 2 3 3 2 2 2 2 2 2 2 3 3 3)	32	40	8	4.9986
0.6	0.4	5	1.5	(2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3)	(5 3 2 3 2 3 2 2 2 3 2 2 2 3 2 3)	33	41	8	5.0000
0.7	0.3	5	1.5	(2 2 2 2 2 2 2 2 1 2 2 2 1 2 2 2)	(5 3 2 3 2 4 2 2 2 4 2 2 2 3 2 3)	30	43	13	4.9971

Analysis

As expected, we only find *one* non-dominated solution for each setting due to the fact that the arrival rate is relatively small. This finding further support our argument in the previous chapter that the BAP-CAP interaction can only find a small number of non-dominated solution for networks with low arrival rate relative to the processing rates. While we cannot say whether this non-dominated solution is also an optimal one, the result suggests that the BAP-CAP interaction is also capable of generating non-dominated solution for complex queueing network.

6.2 Method 2: Single objective BCAP

Using the single objective BCAP, we run experiments with all settings similar to table 6.1. The results are as follows.

Table 6.3: Results for complex topology (single objective BCAP)

#1	**#2**	λ	CV	c	K	$\sum c$	$\sum K$	$\sum B$	θ
0.5	0.5	5	0.5	(2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2)	(2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2)	32	32	0	4.9937
0.7	0.3	5	0.5	(2 2 2 2 2 2 2 2 1 2 2 2 1 2 2 2)	(2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2)	30	32	2	4.9974
0.5	0.5	5	1	(2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2)	(2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2)	32	32	0	4.9921
0.5	0.5	5	1.5	(2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2)	(2 2 2 2 2 2 2 2 2 2 2 2 2 2 4 2)	32	34	2	4.9986
0.6	0.4	5	1.5	(2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2)	(2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2)	32	32	0	4.9982
0.7	0.3	5	1.5	(2 2 2 2 2 2 2 2 1 2 2 2 1 2 2 2)	(2 2 2 2 2 2 2 2 3 2 2 2 3 2 2 2)	30	34	4	4.9984

Analysis

The results from our single objective BCAP methodology (table 6.3) show a higher throughput than that of the BAP (table 6.1) for every setting. As expected, we found that the optimal server allocation in BCAP is different with the server setting in the BAP. Put in another way, we found that $M/G/1/K$ is not an optimal configuration for this particular queueing network structure. This is intuitively acceptable since there are some nodes that receive more arrivals (*i.e.* merging nodes) and other nodes that receive less arrivals (*i.e.* nodes proceeding the splitting nodes). Furthermore, the processing rates of servers are not the same at each node. Extra servers that are allocated at some nodes in the network are compensated with a significant reduce in the number of buffers. In some settings, the BCAP identifies zero-buffer system as an optimal configuration, while in other setting the number of true buffer B is kept low. Even so, the resulting throughput is remarkably high. These encouraging results shows the potential

performance of the single objective BCAP method in complex queueing networks.

As we compare the results from the single objective BCAP (table 6.3) and the BAP-CAP interaction (table 6.2), we found that the single objective BCAP tends to generate optimal solutions with lower number of buffers and/or servers than the non-dominated solution of BAP-CAP interaction. This behavior is consistent with the previous results from the case of pure series, split, and merge topologies. At this point, we still cannot argue whether the non-dominated solution from BAP-CAP interaction is optimal, since they generate higher throughput than the solutions from single objective BCAP. This is intuitively acceptable since the number of buffers and servers are higher in the case of BAP-CAP interaction, and thus the throughput should also be higher.

To get more insight from the complex topology, we set up other experiments where we vary one at a time the arrival rate, coefficient of variation, and the routing probabilities. This way, we can see the effect of changing one variable while keeping the other variables fixed.

6.2.1 Different arrivals

Table 6.4 shows the results of varying the arrival rate of the complex 16-node network.

Table 6.4: Complex topology, different λ

λ	CV	c	K	$\sum c$	$\sum K$	$\sum B$	θ
5	0.5	(2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2)	(2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2)	32	32	0	4.9937
15	0.5	(4 2 2 2 2 3 3 3 3 3 3 3 3 3 3 2)	(4 2 2 2 2 3 3 3 3 3 3 3 3 4 7 3)	44	50	6	14.989
25	0.5	(5 3 3 3 3 5 5 5 5 5 5 5 5 3 3 2)	(5 3 3 3 3 5 5 5 5 5 5 5 5 3 3 2)	65	65	0	24.999

Analysis

The results suggest that changing the arrival rate brings significant effect to the allocation, particularly to the number of servers. This is due to the fact that extra arrivals will require more processing rate so as to minimize blocking. As such, there is a notable increase in the number of servers at each node. Buffers, on the contrary, are not affected heavily as can be seen from the number of pure buffers ($\sum B$) in the network. Since servers act at the same time as buffers, there is no need to add excessive buffers when the number of servers are already high. We can see that there are two zero-buffer systems (for $CV = 0.5$ and $CV = 1.5$), and only six buffers are given to the network with $CV = 1.0$).

6.2.2 Different coefficient of variation

Table 6.5 shows the results of varying the coefficient of variation of the complex 16-node network.

Table 6.5: Complex topology, different CV

λ	CV	c	K	$\sum c$	$\sum K$	$\sum B$	θ
5	0.5	(2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2)	(2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2)	32	32	0	4.9937
5	1	(2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2)	(2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2)	32	32	0	4.9921
5	1.5	(2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2)	(2 2 2 2 2 2 2 2 2 2 2 2 2 2 4 2)	32	34	2	4.9986

Analysis

Varying the coefficient of variation will not result in a dramatic change in optimal server and buffer allocation. It will only reduce the resulting throughput given a similar buffer and server allocation. In extreme cases, however, *i.e.* when the CV is set to 1.5, the resulting optimal allocation may change as compared to the allocation for lower CV's. It can be seen from the result that an additional buffer for $CV = 1.5$ leads to a notable increase in the throughput.

6.2.3 Different routing probability

Table 6.5 shows the results of varying the routing probability of the complex 16-node network.

Table 6.6: Complex topology, different routing probability

#1	**#2**	c	K	$\sum c$	$\sum K$	$\sum B$	θ
0.5	0.5	(2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2)	(2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2)	32	32	0	4.9982
0.7	0.3	(2 2 2 2 2 2 2 2 1 2 2 2 1 2 2 2)	(2 2 2 2 2 2 2 2 3 2 2 2 3 2 2 2)	30	34	4	4.9984
0.9	0.1	(2 2 1 2 1 2 1 1 1 2 1 1 1 2 2 2)	(2 2 2 2 2 2 2 2 1 2 2 2 1 2 2 2)	24	30	6	4.9966

Analysis

As we vary the routing probability at the splitting node, the optimal allocation shows a notable change. Particularly, there is a reduction in the number of allocated server at the node with small routing probability. This simply indicates that a smaller processing rate is capable of handling the arrival to that particular routing and thus the number of server is reduced. However, it is interesting to note that the reduction in the number of server is compensated with an increase in the number of buffer exactly at the node with reduced server. This change is intuitively appealing: the number of servers should be reduced to avoid excessive processing rate at nodes with small routing probability, but extra space is needed to account for the less waiting space. As such, additional buffers are allocated to these nodes to reduce blocking without increasing the processing rate.

Chapter 7

Conclusions, implications and future research

7.1 Conclusion

In this research, we have discussed a joint buffer-server optimization problem for a general $M/G/c/K$ setting. We consider arbitrary networks that consist of series, split and merge topologies, as well as the combination of all three topologies. Two optimization methodologies are discussed, namely the BAP-CAP interaction, and the single objective BCAP. The quality of the two optimization methodologies is assessed in one way by comparing their results with the results of complete enumeration.

Several non-dominated solutions, while not necessarily optimal, can be obtained using BAP-CAP interaction method for a given queueing network structure. An optimal allocation pattern is identified for the series network, namely that there is an extreme allocation of buffers to either the first or the last node in the network, if the total number of buffers cannot be divided equally among the nodes. The server allocation also gives this pattern, but less extreme. For split and merge networks, more buffers and servers are allocated to the splitting and merging node. The BAP-CAP interaction method can be tedious particularly for networks with large arrival rates relative to the processing rates, since there can be numerous non-dominated solutions that must be identified manually among other dominated solutions.

The single objective BCAP, which applies the Lagrangian relaxation to the objective function, identifies *only* a single solution for a given queueing network structure and a set of starting search points. The generated solutions depend heavily on the starting search points, where different starting search points will likely lead to different solutions. In all topologies, many zero-buffer systems are found as the optimal buffer-server allocation. Also, different server-buffer price ratio has a notable effect on the optimal server and buffer allocation. Finally, the single objective BCAP is more robust as compared to the BAP and CAP alone. It also gives a good optimal solution in comparison with the optimal allocation from the complete enumeration.

7.2 Managerial implication

The results from this study can be used especially for system level design of production/ manufacturing lines or in other relevant environments. It provides a reasonable estimate of requirements and allocations for optimal number of machines and waiting spaces for the work in process

(WIP), given data about demand, routing of jobs, machine processing rate and variability. As the result is only a rough approximation of the steady state performance, it can be used as a starting point for a more detailed analysis, *i.e.* using discrete-event simulation to obtain more insight about the system performance.

7.3 Future research

In this paper, we discussed joint buffer-server optimization in $M/G/c/K$ setting. It would be interesting to see how variation in the arrival rate affects optimal buffer and server configuration. That is, by considering a more general networks of $GI/G/c/K$.

Also, the generative model used here, namely Powell algorithm in conjunction with Coggin algorithm, is only one out of abundant search methods available in the literature. As such, future research could be directed towards joint buffer-server optimization using other search methods, *i.e.* simulated annealing, to see how the current methodology performs relative to other methods. This would be an extension from the previous study of buffer, server, and workload allocation problem for $M/M/c/K$ systems by Spinellis *et.al* (2000).

Another potential research direction is using the *genetic algorithm* as the search method in combination with GEM for $M/G/c/K$ queueing networks. This powerfull class of search method allows one to consider a *multi objective buffer-server allocation problem* with three separate objective functions (that is, without combining the three objectives into one like what we did in the single objective BCAP). Genetic algorithm is capable of identifying a set of optimal solutions, called the *Pareto-optimal set*. The pareto-optimal set will provide valuable insight of optimal buffer and server allocation for complex topologies, similar to that of figure 5.1. From this optimal set, the quality of the optimal allocations from the two optimization methodologies discussed here can further be assessed.

Bibliography

Andriansyah, R., Woensel, T.v., Cruz, F.R.B., and Smith, J.M., 2006. Performance evaluation of open zero-buffer multi-server queueing networks using the generalized expansion method. manuscript.

Cruz, F.R.B., Duarte, A.R., and Woensel, T.v., 2006. Buffer allocation in general single-server queueuing networks. accepted in Computers and Operations Research.

Cruz, F.R.B., Duczmal,L., and Woensel, T.v., 2007. A multi-objective approach for buffer allocation in general queueing networks. in review.

Daskalaki, S., and Smith, J.M. 2004. Combining Routing and Buffer Allocation Problems in Series-Parallel Queueing Networks. Annals of Operations Research 125, 47–68.

Glover, F., and Greenberg, H.J., 1989. New approaches for heuristic search: a bilateral linkage with artificial intelligence. European Journal of Operational Research 39, 119–130.

Hillier, F., Boling, R., 1967. Finite queues in series with exponential or Erlang service times: A numerical approach. Operations Research 16, 286–303.

Hillier, F., Boling, R., 1979. On the optimal allocation of work in symmetrically unbalanced production line systems with variable operation times. Management Science 25, 721–728.

Hillier, F., So, K.C., 1989. The assignment of extra servers to stations in tandem queueing systems with small or no buffers. Performance Evaluation 10, 219–231.

Hillier, F., So, K.C., 1995. On the optimal design of tandem queueing systems with finite buffers. Queueing Systems 21, 245–266.

Jain, S., Smith, J. M., 1994. Open finite queueing networks with $M/M/C/K$ parallel servers. Computers & Operations Research 21 (3), 297–317.

Kerbache, L., Smith, J. M., 1987. The generalized expansion method for open finite queueing networks. European Journal of Operational Research 32, 448–461.

Kerbache, L., Smith, J. M., 1988. Asymptotic behavior of the expansion method for open finite queueing networks. Computers & Operations Research 15 (2), 157–169.

Kerbache, L., Smith, J. M., 2000. Multi-objective routing within large scale facilities using open finite queueing networks. European Journal of Operational Research 121, 105–123.

Kimura, T., 1996. A transform free-approximation for the finite capacity $M/G/s$ queue. Operations Research 44, 165–180.

Kuester, J.L., Mize, J.H., 1973. Optimization Techniques with FORTRAN. MacGraw Hill College Div., New York, NY, USA.

Labetoulle, J., and Pujolle, G., 1964. Isolation method in a network of queueus. IEEE Transactions on Software Engineering 6, 373–381.

Lemarechal, C. 2001. Lagrangian relaxation. Computational Combinatorial Optimization, 112–156.

Lemarechal, C. 2001. The omniprescence of Lagrange. 4OR: A Quarterly Journal of Operations Research 1, 7–25.

Magazine, M.J., and Stecke, K.E., 1996. Throughput for production lines with serial work stations and parallel service facilities. Performance Evaluation 25, 211–232.

Powell, M.J.D., 1964. An efficient method for finding the minimum of a function of several variables without calculating derivatives. Computer Journal 7, 155–162.

Smith, J.M., 2003. $M/G/c/K$ blocking probability models and system performance. Performance Evaluation 52, 237–267.

Smith, J.M., and Cruz, F.R.B., 2005. The buffer allocation problem for general finite buffer queueing networks. IIE Transactions 37, 343–365.

Smith, J.M., and Daskalaki, S., 1988. Buffer space allocation in automated asemby lines. Operations Research 36, 343–358.

Smith, J.M., Cruz, F.R.B., and Woensel, T.v., 2006. Topological network design of general finite, multi-server queueing networks. in review.

Spinellis, D., Papadopoulos, C., Smith, J.M., 2000. Large production line optimization using simulated annealing. International Journal of Production Research 38, 509–541.

Suri, R., 1985. An overview of evaluative models for flexible manufacturing systems. Annals of Operations Research 3, 13–21.

Appendix A

Complete results: BAP-CAP interaction

Table A.1: BAP-CAP interaction results, N=3

SERIES							
λ	CV	c	K	$\sum c$	$\sum K$	$\sum B$	θ
3	1.0	(2 2 2)	(3 3 3)	6	9	3	2.9993
		(2 2 3)	(3 3 3)	7	9	2	3.0000
7	1.0	(2 2 2)	(6 6 6)	6	18	12	7.0000
15	1.0	(3 2 2)	(23 10 10)	7	43	36	15.000
		(4 2 2)	(11 8 8)	8	27	19	15.000
		(3 4 4)	(7 7 7)	11	21	10	14.998
	0.5	(2 2 2)	(9 9 20)	6	38	32	15.000
		(2 2 3)	(9 9 9)	7	27	20	14.995
		(2 2 4)	(9 9 9)	8	27	19	15.000
	1.5	(2 2 3)	(11 11 11)	7	33	26	14.997
		(2 3 2)	(11 25 11)	7	47	40	15.000
SPLIT							
λ	CV	c	K	$\sum c$	$\sum K$	$\sum B$	θ
3	1.0	(2 2 1)	(3 2 3)	5	8	3	2.9990
		(2 2 2)	(3 2 2)	6	7	1	2.9998
		(2 2 2)	(3 2 3)	6	8	2	3.0000
		(2 2 3)	(3 2 2)	7	7	0	3.0000
7	1	(2 2 2)	(6 3 3)	6	12	6	7.0000
15	1	(2 2 2)	(7 4 7)	6	18	12	14.997
		(2 2 2)	(9 8 7)	6	24	18	14.999
		(2 2 3)	(7 4 4)	7	15	8	14.997
		(3 2 2)	(7 4 7)	7	18	11	15.000
		(4 4 2)	(7 4 4)	10	15	5	14.999
	0.5	(2 2 2)	(9 7 7)	6	23	17	14.998
		(2 2 3)	(6 6 4)	7	16	9	14.998
		(3 2 2)	(20 6 4)	7	30	23	15.000
		(2 4 2)	(6 6 4)	8	16	8	15.000
	1.5	(2 2 2)	(13 9 9)	6	31	25	14.999
		(3 2 2)	(7 5 5)	7	17	10	15.000
MERGE							
λ	CV	c	K	$\sum c$	$\sum K$	$\sum B$	θ
3	1	(2 2 2)	(2 2 3)	6	7	1	2.9998
		(2 2 3)	(2 2 3)	7	7	0	3.0000
7	1	(2 2 2)	(3 3 6)	6	12	6	6.9979
		(2 2 3)	(3 3 4)	7	10	3	6.9992
		(2 2 3)	(3 3 6)	7	12	5	7.0000
15	1	(2 2 2)	(7 7 23)	6	37	31	14.994
		(2 2 3)	(8 8 11)	7	27	20	14.996
		(2 2 3)	(7 7 23)	7	37	30	15.000
		(2 2 4)	(7 4 10)	8	21	13	15.000
	0.5	(2 2 2)	(6 6 20)	6	32	26	14.994
		(2 3 2)	(7 7 9)	7	23	16	15.000
	1.5	(2 2 2)	(8 8 25)	6	41	35	14.994
		(2 3 2)	(8 8 5)	7	21	14	15.000

Table A.2: BAP-CAP interaction results, N=7

SERIES							
λ	CV	c	K	$\sum c$	$\sum K$	$\sum B$	θ
3	1.0	(2 2 2 2 2 2 2)	(3 3 3 3 3 3 3)	14	21	7	2.9986
7	1.0	(2 2 2 2 2 2 2)	(6 6 6 6 6 6 6)	14	42	28	7.0000
15	0.5	(2 2 2 2 2 2 2)	(5 5 5 5 5 5 6)	14	36	22	14.993
		(2 2 2 2 2 2 3)	(5 5 5 5 5 5 20)	15	50	35	15.000
		(5 5 5 5 5 5 2)	(5 5 5 5 5 5 6)	32	36	4	14.998
		(5 5 5 5 5 5 4)	(6 6 6 6 6 6 9)	34	45	11	15.000
	1.0	(2 2 2 2 2 2 3)	(10 10 10 10 10 10 23)	15	83	68	15.000
		(2 2 2 2 2 2 4)	(7 7 7 7 7 7 7)	16	49	33	14.995
		(4 4 4 4 4 4 4)	(10 10 10 10 10 10 10)	28	70	42	15.000
	1.5	(2 2 2 2 2 2 3)	(11 11 11 11 11 11 25)	15	91	76	15.000
		(2 2 2 2 2 2 4)	(11 11 11 11 11 11 11)	16	77	61	15.000
SPLIT							
λ	CV	c	K	$\sum c$	$\sum K$	$\sum B$	θ
3	1.0	(2 2 2 1 1 1 1)	(3 2 2 2 2 2 2)	10	15	5	2.9980
7	1	(2 2 2 1 1 1 1)	(6 3 3 3 3 3 3)	10	24	14	6.9889
		(2 2 2 1 1 1 1)	(6 3 3 4 4 4 4)	10	28	18	6.9980
		(2 2 2 1 1 1 2)	(4 3 3 4 4 4 2)	11	24	13	6.9985
		(2 2 2 1 1 1 2)	(8 3 3 4 4 4 2)	11	28	17	6.9992
		(3 2 2 1 1 1 2)	(6 3 3 4 4 4 2)	12	26	14	6.9995
		(2 2 2 2 2 2 2)	(6 3 3 2 2 2 2)	14	20	6	7.0000
15	1	(2 2 2 2 2 2 2)	(10 7 7 3 3 3 3)	14	36	22	14.990
		(3 2 2 2 2 2 2)	(10 7 7 3 3 3 3)	15	36	21	15.000
	0.5	(2 2 2 2 2 2 2)	(9 6 6 3 3 3 3)	14	33	19	14.994
		(2 2 2 2 2 2 2)	(10 8 8 4 4 4 4)	14	42	28	14.999
		(3 2 2 2 2 2 2)	(9 6 6 3 3 3 3)	15	33	18	15.000
	1.5	(2 2 2 2 2 2 2)	(11 8 8 4 4 4 4)	14	43	29	14.997
		(3 2 2 2 2 2 2)	(25 8 8 4 4 4 4)	15	57	42	15.000
MERGE							
λ	CV	c	K	$\sum c$	$\sum K$	$\sum B$	θ
3	1.0	(1 1 1 1 2 2 2)	(2 2 2 2 2 2 3)	10	15	5	2.9991
7	1.0	(2 2 2 2 2 2 2)	(2 2 2 2 3 3 6)	14	20	6	6.9988
15	1.0	(2 2 2 2 2 2 2)	(3 3 3 3 7 7 23)	14	49	35	14.989
		(3 2 2 2 2 2 2)	(3 3 3 3 8 8 7)	15	35	20	15.000
		(2 2 2 2 2 2 4)	(3 3 3 3 7 7 7)	16	33	17	14.998
		(3 2 2 2 2 2 4)	(3 3 3 3 7 7 7)	17	33	16	14.999
	0.5	(2 2 2 2 2 2 2)	(3 3 3 3 8 4 9)	14	33	19	14.999
		(2 2 2 2 2 3 2)	(3 3 3 3 6 6 6)	15	30	15	14.997
		(2 2 2 2 2 3 2)	(3 3 3 3 8 7 9)	15	36	21	15.000
		(2 2 2 2 2 2 4)	(3 3 3 3 6 6 6)	16	30	14	14.998
		(2 2 2 2 2 2 4)	(3 3 3 3 6 4 9)	16	31	15	15.000
	1.5	(2 2 2 2 2 2 2)	(4 4 4 4 8 8 25)	14	57	43	14.993
		(2 2 2 2 2 2 2)	(4 4 4 4 9 9 26)	14	60	46	14.994
		(2 2 2 2 2 3 2)	(4 4 4 4 8 8 5)	15	37	22	15.000

Table A.3: BAP-CAP interaction results, N=15

SERIES							
λ	CV	c	K	$\sum c$	$\sum K$	$\sum B$	θ
3	1.0	(2 2 2 2 2 2 2 2 2 2 2 2 2 2 2)	(3 3 3 3 3 3 3 3 3 3 3 3 3 3 3)	30	45	15	2.9971
7	1.0	(2 2 2 2 2 2 2 2 2 2 2 2 2 2 2)	(4 6 6 6 6 6 6 6 6 6 6 6 6 6 6)	30	88	58	6.9862
		(2 2 2 2 2 2 2 2 2 2 2 2 2 2 2)	(8 8 8 8 8 8 8 8 8 8 8 8 8 8 8)	30	120	90	6.9983
		(3 2 2 2 2 2 2 2 2 2 2 2 2 2 2)	(4 6 6 6 6 6 6 6 6 6 6 6 6 6 6)	31	88	57	7.0000
15	1	(2 2 2 2 2 2 2 2 2 2 2 2 2 2 3)	(10 10 10 10 10 10 10 10 10 10 10 10 10 10 23)	31	163	132	15.000
		(2 2 2 2 2 2 2 2 2 2 2 2 2 2 4)	(7 7 7 7 7 7 7 7 7 7 7 7 7 7 7)	32	105	73	14.989
		(4 4 4 4 4 4 4 4 4 4 4 4 4 4 4)	(10 10 10 10 10 10 10 10 10 10 10 10 10 10 10)	60	150	90	14.999
	0.5	(2 2 2 2 2 2 2 2 2 2 2 2 2 2 3)	(9 9 9 9 9 9 9 9 9 9 9 9 9 9 20)	31	146	115	15.000
		(2 2 2 2 2 2 2 2 2 2 2 2 2 2 4)	(9 9 9 9 9 9 9 9 9 9 9 9 9 9 6)	32	132	100	14.984
		(3 3 3 3 3 3 3 3 3 3 3 3 3 3 4)	(9 9 9 9 9 9 9 9 9 9 9 9 9 9 9)	46	135	89	14.999
	1.5	(2 2 2 2 2 2 2 2 2 2 2 2 2 2 3)	(11 11 11 11 11 11 11 11 11 11 11 11 11 11 25)	31	179	148	15.000
		(2 2 2 2 2 2 2 2 2 2 2 2 2 2 4)	(11 11 11 11 11 11 11 11 11 11 11 11 11 11 11)	32	165	133	14.999
SPLIT							
λ	CV	c	K	$\sum c$	$\sum K$	$\sum B$	θ
3	1	(2 2 2 1 1 1 1 1 1 1 1 1 1 1 1)	(3 2 2 2 2 2 2 1 1 1 1 1 1 1 1)	18	23	5	2.9940
		(2 2 2 1 1 1 1 1 1 1 1 1 1 1 1)	(3 2 2 2 2 2 2 2 2 2 2 2 2 2 2)	18	31	13	2.9991
		(2 2 2 1 1 1 1 1 1 1 1 1 1 1 2)	(3 2 2 2 2 2 2 1 1 1 1 1 1 1 1)	19	23	4	2.9912
		(2 2 2 1 1 1 1 1 1 1 1 1 1 1 2)	(3 2 2 2 2 2 2 1 1 1 1 1 1 1 2)	19	24	5	2.9950
7	1	(2 2 2 2 2 2 2 1 1 1 1 1 1 1 2)	(4 3 3 2 2 2 2 2 2 2 2 2 2 2 2)	23	34	11	6.9977
		(3 2 2 2 2 2 2 1 1 1 1 1 1 1 1)	(6 3 3 2 2 2 2 2 2 2 2 2 2 2 2)	23	36	13	6.9978
15	1	(2 2 2 2 2 2 2 1 1 1 1 1 1 1 2)	(10 7 7 3 3 3 3 4 4 4 4 4 4 4 2)	23	66	43	14.994
		(2 2 2 2 2 2 2 1 1 1 1 1 1 1 2)	(12 8 8 5 5 5 5 5 5 5 5 5 5 5 3)	23	86	63	14.999
		(3 2 2 2 2 2 2 1 1 1 1 1 1 1 2)	(23 7 7 3 3 3 3 4 4 4 4 4 4 4 2)	24	79	55	14.998
		(2 2 2 2 2 2 2 2 2 2 2 2 2 2 2)	(10 7 7 3 3 3 3 2 2 2 2 2 2 2 2)	30	52	22	14.995
		(2 2 2 2 2 2 2 2 2 2 2 2 2 2 2)	(10 7 7 3 3 3 3 4 4 4 4 4 4 4 2)	30	66	36	15.000
		(3 2 2 2 2 2 2 2 2 2 2 2 2 2 2)	(10 7 7 3 3 3 3 2 2 2 2 2 2 2 2)	31	52	21	15.000
	0.5	(2 2 2 2 2 2 2 1 1 1 1 1 1 1 2)	(9 6 6 3 3 3 3 3 3 3 3 3 3 3 2)	23	56	33	14.993
		(2 2 2 2 2 2 2 1 1 1 1 1 1 1 2)	(9 6 6 3 3 3 3 3 3 3 3 3 3 3 3)	23	57	34	14.995
		(3 2 2 2 2 2 2 1 1 1 1 1 1 1 1)	(21 7 7 4 4 4 4 3 3 3 3 3 3 3 4)	23	76	53	14.996
		(2 2 2 2 2 2 2 1 1 1 1 1 1 1 3)	(9 6 6 3 3 3 3 3 3 3 3 3 3 3 2)	24	56	32	14.996
		(2 2 2 2 2 2 2 2 2 2 2 2 2 2 1)	(9 6 6 3 3 3 3 2 2 2 2 2 2 2 2)	29	49	20	14.884
		(2 2 2 2 2 2 2 2 2 2 2 2 2 2 2)	(9 6 6 3 3 3 3 2 2 2 2 2 2 2 2)	30	49	19	14.996
		(2 2 2 2 2 2 2 2 2 2 2 2 2 2 2)	(9 6 6 3 3 3 3 2 2 2 2 2 2 2 3)	30	50	20	15.000
		(2 2 2 2 2 2 2 2 2 2 2 2 2 2 3)	(9 6 6 3 3 3 3 2 2 2 2 2 2 2 2)	31	49	18	15.000
	1.5	(2 2 2 2 2 2 2 2 2 2 2 2 2 2 2)	(5 8 8 4 4 4 4 2 2 2 2 2 2 2 2)	30	53	23	14.996
		(3 2 2 2 2 2 2 2 2 2 2 2 2 2 2)	(25 8 8 4 4 4 4 2 2 2 2 2 2 2 2)	31	73	42	15.000
		(4 2 2 2 2 2 2 2 2 2 2 2 2 2 2)	(11 8 8 4 4 4 4 2 2 2 2 2 2 2 2)	32	59	27	15.000
		(5 2 2 2 2 2 2 2 2 2 2 2 2 2 2)	(5 8 8 4 4 4 4 2 2 2 2 2 2 2 2)	33	53	20	14.998
		(5 2 2 2 2 2 2 2 2 2 2 2 2 2 2)	(7 8 8 4 4 4 4 2 2 2 2 2 2 2 2)	33	55	22	15.000
MERGE							
λ	CV	c	K	$\sum c$	$\sum K$	$\sum B$	θ
3	1.0	(1 1 1 1 1 1 1 1 1 1 1 1 2 2 2)	(1 1 1 1 1 1 1 1 2 2 2 2 2 2 3)	18	23	5	2.9969
		(1 1 1 1 1 1 1 1 1 1 1 1 2 2 2)	(2 2 2 2 2 2 2 2 2 2 2 2 2 2 3)	18	31	13	2.9991
7	1	(1 1 1 1 1 1 1 1 2 2 2 2 2 2 2)	(2 2 2 2 2 2 2 2 2 2 2 2 3 3 6)	22	36	14	6.9964
15	1	(2 2 2 2 2 2 2 2 2 2 2 2 2 2 2)	(2 2 2 2 2 2 2 2 3 3 3 3 7 7 23)	30	65	35	14.987
		(2 2 2 2 2 2 2 2 2 2 2 2 2 2 3)	(2 2 2 2 2 2 2 2 3 3 3 3 8 8 10)	31	54	23	14.990
		(2 2 2 2 2 2 2 2 2 2 2 2 2 2 3)	(2 2 2 2 2 2 2 2 3 3 3 3 7 7 23)	31	65	34	15.000
		(2 2 2 2 2 2 2 2 2 2 2 2 2 3 3)	(2 2 2 2 2 2 2 2 3 3 3 3 7 7 7)	32	49	17	15.000
	0.5	(2 2 2 2 2 2 2 2 2 2 2 2 2 2 2)	(2 2 2 2 2 2 2 3 3 3 3 6 6 20)	30	60	30	14.989
		(2 2 2 2 2 2 2 2 2 2 2 2 2 2 2)	(2 2 2 2 2 2 2 3 3 3 3 7 7 22)	30	64	34	14.990
		(2 2 2 2 2 2 2 2 2 2 2 2 2 3 2)	(2 2 2 2 2 2 2 3 3 3 3 6 6 6)	31	46	15	14.996
		(2 2 2 2 2 2 2 2 2 2 2 2 2 2 3)	(2 2 2 2 2 2 2 3 3 3 3 6 6 20)	31	60	29	15.000
		(2 2 2 2 2 2 2 2 2 2 2 2 2 2 4)	(2 2 2 2 2 2 2 3 3 3 3 6 6 6)	32	46	14	14.997
		(2 2 2 2 2 2 2 2 2 2 2 2 2 2 4)	(2 2 2 2 2 2 2 3 3 3 3 6 4 9)	32	47	15	15.000
	1.5	(2 2 2 2 2 2 2 2 2 2 2 2 2 2 2)	(2 2 2 2 2 2 2 2 4 4 4 4 8 8 25)	30	73	43	14.991
		(2 2 2 2 2 2 2 2 2 2 2 2 2 2 2)	(2 2 2 2 2 2 2 2 4 4 4 4 9 9 26)	30	76	46	14.992
		(2 2 2 2 2 2 2 2 2 2 2 2 2 3 2)	(2 2 2 2 2 2 2 2 4 4 4 4 8 5 11)	31	56	25	14.996
		(2 2 2 2 2 2 2 2 2 2 2 2 2 3 2)	(2 2 2 2 2 2 2 2 4 4 4 4 8 8 11)	31	59	28	15.000
		(2 2 2 2 2 2 2 2 2 2 2 2 2 2 4)	(2 2 2 2 2 2 2 2 4 4 4 4 8 5 11)	32	56	24	15.000

Appendix B

Complete results: single objective BCAP

Table B.1: Single objective BCAP results, N=3

SERIES							
λ	CV	c	K	$\sum c$	$\sum K$	$\sum B$	θ
3	0.5	(2 2 2)	(2 2 2)	6	6	0	2.9963
	1	(2 2 2)	(2 2 2)	6	6	0	2.9954
	1.5	(2 2 2)	(2 2 2)	6	6	0	2.9943
7	0.5	(2 2 2)	(2 2 2)	6	6	0	6.8313
	1	(2 2 2)	(2 2 2)	6	6	0	6.7948
	1.5	(2 2 2)	(2 3 2)	6	7	1	6.9306
15	0.5	(4 2 3)	(7 5 3)	9	15	6	14.999
	1	(4 2 3)	(8 2 3)	9	13	4	14.999
	1.5	(3 2 2)	(13 5 3)	7	21	14	14.998
SPLIT							
λ	CV	c	K	$\sum c$	$\sum K$	$\sum B$	θ
3	0.5	(2 2 2)	(2 2 2)	6	6	0	2.9998
	1	(2 2 2)	(2 2 2)	6	6	0	2.9998
	1.5	(2 2 2)	(2 2 2)	6	6	0	2.9997
7	0.5	(3 2 2)	(3 2 3)	7	8	1	6.9997
	1	(2 2 2)	(2 2 3)	6	7	1	6.9993
	1.5	(3 2 2)	(3 2 3)	7	8	1	6.9966
15	0.5	(3 2 2)	(3 2 2)	7	7	0	15.000
	1	(3 2 2)	(3 2 2)	7	7	0	14.999
	1.5	(3 2 2)	(3 2 2)	7	7	0	14.999
MERGE							
λ	CV	c	K	$\sum c$	$\sum K$	$\sum B$	θ
3	0.5	(2 2 2)	(2 2 2)	6	6	0	2.9998
	1	(2 2 2)	(2 2 2)	6	6	0	2.9998
	1.5	(2 2 2)	(2 2 2)	6	6	0	2.9997
7	0.5	(2 2 3)	(3 3 3)	7	9	2	6.9985
	1	(2 3 3)	(3 3 3)	8	9	1	6.9999
	1.5	(3 3 3)	(3 3 3)	9	9	0	6.9999
15	0.5	(2 2 4)	(3 3 4)	8	10	2	14.999
	1	(2 2 5)	(7 2 5)	9	14	5	14.999
	1.5	(2 2 5)	(8 7 5)	9	20	11	14.997

Table B.2: Single objective BCAP results, N=7

SERIES							
λ	CV	c	K	$\sum c$	$\sum K$	$\sum B$	θ
3	0.5	(2 2 2 2 2 2 2)	(2 2 2 2 2 3 5)	14	18	4	3.0000
	1	(2 2 2 2 2 2 2)	(4 2 2 2 2 2 2)	14	16	2	2.9998
	1.5	(2 2 2 2 2 2 2)	(4 2 2 2 2 2 2)	14	16	2	2.9997
7	0.5	(2 2 2 2 2 2 2)	(7 2 2 7 2 2 3)	14	25	11	6.9988
	1	(2 2 2 2 2 2 2)	(8 2 2 3 2 2 6)	14	25	11	6.9992
	1.5	(3 3 3 3 3 3 3)	(3 3 3 3 3 3 3)	21	21	0	6.9866
15	0.5	(2 2 2 2 2 2 4)	(2 2 2 2 2 2 5)	16	17	1	14.936
	1	(3 2 2 2 2 2 2)	(3 3 2 2 2 2 3)	15	17	2	15.000
	1.5	(3 2 2 2 2 2 2)	(3 2 2 2 2 2 3)	15	16	1	15.000
SPLIT							
λ	CV	c	K	$\sum c$	$\sum K$	$\sum B$	θ
3	0.5	(2 2 2 1 1 1 1)	(2 2 2 2 2 2 2)	10	14	4	2.9984
	1	(2 2 2 1 1 1 1)	(2 2 2 2 2 2 2)	10	14	4	2.9976
	1.5	(2 2 2 1 1 1 1)	(2 2 2 2 2 2 2)	10	14	4	2.9967
7	0.5	(3 2 2 2 2 2 2)	(3 2 2 2 2 2 2)	15	15	0	6.9989
	1	(3 2 2 2 2 2 2)	(3 2 2 2 2 2 2)	15	15	0	6.9986
	1.5	(3 2 2 2 2 2 2)	(3 2 2 2 2 2 2)	15	15	0	6.9982
15	0.5	(4 2 2 2 2 2 2)	(7 2 2 2 2 2 2)	16	19	3	14.999
	1	(5 2 2 2 2 2 2)	(5 2 2 2 2 2 2)	17	17	0	15.000
	1.5	(4 2 2 2 2 2 2)	(4 2 2 2 2 2 3)	16	17	1	14.996
MERGE							
λ	CV	c	K	$\sum c$	$\sum K$	$\sum B$	θ
3	0.5	(1 1 1 1 2 2 2)	(2 2 2 2 2 2 2)	10	14	4	2.9994
	1	(1 1 1 1 2 2 2)	(2 2 2 2 2 2 2)	10	14	4	2.9991
	1.5	(1 1 1 1 2 2 2)	(2 2 2 2 2 2 2)	10	14	4	2.9988
7	0.5	(2 2 2 2 2 2 3)	(2 2 2 2 2 2 3)	15	15	0	6.9990
	1	(2 2 2 2 2 2 3)	(2 2 2 2 2 2 3)	15	15	0	6.9988
	1.5	(2 2 2 2 2 2 3)	(2 2 2 2 2 2 3)	15	15	0	6.9985
15	0.5	(2 2 2 2 2 2 5)	(3 3 2 4 2 2 5)	17	21	4	14.999
	1	(3 2 2 2 3 3 4)	(3 3 3 5 3 3 4)	19	24	5	15.000
	1.5	(3 2 2 2 2 2 5)	(3 3 3 5 2 2 5)	18	23	5	14.999

Table B.3: Single objective BCAP results, N=15

SERIES							
λ	CV	c	K	$\sum c$	$\sum K$	$\sum B$	θ
3	0.5	(2 2 2 2 2 2 2 2 2 2 2 2 2 2 2)	(2 2 2 2 2 2 2 2 2 2 2 2 2 2 2)	30	30	0	2.9897
	1	(2 2 2 2 2 2 2 2 2 2 2 2 2 2 2)	(2 2 2 2 2 2 2 2 2 2 2 2 2 2 3)	30	31	1	2.9813
	1.5	(2 2 2 2 2 2 2 2 2 2 2 2 2 2 2)	(2 2 2 2 2 2 2 2 2 2 2 2 2 2 3)	30	31	1	2.9771
7	0.5	(2 2 2 2 2 2 2 2 2 2 2 2 2 2 2)	(2 2 2 2 2 2 2 2 2 2 2 2 2 2 2)	30	30	0	6.4364
	1	(2 2 2 2 2 2 2 2 2 2 2 2 2 2 2)	(2 2 2 2 2 2 2 2 2 2 2 2 2 2 2)	30	30	0	6.3387
	1.5	(2 2 2 2 2 2 2 2 2 2 2 2 2 2 2)	(2 2 2 2 2 2 2 2 2 2 2 2 2 2 2)	30	30	0	6.2468
15	0.5	(2 2 2 2 2 2 2 2 2 2 2 2 2 5 2)	(2 2 2 2 2 2 2 2 2 2 2 2 2 5 2)	33	33	0	14.996
	1	(2 2 2 2 2 2 2 5 2 2 2 2 2 5 2)	(2 2 2 2 2 2 2 6 2 2 2 2 2 5 2)	36	37	1	14.998
	1.5	(2 2 2 5 2 2 2 2 2 2 2 2 2 2 2)	(5 5 2 7 2 2 2 2 2 2 2 2 2 2 2)	33	41	8	14.999
SPLIT							
λ	CV	c	K	$\sum c$	$\sum K$	$\sum B$	θ
3	0.5	(2 2 2 1 1 1 1 1 1 1 1 1 1 1 1)	(2 2 2 2 2 2 2 1 1 1 1 1 1 1 2)	18	23	5	2.9977
	1	(2 2 2 1 1 1 1 1 1 1 1 1 1 1 1)	(2 2 2 2 2 2 2 1 1 1 1 1 1 1 2)	18	23	5	2.9971
	1.5	(2 2 2 1 1 1 1 1 1 1 1 1 1 1 1)	(2 2 2 2 2 2 2 1 1 1 1 1 1 1 2)	18	23	5	2.9965
7	0.5	(2 2 2 2 2 2 2 1 1 1 1 1 1 1 2)	(2 2 2 2 2 2 2 2 2 2 2 2 2 2 2)	23	30	7	6.9925
	1	(2 2 2 2 2 2 2 1 1 1 1 1 1 1 2)	(2 2 2 2 2 2 2 2 2 2 2 2 2 2 2)	23	30	7	6.9890
	1.5	(2 2 2 2 2 2 2 1 1 1 1 1 1 1 2)	(2 2 2 2 2 2 2 2 2 2 2 2 2 2 2)	23	30	7	6.9847
15	0.5	(2 2 2 2 2 2 2 2 2 2 2 2 2 2 2)	(2 2 2 2 2 2 2 2 2 2 2 2 2 2 2)	30	30	0	14.996
	1	(5 2 2 2 2 2 2 2 2 2 2 2 2 2 2)	(5 2 2 2 2 2 2 2 2 2 2 2 2 2 2)	33	33	0	14.995
	1.5	(5 2 2 2 2 2 2 2 2 2 2 2 2 2 2)	(6 2 2 2 2 2 2 2 2 2 2 2 2 2 2)	33	34	1	14.994
MERGE							
λ	CV	c	K	$\sum c$	$\sum K$	$\sum B$	θ
3	0.5	(1 1 1 1 1 1 1 1 1 1 1 2 2 2)	(1 1 1 1 1 1 1 2 2 2 2 2 2 2)	18	22	4	2.9975
	1	(1 1 1 1 1 1 1 1 1 1 1 2 2 2)	(1 1 1 1 1 1 1 2 2 2 2 2 2 2)	18	22	4	2.9969
	1.5	(1 1 1 1 1 1 1 1 1 1 1 2 2 2)	(1 1 1 1 1 1 1 2 2 2 2 2 2 2)	18	22	4	2.9963
7	0.5	(1 1 1 1 1 1 1 2 2 2 2 2 2 2)	(2 2 2 2 2 2 2 2 2 2 2 2 2 2)	22	30	8	6.9974
	1	(1 1 1 1 1 1 1 2 2 2 2 2 2 2)	(2 2 2 2 2 2 2 2 2 2 2 2 2 2)	22	30	8	6.9964
	1.5	(1 1 1 1 1 1 1 2 2 2 2 2 2 2)	(2 2 2 2 2 2 2 2 2 2 2 2 2 2)	22	30	8	6.9952
15	0.5	(2 2 2 2 2 2 2 2 2 2 2 3 3 4)	(2 2 2 2 2 2 2 3 3 3 4 3 3 4)	34	39	5	14.998
	1	(2 2 2 2 2 2 2 2 2 2 2 3 3 4)	(2 2 2 2 2 2 2 3 3 3 5 3 3 4)	34	40	6	14.997
	1.5	(2 2 2 2 2 2 2 2 2 2 2 2 2 5)	(2 2 2 2 2 2 2 3 3 3 3 7 7 5)	33	47	14	14.992

Table B.4: Different server-buffer price ratio, N=3

SERIES	$c-K$ **price ratio**		c	K	$\sum c$	$\sum K$	$\sum B$	θ
	c	K						
λ**= 3**	1	1	(2 2 2)	(2 2 2)	6	6	0	2.9954
CV**=1.0**	2	1	(1 1 1)	(6 6 6)	3	18	15	2.9990
	3	1	(1 1 1)	(6 6 6)	3	18	15	2.9990
	4	1	(1 1 1)	(6 6 6)	3	18	15	2.9990
	10	1	(1 1 1)	(7 6 6)	3	19	16	2.9994
	1	2	(2 2 2)	(2 2 2)	6	6	0	2.9954
	1	3	(2 2 2)	(2 2 2)	6	6	0	2.9954
	1	4	(2 2 2)	(2 2 2)	6	6	0	2.9954
	1	10	(2 2 2)	(2 2 2)	6	6	0	2.9954
SPLIT	$c-K$ **price ratio**		c	K	$\sum c$	$\sum K$	$\sum B$	θ
	c	K						
λ**= 7**	1	1	(2 2 2)	(2 2 3)	6	7	1	6.9993
CV**=1.0**	2	1	(2 1 1)	(7 6 6)	4	19	15	6.9979
	3	1	(2 1 1)	(7 7 7)	4	21	17	6.9989
	4	1	(2 1 1)	(7 7 7)	4	21	17	6.9989
	10	1	(2 1 1)	(8 7 7)	4	22	18	6.9992
	1	2	(4 2 2)	(4 2 2)	8	8	0	7.0000
	1	3	(4 2 2)	(4 2 2)	8	8	0	7.0000
	1	4	(4 2 2)	(4 2 2)	8	8	0	7.0000
	1	10	(4 2 2)	(4 2 2)	8	8	0	7.0000
MERGE	$c-K$ **price ratio**		c	K	$\sum c$	$\sum K$	$\sum B$	θ
	c	K						
λ**= 15**	1	1	(2 2 5)	(7 2 5)	9	14	5	14.999
CV**=1.0**	2	1	(2 2 4)	(7 7 7)	8	21	13	14.999
	3	1	(2 2 4)	(8 7 7)	8	22	14	14.998
	4	1	(2 2 4)	(8 7 8)	8	23	15	14.999
	10	1	(2 2 4)	(9 8 8)	8	25	17	14.999
	1	2	(4 2 5)	(4 3 5)	11	12	1	14.999
	1	3	(4 2 5)	(4 3 5)	11	12	1	14.999
	1	4	(4 2 5)	(4 3 5)	11	12	1	14.999
	1	10	(4 2 5)	(4 3 6)	11	13	2	14.999

Table B.5: Different server-buffer price ratio, N=7

SERIES	$c-K$ **price ratio**		c	K	$\sum c$	$\sum K$	$\sum B$	θ
	c	K						
λ **= 7**	1	1	(2 2 2 2 2 2 2)	(8 2 2 3 2 2 6)	14	25	11	6.9992
CV**=1.0**	2	1	(2 2 2 2 2 2 2)	(8 2 2 3 2 2 6)	14	25	11	6.9992
	3	1	(2 2 2 2 2 2 2)	(8 2 2 3 2 2 7)	14	26	12	6.9997
	4	1	(2 2 2 2 2 2 2)	(9 2 2 3 2 2 7)	14	27	13	6.9997
	10	1	(2 2 2 2 2 2 2)	(9 2 2 3 2 2 8)	14	28	14	6.9997
	1	2	(3 2 2 2 2 2 2)	(5 2 2 2 2 2 2)	15	17	2	6.9994
	1	3	(4 2 2 2 2 2 2)	(5 2 2 2 2 3 3)	16	19	3	6.9998
	1	4	(4 2 2 2 2 2 2)	(4 2 2 2 2 2 2)	16	16	0	6.9998
	1	10	(4 2 2 2 2 2 2)	(4 2 2 2 2 2 2)	16	16	0	6.9998
SPLIT	$c-K$ **price ratio**		c	K	$\sum c$	$\sum K$	$\sum B$	θ
	c	K						
λ**= 15**	1	1	(5 2 2 2 2 2 2)	(5 2 2 2 2 2 2)	17	17	0	14.998
CV**=1.0**	2	1	(4 2 2 2 2 2 2)	(8 2 2 2 2 2 2)	16	20	4	14.999
	3	1	(4 2 2 2 2 2 2)	(8 2 2 2 2 2 2)	16	20	4	14.999
	4	1	(4 2 2 2 2 2 2)	(8 2 2 2 2 2 2)	16	20	4	14.999
	10	1	(4 2 2 2 2 2 2)	(9 2 2 2 2 2 2)	16	21	5	14.946
	1	2	(5 2 2 2 2 2 2)	(5 2 2 2 2 2 2)	17	17	0	14.999
	1	3	(5 2 2 2 2 2 2)	(5 2 2 2 2 2 2)	17	17	0	14.999
	1	4	(5 2 2 2 2 2 2)	(5 2 2 2 2 2 2)	17	17	0	14.999
	1	10	(4 2 2 3 3 3 3)	(5 2 2 3 3 3 3)	20	21	1	15.000
MERGE	$c-K$ **price ratio**		c	K	$\sum c$	$\sum K$	$\sum B$	θ
	c	K						
λ **= 3**	1	1	(1 1 1 1 2 2 2)	(2 2 2 2 2 2 2)	10	14	4	2.9991
CV**=1.0**	2	1	(1 1 1 1 2 2 1)	(2 2 2 2 2 2 5)	9	17	8	2.9984
	3	1	(1 1 1 1 1 1 1)	(2 2 2 2 3 3 6)	7	20	13	2.9984
	4	1	(1 1 1 1 1 1 1)	(2 2 2 2 3 3 6)	7	20	13	2.9984
	10	1	(1 1 1 1 1 1 1)	(2 2 2 2 4 4 6)	7	22	15	2.9992
	1	2	(1 1 1 1 2 2 2)	(2 2 2 2 2 2 2)	10	14	4	2.9991
	1	3	(1 1 1 1 2 2 2)	(2 2 2 2 2 2 2)	10	14	4	2.9991
	1	4	(1 1 1 1 2 2 2)	(2 2 2 2 2 2 2)	10	14	4	2.9991
	1	10	(2 1 1 1 2 2 2)	(2 2 2 2 2 2 2)	11	14	3	2.9994

Table B.6: Different server-buffer price ratio, N=15

SERIES	$c-K$ **price ratio**		c	K	$\sum c$	$\sum K$	$\sum B$	θ
	c	K						
λ**= 15**	1	1	(2 2 2 2 2 2 2 5 2 2 2 2 2 5 2)	(2 2 2 2 2 2 2 6 2 2 2 2 2 5 2)	36	37	1	14.998
CV**=1.0**	2	1	(2 2 2 2 2 2 2 5 2 2 2 2 2 3 2)	(2 2 2 2 2 2 2 7 2 2 2 2 2 3 2)	34	36	2	14.998
	3	1	(2 2 2 2 2 2 2 5 2 2 2 2 2 3 2)	(2 2 2 2 2 2 2 7 2 2 2 2 2 3 2)	34	36	2	14.999
	4	1	(5 2 2 2 2 2 2 5 2 2 2 2 2 2 2)	(7 2 2 2 2 2 2 7 2 2 2 2 2 2 2)	36	40	4	14.999
	10	1	(2 2 2 2 2 2 2 2 2 2 2 2 2 4 2)	(4 2 2 4 4 4 4 4 2 2 2 4 2 7 5)	32	52	20	14.989
	1	2	(2 2 2 2 2 2 2 2 2 2 2 2 2 5 2)	(2 2 2 2 2 2 2 2 2 2 2 2 2 5 2)	33	33	0	14.992
	1	3	(5 5 5 5 5 5 5 5 5 2 5 5 5 5 4)	(5 5 5 5 5 5 5 5 5 5 5 5 5 5 6)	71	76	5	14.925
	1	4	(5 5 5 5 5 5 5 5 5 2 5 5 5 5 4)	(5 5 5 5 5 5 5 5 5 5 5 5 5 5 6)	71	76	5	14.929
	1	10	(5 5 5 5 5 5 5 5 5 2 5 5 5 5 4)	(5 5 5 5 5 5 5 5 5 5 5 5 5 5 6)	71	76	5	14.929
SPLIT	$c-K$ **price ratio**		c	K	$\sum c$	$\sum K$	$\sum B$	θ
	c	K						
λ**= 3**	1	1	(2 2 2 1 1 1 1 1 1 1 1 1 1 1 1)	(2 2 2 2 2 2 2 1 1 1 1 1 1 1 2)	18	23	5	2.9971
CV**=1.0**	2	1	(1 2 2 1 1 1 1 1 1 1 1 1 1 1 1)	(6 2 2 2 2 2 2 1 1 1 1 1 1 1 1)	17	26	9	2.9968
	3	1	(1 1 1 1 1 1 1 1 1 1 1 1 1 1 1)	(6 3 3 2 2 2 2 1 1 1 1 1 1 1 1)	15	28	13	2.9962
	4	1	(1 1 1 1 1 1 1 1 1 1 1 1 1 1 1)	(6 3 3 2 2 2 2 1 1 1 1 1 1 1 1)	15	28	13	2.9962
	10	1	(1 1 1 1 1 1 1 1 1 1 1 1 1 1 1)	(7 4 4 2 2 2 2 2 2 2 2 2 2 2 2)	15	39	24	2.9991
	1	2	(2 2 2 1 1 1 1 1 1 1 1 1 1 1 1)	(2 2 2 2 2 2 2 1 1 1 1 1 1 1 2)	18	23	5	2.9971
	1	3	(2 2 2 1 1 1 1 1 1 1 1 1 1 1 1)	(2 2 2 2 2 2 2 1 1 1 1 1 1 1 2)	18	23	5	2.9971
	1	4	(2 2 2 1 1 1 1 1 1 1 1 1 1 1 1)	(2 2 2 2 2 2 2 1 1 1 1 1 1 1 2)	18	23	5	2.9971
	1	10	(3 2 2 1 1 1 1 1 1 1 1 1 1 1 1)	(3 2 2 2 2 2 2 1 1 1 1 1 1 1 1)	19	23	4	2.9973
MERGE	$c-K$ **price ratio**		c	K	$\sum c$	$\sum K$	$\sum B$	θ
	c	K						
λ**= 7**	1	1	(1 1 1 1 1 1 1 1 2 2 2 2 2 2 2)	(2 2 2 2 2 2 2 2 2 2 2 2 2 2 2)	22	30	8	6.9964
CV**=1.0**	2	1	(1 1 1 1 1 1 1 1 1 1 1 1 1 1 2)	(2 2 2 2 2 2 2 2 3 3 3 3 6 6 6)	16	46	30	6.9923
	3	1	(1 1 1 1 1 1 1 1 1 1 1 1 1 1 2)	(2 2 2 2 2 2 2 2 4 4 4 4 7 7 7)	16	53	37	6.9962
	4	1	(1 1 1 1 1 1 1 1 1 1 1 1 1 1 2)	(3 2 2 2 2 2 2 2 4 4 4 4 7 7 7)	16	54	38	6.9967
	10	1	(1 1 1 1 1 1 1 1 1 1 1 1 1 1 2)	(3 3 3 3 3 3 3 3 4 4 4 4 7 7 8)	16	62	46	6.9986
	1	2	(1 1 1 1 1 1 1 1 2 2 2 2 2 2 3)	(2 2 2 2 2 2 2 2 2 2 2 2 2 2 3)	23	31	8	6.9964
	1	3	(2 1 1 1 1 1 1 1 2 2 2 2 2 2 3)	(2 2 2 2 2 2 2 2 2 2 2 2 2 2 3)	24	31	7	6.9969
	1	4	(2 1 1 1 1 1 1 1 2 2 2 2 3 3 2)	(2 2 2 2 2 2 2 2 2 2 2 2 3 3 2)	25	32	7	6.9977
	1	10	(2 2 2 2 2 2 2 2 2 2 2 2 2 2 2)	(2 2 2 2 2 2 2 2 2 2 2 2 2 2 2)	30	30	0	6.9999

Table B.7: Different methods, N=3

SERIES							
λ	**Method**	c	K	$\sum c$	$\sum K$	$\sum B$	θ
3	BCAP	(2 2 2)	(2 2 2)	6	6	0	2.9954
	BAP	(2 2 2)	(3 3 3)	6	9	3	2.9990
	CAP	(3 3 3)	(2 2 2)	9	6	-3	2.9998
7	BCAP	(2 2 2)	(2 2 2)	6	6	0	6.7948
	BAP	(2 2 2)	(6 6 6)	6	18	12	6.9950
	CAP	(2 4 4)	(2 2 2)	10	6	-4	6.9977
15	BCAP	(4 2 3)	(8 2 3)	9	13	4	14.999
	BAP	(4 2 3)	(6 24 10)	9	40	31	14.993
	CAP	(2 3 6)	(8 2 3)	11	13	2	14.998
SPLIT							
λ	**Method**	c	K	$\sum c$	$\sum K$	$\sum B$	θ
3	BCAP	(2 2 2)	(2 2 2)	6	6	0	2.9998
	BAP	(2 2 2)	(3 2 2)	6	7	1	2.9990
	CAP	(3 2 2)	(2 2 2)	7	6	-1	2.9998
7	BCAP	(2 2 2)	(2 2 3)	6	7	1	6.9993
	BAP	(2 2 2)	(6 3 3)	6	12	6	6.9970
	CAP	(3 2 2)	(2 2 3)	7	7	0	6.9998
15	BCAP	(3 2 2)	(3 2 2)	7	7	0	14.999
	BAP	(3 2 2)	(12 8 8)	7	28	21	14.997
	CAP	(9 2 2)	(3 2 2)	13	7	-6	15.000
MERGE							
λ	**Method**	c	K	$\sum c$	$\sum K$	$\sum B$	θ
3	BCAP	(2 2 2)	(2 2 2)	6	6	0	2.9998
	BAP	(2 2 2)	(2 2 3)	6	7	1	3.0000
	CAP	(2 2 3)	(2 2 2)	7	6	-1	2.9999
7	BCAP	(2 3 3)	(3 3 3)	8	9	1	6.9999
	BAP	(2 3 3)	(3 3 4)	8	10	2	6.9990
	CAP	(2 3 2)	(3 3 3)	7	9	2	6.9999
15	BCAP	(2 2 5)	(7 2 5)	9	14	5	14.999
	BAP	(2 2 5)	(7 7 5)	9	19	10	14.997
	CAP	(3 2 5)	(7 2 5)	10	14	4	15.000

Table B.8: Different methods, N=7

SERIES							
λ	**Method**	c	K	$\sum c$	$\sum K$	$\sum B$	θ
3	BCAP	(2 2 2 2 2 2 2)	(4 2 2 2 2 2 2)	14	16	2	2.9998
	BAP	(2 2 2 2 2 2 2)	(3 3 3 3 3 3 3)	14	21	7	2.9980
	CAP	(3 2 2 2 2 2 2)	(4 2 2 2 2 2 2)	15	16	1	3.0000
7	BCAP	(2 2 2 2 2 2 2)	(8 2 2 3 2 2 6)	14	25	11	6.9992
	BAP	(2 2 2 2 2 2 2)	(6 6 6 6 6 6 6)	14	42	28	6.9890
	CAP	(4 2 2 2 2 2 2)	(8 2 2 3 2 2 6)	16	25	9	7.0000
15	BCAP	(3 2 2 2 2 2 2)	(3 3 2 2 2 2 3)	15	17	2	15.000
	BAP	(3 2 2 2 2 2 2)	(9 24 24 23 24 24 24)	15	152	137	14.977
	CAP	(3 3 2 2 2 7 2)	(3 3 2 2 2 2 3)	21	17	-4	14.996
SPLIT							
λ	**Method**	c	K	$\sum c$	$\sum K$	$\sum B$	θ
3	BCAP	(2 2 2 1 1 1 1)	(2 2 2 2 2 2 2)	10	14	4	2.9976
	BAP	(2 2 2 1 1 1 1)	(3 2 2 2 2 2 2)	10	15	5	2.9980
	CAP	(2 2 2 1 1 1 1)	(2 2 2 2 2 2 2)	10	14	4	2.9976
7	BCAP	(3 2 2 2 2 2 2)	(3 2 2 2 2 2 2)	15	15	0	6.9986
	BAP	(3 2 2 2 2 2 2)	(4 3 3 2 2 2 2)	15	18	3	6.9970
	CAP	(3 2 2 2 2 2 2)	(3 2 2 2 2 2 2)	15	15	0	7.0000
15	BCAP	(5 2 2 2 2 2 2)	(5 2 2 2 2 2 2)	17	17	0	15.000
	BAP	(5 2 2 2 2 2 2)	(5 7 7 3 3 3 3)	17	31	14	14.993
	CAP	(3 2 2 2 2 2 2)	(5 2 2 2 2 2 2)	15	17	2	14.999
MERGE							
λ	**Method**	c	K	$\sum c$	$\sum K$	$\sum B$	θ
3	BCAP	(3 2 2 2 3 3 4)	(3 3 3 5 3 3 4)	19	24	5	15.000
	BAP	(3 2 2 2 3 3 4)	(3 3 3 3 4 4 7)	19	27	8	14.994
	CAP	(2 2 2 2 2 2 3)	(3 3 3 5 3 3 4)	15	24	9	15.000
7	BCAP	(2 2 2 2 2 2 3)	(2 2 2 2 2 2 3)	15	15	0	6.9988
	BAP	(2 2 2 2 2 2 3)	(3 3 3 3 4 4 5)	15	25	10	6.9970
	CAP	(2 2 2 2 3 3 2)	(2 2 2 2 2 2 3)	16	15	-1	6.9992
15	BCAP	(1 1 1 1 2 2 2)	(2 2 2 2 2 2 2)	10	14	4	2.9991
	BAP	(1 1 1 1 2 2 2)	(2 2 2 2 2 2 3)	10	15	5	2.9980
	CAP	(1 1 1 1 2 2 3)	(2 2 2 2 2 2 2)	11	14	3	2.9991

Table B.9: Different methods, N=15

SERIES							
λ	**Method**	c	K	$\sum c$	$\sum K$	$\sum B$	θ
3	BCAP	(2 2 2 2 2 2 2 2 2 2 2 2 2 2 2)	(2 2 2 2 2 2 2 2 2 2 2 2 2 2 3)	30	31	1	2.9813
	BAP	(2 2 2 2 2 2 2 2 2 2 2 2 2 2 2)	(3 3 3 3 3 3 3 3 3 3 3 3 3 3 3)	30	45	15	2.9950
	CAP	(2 2 2 2 2 2 2 2 2 2 2 2 2 2 3)	(2 2 2 2 2 2 2 2 2 2 2 2 2 2 3)	31	31	0	2.9999
7	BCAP	(2 2 2 2 2 2 2 2 2 2 2 2 2 2 2)	(2 2 2 2 2 2 2 2 2 2 2 2 2 2 2)	30	30	0	6.3387
	BAP	(2 2 2 2 2 2 2 2 2 2 2 2 2 2 2)	(6 6 6 6 6 6 6 6 6 6 6 6 6 6 6)	30	90	60	6.9780
	CAP	(2 2 2 2 2 2 2 2 2 2 2 2 2 7 5)	(2 2 2 2 2 2 2 2 2 2 2 2 2 2 2)	38	30	-8	6.9068
15	BCAP	(3 3 3 3 2 2 3 3 2 2 3 3 2 2 2)	(3 3 3 3 3 3 3 3 2 2 3 3 2 2 2)	38	40	2	15.000
	BAP	(3 3 3 3 2 2 3 3 2 2 3 3 2 2 2)	(11 11 11 11 23 23 11 11 23 23 10 10 23 24 24)	38	249	211	14.964
	CAP	(2 2 2 2 2 2 2 2 2 8 2 2 2 2 2)	(3 3 3 3 3 3 3 3 2 2 3 3 2 2 2)	36	40	4	14.999
SPLIT							
λ	**Method**	c	K	$\sum c$	$\sum K$	$\sum B$	θ
3	BCAP	(2 2 2 1 1 1 1 1 1 1 1 1 1 1 1)	(2 2 2 2 2 2 2 1 1 1 1 1 1 1 2)	18	23	5	2.9971
	BAP	(2 2 2 1 1 1 1 1 1 1 1 1 1 1 1)	(3 2 2 2 2 2 2 1 1 1 1 1 1 1 1)	18	23	5	2.9940
	CAP	(2 2 2 1 1 1 1 1 1 1 1 1 1 1 1)	(2 2 2 2 2 2 2 1 1 1 1 1 1 1 2)	18	23	5	2.9974
7	BCAP	(2 2 2 2 2 2 2 1 1 1 1 1 1 1 2)	(2 2 2 2 2 2 2 2 2 2 2 2 2 2 2)	23	30	7	6.9890
	BAP	(2 2 2 2 2 2 2 1 1 1 1 1 1 1 2)	(6 3 3 2 2 2 2 2 2 2 2 2 2 2 2)	23	36	13	6.9920
	CAP	(2 2 2 2 2 2 2 1 1 1 1 1 1 1 2)	(2 2 2 2 2 2 2 2 2 2 2 2 2 2 2)	23	30	7	6.9890
15	BCAP	(5 2 2 2 2 2 2 2 2 2 2 2 2 2 2)	(5 2 2 2 2 2 2 2 2 2 2 2 2 2 2)	33	33	0	14.995
	BAP	(5 2 2 2 2 2 2 2 2 2 2 2 2 2 2)	(5 7 7 3 3 3 3 2 2 2 2 2 2 2 2)	33	47	14	14.991
	CAP	(5 2 5 2 2 2 2 2 2 2 2 2 2 2 2)	(5 2 2 2 2 2 2 2 2 2 2 2 2 2 2)	36	33	-3	14.999
MERGE							
λ	**Method**	c	K	$\sum c$	$\sum K$	$\sum B$	θ
3	BCAP	(1 1 1 1 1 1 1 1 1 1 1 1 2 2 2)	(1 1 1 1 1 1 1 1 2 2 2 2 2 2 2)	18	22	4	2.9969
	BAP	(1 1 1 1 1 1 1 1 1 1 1 1 2 2 2)	(1 1 1 1 1 1 1 1 2 2 2 2 2 2 3)	18	23	5	2.9940
	CAP	(1 1 1 1 1 1 1 1 1 1 1 1 2 2 3)	(1 1 1 1 1 1 1 1 2 2 2 2 2 2 2)	19	22	3	2.9970
7	BCAP	(1 1 1 1 1 1 1 1 2 2 2 2 2 2 2)	(2 2 2 2 2 2 2 2 2 2 2 2 2 2 2)	22	30	8	6.9964
	BAP	(1 1 1 1 1 1 1 1 2 2 2 2 2 2 2)	(2 2 2 2 2 2 2 2 2 2 2 2 3 3 6)	22	36	14	6.9920
	CAP	(1 1 1 1 1 1 1 1 2 2 2 2 3 2 2)	(2 2 2 2 2 2 2 2 2 2 2 2 2 2 2)	23	30	7	6.9966
15	BCAP	(2 2 2 2 2 2 2 2 2 2 2 2 3 3 4)	(2 2 2 2 2 2 2 2 3 3 3 5 3 3 4)	34	40	6	14.997
	BAP	(2 2 2 2 2 2 2 2 2 2 2 2 3 3 4)	(2 2 2 2 2 2 2 2 3 3 3 3 4 4 7)	34	43	9	14.991
	CAP	(2 2 2 2 2 2 2 2 2 2 2 2 2 2 3)	(2 2 2 2 2 2 2 2 3 3 3 5 3 3 4)	31	40	9	14.997

Appendix C

Complete results: complete enumeration

The boldfaced settings in the following tables represents the optimal buffer and server configuration for a given total number of buffers and servers.

Table C.1: Complete enumeration, $N = 3$, max.$\sum c = 6$, max.$\sum K = 9$

c	K	θ	$\sum c$	$\sum K$	c	K	θ	$\sum c$	$\sum K$
(1 1 1)	**(1 1 1)**	**2.6350**	**3**	**3**	(1 1 1)	(2 4 2)	2.9116	3	8
(1 1 1)	(1 1 2)	2.6951	3	4	(1 1 1)	(4 2 2)	2.9345	3	8
(1 1 1)	(1 2 1)	2.6946	3	4	**(1 1 1)**	**(3 2 3)**	**2.9441**	**3**	**8**
(1 1 1)	**(2 1 1)**	**2.7485**	**3**	**4**	(1 1 1)	(3 3 2)	2.9440	3	8
(1 1 1)	(1 2 2)	2.7582	3	5	(1 1 1)	(2 3 3)	2.9262	3	8
(1 1 1)	**(2 1 2)**	**2.8156**	**3**	**5**	(1 1 1)	(1 1 7)	2.7165	3	9
(1 1 1)	(2 2 1)	2.8148	3	5	(1 1 1)	(1 7 1)	2.7166	3	9
(1 1 1)	(3 1 1)	2.7107	3	5	(1 1 1)	(7 1 1)	2.7952	3	9
(1 1 1)	(1 3 1)	2.7105	3	5	(1 1 1)	(1 2 6)	2.7815	3	9
(1 1 1)	(1 1 3)	2.7814	3	5	(1 1 1)	(1 6 2)	2.7815	3	9
(1 1 1)	(1 1 4)	2.7149	3	6	(1 1 1)	(2 6 1)	2.8407	3	9
(1 1 1)	(1 4 1)	2.7149	3	6	(1 1 1)	(2 1 6)	2.8407	3	9
(1 1 1)	(4 1 1)	2.7911	3	6	(1 1 1)	(6 1 2)	2.8650	3	9
(1 1 1)	(1 2 3)	2.7751	3	6	(1 1 1)	(6 2 1)	2.8642	3	9
(1 1 1)	(1 3 2)	2.7751	3	6	(1 1 1)	(1 3 5)	2.7985	3	9
(1 1 1)	(2 3 1)	2.8333	3	6	(1 1 1)	(1 5 3)	2.7985	3	9
(1 1 1)	(2 1 3)	2.8338	3	6	(1 1 1)	(3 5 1)	2.8764	3	9
(1 1 1)	(3 1 2)	2.8506	3	6	(1 1 1)	(3 1 5)	2.8765	3	9
(1 1 1)	(3 2 1)	2.8498	3	6	(1 1 1)	(5 1 3)	2.8834	3	9
(1 1 1)	**(2 2 2)**	**2.8862**	**3**	**6**	(1 1 1)	(5 3 1)	2.8828	3	9
(1 1 1)	(1 1 5)	2.7161	3	7	(1 1 1)	(1 4 4)	2.8020	3	9
(1 1 1)	(1 5 1)	2.7161	3	7	(1 1 1)	(4 1 4)	2.8857	3	9
(1 1 1)	(5 1 1)	2.7941	3	7	(1 1 1)	(4 4 1)	2.8854	3	9
(1 1 1)	(1 2 4)	2.7798	3	7	(1 1 1)	(2 2 5)	2.9133	3	9
(1 1 1)	(1 4 2)	2.7798	3	7	(1 1 1)	(2 5 2)	2.9133	3	9
(1 1 1)	(2 1 4)	2.8389	3	7	(1 1 1)	(5 2 2)	2.9378	3	9
(1 1 1)	(2 4 1)	2.8387	3	7	(1 1 1)	(3 2 4)	2.9501	3	9
(1 1 1)	(4 1 2)	2.8610	3	7	(1 1 1)	(3 4 2)	2.9501	3	9
(1 1 1)	(4 2 1)	2.8601	3	7	(1 1 1)	(2 4 3)	2.9320	3	9
(1 1 1)	(1 3 3)	2.7924	3	7	(1 1 1)	(2 3 4)	2.9320	3	9
(1 1 1)	(3 1 3)	2.8695	3	7	(1 1 1)	(4 3 2)	2.9554	3	9
(1 1 1)	(3 3 1)	2.8690	3	7	(1 1 1)	(4 2 3)	2.9554	3	9
(1 1 1)	(2 2 3)	2.9060	3	7	**(1 1 1)**	**(3 3 3)**	**2.9652**	**3**	**9**
(1 1 1)	(2 3 2)	2.9059	3	7	(1 1 2)	(1 1 2)	2.7158	4	4
(1 1 1)	**(3 2 2)**	**2.9234**	**3**	**7**	(1 1 2)	(1 2 2)	2.7808	4	5
(1 1 1)	(1 1 6)	2.7164	3	8	(1 1 2)	(2 1 2)	2.8400	4	5
(1 1 1)	(1 6 1)	2.7165	3	8	(1 1 2)	(1 1 3)	2.7164	4	5
(1 1 1)	(6 1 1)	2.7949	3	8	(1 1 2)	(1 1 4)	2.7165	4	6
(1 1 1)	(1 2 5)	2.7811	3	8	(1 1 2)	(1 2 3)	2.7815	4	6
(1 1 1)	(1 5 2)	2.7811	3	8	(1 1 2)	(1 3 2)	2.7982	4	6
(1 1 1)	(2 1 5)	2.8403	3	8	(1 1 2)	(2 1 3)	2.8408	4	6
(1 1 1)	(2 5 1)	2.8403	3	8	(1 1 2)	(3 1 2)	2.8761	4	6
(1 1 1)	(5 1 2)	2.8641	3	8	(1 1 2)	(2 2 2)	2.9129	4	6
(1 1 1)	(5 2 1)	2.8632	3	8	(1 1 2)	(1 1 5)	2.7165	4	7
(1 1 1)	(1 3 4)	2.7972	3	8	(1 1 2)	(1 2 4)	2.7816	4	7
(1 1 1)	(1 4 3)	2.7972	3	8	(1 1 2)	(1 4 2)	2.8030	4	7
(1 1 1)	(3 1 4)	2.8749	3	8	(1 1 2)	(2 1 4)	2.8409	4	7
(1 1 1)	(3 4 1)	2.8747	3	8	(1 1 2)	(4 1 2)	2.8869	4	7
(1 1 1)	(4 1 3)	2.8802	3	8	(1 1 2)	(1 3 3)	2.7989	4	7
(1 1 1)	(4 3 1)	2.8797	3	8	(1 1 2)	(3 1 3)	2.8770	4	7
(1 1 1)	(2 2 4)	2.9117	3	8	(1 1 2)	(2 2 3)	2.9138	4	7

Table C.2: Complete enumeration, $N = 3$, max.$\sum c = 6$, max.$\sum K = 9$

c	K	θ	$\sum c$	$\sum K$	c	K	θ	$\sum c$	$\sum K$
(1 1 2)	(2 3 2)	2.9334	4	7	(1 2 1)	(2 4 1)	2.8409	4	7
(1 1 2)	(3 2 2)	2.9515	4	7	(1 2 1)	(4 2 1)	2.8868	4	7
(1 1 2)	(1 1 6)	2.7165	4	8	(1 2 1)	(1 3 3)	2.7989	4	7
(1 1 2)	(1 2 5)	2.7816	4	8	(1 2 1)	(3 3 1)	2.8770	4	7
(1 1 2)	(1 5 2)	2.8044	4	8	(1 2 1)	(2 2 3)	2.9334	4	7
(1 1 2)	(2 1 5)	2.8409	4	8	(1 2 1)	(2 3 2)	2.9138	4	7
(1 1 2)	(5 1 2)	2.8901	4	8	(1 2 1)	(3 2 2)	2.9515	4	7
(1 1 2)	(1 3 4)	2.7990	4	8	(1 2 1)	(1 6 1)	2.7166	4	8
(1 1 2)	(1 4 3)	2.8038	4	8	(1 2 1)	(1 2 5)	2.8044	4	8
(1 1 2)	(3 1 4)	2.8771	4	8	(1 2 1)	(1 5 2)	2.7816	4	8
(1 1 2)	(4 1 3)	2.8877	4	8	(1 2 1)	(2 5 1)	2.8409	4	8
(1 1 2)	(2 2 4)	2.9140	4	8	(1 2 1)	(5 2 1)	2.8900	4	8
(1 1 2)	(2 4 2)	2.9393	4	8	(1 2 1)	(1 3 4)	2.8038	4	8
(1 1 2)	(4 2 2)	2.9630	4	8	(1 2 1)	(1 4 3)	2.7990	4	8
(1 1 2)	(3 2 3)	2.9525	4	8	(1 2 1)	(3 4 1)	2.8771	4	8
(1 1 2)	(3 3 2)	2.9728	4	8	(1 2 1)	(4 3 1)	2.8877	4	8
(1 1 2)	(2 3 3)	2.9343	4	8	(1 2 1)	(2 2 4)	2.9393	4	8
(1 1 2)	(1 1 7)	2.7165	4	9	(1 2 1)	(2 4 2)	2.9140	4	8
(1 1 2)	(1 2 6)	2.7816	4	9	(1 2 1)	(4 2 2)	2.9630	4	8
(1 1 2)	(1 6 2)	2.8047	4	9	(1 2 1)	(3 2 3)	2.9728	4	8
(1 1 2)	(2 1 6)	2.8409	4	9	(1 2 1)	(3 3 2)	2.9525	4	8
(1 1 2)	(6 1 2)	2.8910	4	9	(1 2 1)	(2 3 3)	2.9343	4	8
(1 1 2)	(1 3 5)	2.7990	4	9	(1 2 1)	(1 7 1)	2.7166	4	9
(1 1 2)	(1 5 3)	2.8051	4	9	(1 2 1)	(1 2 6)	2.8047	4	9
(1 1 2)	(3 1 5)	2.8771	4	9	(1 2 1)	(1 6 2)	2.7816	4	9
(1 1 2)	(5 1 3)	2.8909	4	9	(1 2 1)	(2 6 1)	2.8409	4	9
(1 1 2)	(1 4 4)	2.8039	4	9	(1 2 1)	(6 2 1)	2.8910	4	9
(1 1 2)	(4 1 4)	2.8878	4	9	(1 2 1)	(1 3 5)	2.8051	4	9
(1 1 2)	(2 2 5)	2.9140	4	9	(1 2 1)	(1 5 3)	2.7990	4	9
(1 1 2)	(2 5 2)	2.9410	4	9	(1 2 1)	(3 5 1)	2.8771	4	9
(1 1 2)	(5 2 2)	2.9664	4	9	(1 2 1)	(5 3 1)	2.8909	4	9
(1 1 2)	(3 2 4)	2.9526	4	9	(1 2 1)	(1 4 4)	2.8039	4	9
(1 1 2)	(3 4 2)	2.9791	4	9	(1 2 1)	(4 4 1)	2.8878	4	9
(1 1 2)	(2 4 3)	2.9402	4	9	(1 2 1)	(2 2 5)	2.9410	4	9
(1 1 2)	(2 3 4)	2.9344	4	9	(1 2 1)	(2 5 2)	2.9140	4	9
(1 1 2)	(4 3 2)	2.9846	4	9	(1 2 1)	(5 2 2)	2.9664	4	9
(1 1 2)	(4 2 3)	2.9640	4	9	(1 2 1)	(3 2 4)	2.9791	4	9
(1 1 2)	(3 3 3)	2.9738	4	9	(1 2 1)	(3 4 2)	2.9526	4	9
(1 2 1)	(1 2 1)	2.7158	4	4	(1 2 1)	(2 4 3)	2.9344	4	9
(1 2 1)	(1 2 2)	2.7808	4	5	(1 2 1)	(2 3 4)	2.9402	4	9
(1 2 1)	(2 2 1)	2.8399	4	5	(1 2 1)	(4 3 2)	2.9640	4	9
(1 2 1)	(1 3 1)	2.7165	4	5	(1 2 1)	(4 2 3)	2.9846	4	9
(1 2 1)	(2 2 2)	2.9129	4	6	(1 2 1)	(3 3 3)	2.9738	4	9
(1 2 1)	(1 4 1)	2.7166	4	6	**(2 1 1)**	**(2 1 1)**	**2.7935**	**4**	**4**
(1 2 1)	(1 2 3)	2.7982	4	6	**(2 1 1**)	**(2 1 2)**	**2.8635**	**4**	**5**
(1 2 1)	(1 3 2)	2.7815	4	6	(2 1 1)	(2 2 1)	2.8626	4	5
(1 2 1)	(2 3 1)	2.8408	4	6	(2 1 1)	(3 1 1)	2.7950	4	5
(1 2 1)	(3 2 1)	2.8761	4	6	(2 1 1)	(4 1 1)	2.7953	4	6
(1 2 1)	(2 2 2)	2.9129	4	6	(2 1 1)	(2 3 1)	2.8822	4	6
(1 2 1)	(1 5 1)	2.7166	4	7	(2 1 1)	(2 1 3)	2.8827	4	6
(1 2 1)	(1 2 4)	2.8030	4	7	(2 1 1)	(3 1 2)	2.8651	4	6
(1 2 1)	(1 4 2)	2.7816	4	7	(2 1 1)	(3 2 1)	2.8643	4	6

Table C.3: Complete enumeration, $N = 3$, max.$\sum c = 6$, max.$\sum K = 9$

c	K	θ	$\sum c$	$\sum K$	c	K	θ	$\sum c$	$\sum K$
(2 1 1)	**(2 2 2)**	**2.9372**	**4**	**6**	(1 2 2)	(1 4 2)	2.8049	5	7
(2 1 1)	(5 1 1)	2.7953	4	7	(1 2 2)	(1 3 3)	2.8055	5	7
(2 1 1)	(2 1 4)	2.8882	4	7	(1 2 2)	(2 2 3)	2.9416	5	7
(2 1 1)	(2 4 1)	2.8880	4	7	(1 2 2)	(2 3 2)	2.9416	5	7
(2 1 1)	(4 1 2)	2.8654	4	7	(1 2 2)	(3 2 2)	2.9806	5	7
(2 1 1)	(4 2 1)	2.8645	4	7	(1 2 2)	(1 2 5)	2.8049	5	8
(2 1 1)	(3 1 3)	2.8844	4	7	(1 2 2)	(1 5 2)	2.8049	5	8
(2 1 1)	(3 3 1)	2.8839	4	7	(1 2 2)	(1 3 4)	2.8056	5	8
(2 1 1)	**(2 2 3)**	**2.9582**	**4**	**7**	(1 2 2)	(1 4 3)	2.8056	5	8
(2 1 1)	(2 3 2)	2.9581	4	7	(1 2 2)	(2 2 4)	2.9417	5	8
(2 1 1)	(3 2 2)	2.9389	4	7	(1 2 2)	(2 4 2)	2.9417	5	8
(2 1 1)	(6 1 1)	2.7953	4	8	(1 2 2)	(4 2 2)	2.9925	5	8
(2 1 1)	(2 1 5)	2.8898	4	8	(1 2 2)	(3 2 3)	2.9816	5	8
(2 1 1)	(2 5 1)	2.8897	4	8	(1 2 2)	(3 3 2)	2.9816	5	8
(2 1 1)	(5 1 2)	2.8654	4	8	(1 2 2)	(2 3 3)	2.9425	5	8
(2 1 1)	(5 2 1)	2.8645	4	8	(1 2 2)	(1 2 6)	2.8049	5	9
(2 1 1)	(3 1 4)	2.8899	4	8	(1 2 2)	(1 6 2)	2.8049	5	9
(2 1 1)	(3 4 1)	2.8897	4	8	(1 2 2)	(1 3 5)	2.8056	5	9
(2 1 1)	(4 1 3)	2.8847	4	8	(1 2 2)	(1 5 3)	2.8056	5	9
(2 1 1)	(4 3 1)	2.8841	4	8	(1 2 2)	(1 4 4)	2.8057	5	9
(2 1 1)	(2 2 4)	2.9643	4	8	(1 2 2)	(2 2 5)	2.9417	5	9
(2 1 1)	(2 4 2)	2.9642	4	8	(1 2 2)	(2 5 2)	2.9417	5	9
(2 1 1)	(4 2 2)	2.9392	4	8	**(1 2 2)**	**(5 2 2)**	**2.9961**	**5**	**9**
(2 1 1)	(3 2 3)	2.9600	4	8	(1 2 2)	(3 2 4)	2.9817	5	9
(2 1 1)	(3 3 2)	2.9599	4	8	(1 2 2)	(3 4 2)	2.9817	5	9
(2 1 1)	**(2 3 3)**	**2.9796**	**4**	**8**	(1 2 2)	(2 4 3)	2.9427	5	9
(2 1 1)	(7 1 1)	2.7953	4	9	(1 2 2)	(2 3 4)	2.9427	5	9
(2 1 1)	(2 6 1)	2.8902	4	9	(1 2 2)	(4 3 2)	2.9935	5	9
(2 1 1)	(2 1 6)	2.8903	4	9	(1 2 2)	(4 2 3)	2.9935	5	9
(2 1 1)	(6 1 2)	2.8654	4	9	(1 2 2)	(3 3 3)	2.9826	5	9
(2 1 1)	(6 2 1)	2.8646	4	9	(2 1 2)	(2 1 2)	2.8894	5	5
(2 1 1)	(3 5 1)	2.8914	4	9	(2 1 2)	(2 1 3)	2.8903	5	6
(2 1 1)	(3 1 5)	2.8915	4	9	(2 1 2)	(3 1 2)	2.8911	5	6
(2 1 1)	(5 1 3)	2.8847	4	9	**(2 1 2)**	**(2 2 2)**	**2.9657**	**5**	**6**
(2 1 1)	(5 3 1)	2.8842	4	9	(2 1 2)	(2 1 4)	2.8904	5	7
(2 1 1)	(4 1 4)	2.8902	4	9	(2 1 2)	(4 1 2)	2.8914	5	7
(2 1 1)	(4 4 1)	2.8899	4	9	(2 1 2)	(3 1 3)	2.8920	5	7
(2 1 1)	(2 2 5)	2.9661	4	9	(2 1 2)	(2 2 3)	2.9667	5	7
(2 1 1)	(2 5 2)	2.9661	4	9	**(2 1 2)**	**(2 3 2)**	**2.9874**	**5**	**7**
(2 1 1)	(5 2 2)	2.9392	4	9	(2 1 2)	(3 2 2)	2.9676	5	7
(2 1 1)	(3 2 4)	2.9661	4	9	(2 1 2)	(2 1 5)	2.8904	5	8
(2 1 1)	(3 4 2)	2.9661	4	9	(2 1 2)	(5 1 2)	2.8914	5	8
(2 1 1)	(2 4 3)	2.9859	4	9	(2 1 2)	(3 1 4)	2.8921	5	8
(2 1 1)	**(2 3 4)**	**2.9859**	**4**	**9**	(2 1 2)	(4 1 3)	2.8923	5	8
(2 1 1)	(4 3 2)	2.9602	4	9	(2 1 2)	(2 2 4)	2.9668	5	8
(2 1 1)	(4 2 3)	2.9602	4	9	(2 1 2)	(2 4 2)	2.9938	5	8
(2 1 1)	(3 3 3)	2.9814	4	9	(2 1 2)	(4 2 2)	2.9678	5	8
(1 2 2)	(1 2 2)	2.8040	5	5	(2 1 2)	(3 2 3)	2.9685	5	8
(1 2 2)	(1 2 3)	2.8048	5	6	(2 1 2)	(3 3 2)	2.9893	5	8
(1 2 2)	(1 3 2)	2.8048	5	6	(2 1 2)	(2 3 3)	2.9884	5	8
(1 2 2)	(2 2 2)	2.9406	5	6	(2 1 2)	(2 1 6)	2.8904	5	9
(1 2 2)	(1 2 4)	2.8049	5	7	(2 1 2)	(6 1 2)	2.8914	5	9

Table C.4: Complete enumeration, $N = 3$, max.$\sum c = 6$, max.$\sum K = 9$

c	K	θ	$\sum c$	$\sum K$	c	K	θ	$\sum c$	$\sum K$
(2 1 2)	(3 1 5)	2.8922	5	9	(3 1 1)	(5 1 1)	2.7953	5	7
(2 1 2)	(5 1 3)	2.8923	5	9	(3 1 1)	(4 1 2)	2.8654	5	7
(2 1 2)	(4 1 4)	2.8924	5	9	(3 1 1)	(4 2 1)	2.8646	5	7
(2 1 2)	(2 2 5)	2.9669	5	9	(3 1 1)	(3 1 3)	2.8847	5	7
(2 1 2)	(2 5 2)	2.9957	5	9	(3 1 1)	(3 3 1)	2.8842	5	7
(2 1 2)	(5 2 2)	2.9679	5	9	(3 1 1)	(3 2 2)	2.9392	5	7
(2 1 2)	(3 2 4)	2.9687	5	9	(3 1 1)	(6 1 1)	2.7953	5	8
(2 1 2)	(3 4 2)	2.9957	5	9	(3 1 1)	(5 1 2)	2.8654	5	8
(2 1 2)	(2 4 3)	2.9948	5	9	(3 1 1)	(5 2 1)	2.8646	5	8
(2 1 2)	(2 3 4)	2.9886	5	9	(3 1 1)	(3 1 4)	2.8902	5	8
(2 1 2)	(4 3 2)	2.9896	5	9	(3 1 1)	(3 4 1)	2.8900	5	8
(2 1 2)	(4 2 3)	2.9688	5	9	(3 1 1)	(4 1 3)	2.8847	5	8
(2 1 2)	(3 3 3)	2.9903	5	9	(3 1 1)	(4 3 1)	2.8842	5	8
(2 2 1)	**(2 2 1)**	**2.8894**	**5**	**5**	(3 1 1)	(4 2 2)	2.9392	5	8
(2 2 1)	(2 3 1)	2.8903	5	6	(3 1 1)	(3 2 3)	2.9603	5	8
(2 2 1)	(3 2 1)	2.8911	5	6	(3 1 1)	(3 3 2)	2.9602	5	8
(2 2 1)	(2 2 2)	2.9657	5	6	(3 1 1)	(7 1 1)	2.7953	5	9
(2 2 1)	(2 4 1)	2.8904	5	7	(3 1 1)	(6 1 2)	2.8654	5	9
(2 2 1)	(4 2 1)	2.8913	5	7	(3 1 1)	(6 2 1)	2.8646	5	9
(2 2 1)	(3 3 1)	2.8920	5	7	(3 1 1)	(3 5 1)	2.8917	5	9
(2 2 1)	(2 2 3)	2.9874	5	7	(3 1 1)	(3 1 5)	2.8918	5	9
(2 2 1)	(2 3 2)	2.9667	5	7	(3 1 1)	(5 1 3)	2.8847	5	9
(2 2 1)	(3 2 2)	2.9676	5	7	(3 1 1)	(5 3 1)	2.8842	5	9
(2 2 1)	(2 5 1)	2.8904	5	8	(3 1 1)	(4 1 4)	2.8902	5	9
(2 2 1)	(5 2 1)	2.8914	5	8	(3 1 1)	(4 4 1)	2.8900	5	9
(2 2 1)	(3 4 1)	2.8921	5	8	(3 1 1)	(5 2 2)	2.9392	5	9
(2 2 1)	(4 3 1)	2.8923	5	8	(3 1 1)	(3 2 4)	2.9664	5	9
(2 2 1)	**(2 2 4)**	**2.9938**	**5**	**8**	(3 1 1)	(3 4 2)	2.9664	5	9
(2 2 1)	(2 4 2)	2.9668	5	8	(3 1 1)	(4 3 2)	2.9602	5	9
(2 2 1)	(4 2 2)	2.9678	5	8	(3 1 1)	(4 2 3)	2.9603	5	9
(2 2 1)	(3 2 3)	2.9893	5	8	(3 1 1)	(3 3 3)	2.9818	5	9
(2 2 1)	(3 3 2)	2.9685	5	8	(1 3 1)	(1 3 1)	2.7166	5	5
(2 2 1)	(2 3 3)	2.9884	5	8	(1 3 1)	(1 4 1)	2.7166	5	6
(2 2 1)	(2 6 1)	2.8904	5	9	(1 3 1)	(1 3 2)	2.7816	5	6
(2 2 1)	(6 2 1)	2.8914	5	9	(1 3 1)	(2 3 1)	2.8409	5	6
(2 2 1)	(3 5 1)	2.8922	5	9	(1 3 1)	(1 5 1)	2.7166	5	7
(2 2 1)	(5 3 1)	2.8923	5	9	(1 3 1)	(1 4 2)	2.7816	5	7
(2 2 1)	(4 4 1)	2.8924	5	9	(1 3 1)	(2 4 1)	2.8409	5	7
(2 2 1)	(2 2 5)	2.9957	5	9	(1 3 1)	(1 3 3)	2.7990	5	7
(2 2 1)	(2 5 2)	2.9669	5	9	(1 3 1)	(3 3 1)	2.8771	5	7
(2 2 1)	(5 2 2)	2.9679	5	9	(1 3 1)	(2 3 2)	2.9140	5	7
(2 2 1)	(3 2 4)	2.9957	5	9	(1 3 1)	(1 6 1)	2.7166	5	8
(2 2 1)	(3 4 2)	2.9687	5	9	(1 3 1)	(1 5 2)	2.7816	5	8
(2 2 1)	(2 4 3)	2.9886	5	9	(1 3 1)	(2 5 1)	2.8409	5	8
(2 2 1)	(2 3 4)	2.9948	5	9	(1 3 1)	(1 3 4)	2.8039	5	8
(2 2 1)	(4 3 2)	2.9688	5	9	(1 3 1)	(1 4 3)	2.7990	5	8
(2 2 1)	(4 2 3)	2.9896	5	9	(1 3 1)	(3 4 1)	2.8771	5	8
(2 2 1)	(3 3 3)	2.9903	5	9	(1 3 1)	(4 3 1)	2.8879	5	8
(3 1 1)	(3 1 1)	2.7953	5	5	(1 3 1)	(2 4 2)	2.9140	5	8
(3 1 1)	(4 1 1)	2.7953	5	6	(1 3 1)	(3 3 2)	2.9526	5	8
(3 1 1)	(3 1 2)	2.8654	5	6	(1 3 1)	(2 3 3)	2.9344	5	8
(3 1 1)	(3 2 1)	2.8645	5	6	(1 3 1)	(1 7 1)	2.7166	5	9

Table C.5: Complete enumeration, $N = 3$, max.$\sum c = 6$, max.$\sum K = 9$

c	K	θ	$\sum c$	$\sum K$	c	K	θ	$\sum c$	$\sum K$
(1 3 1)	(1 6 2)	2.7816	5	9	(1 1 4)	(2 1 4)	2.8409	6	7
(1 3 1)	(2 6 1)	2.8409	5	9	(1 1 4)	(1 1 6)	2.7165	6	8
(1 3 1)	(1 3 5)	2.8052	5	9	(1 1 4)	(1 2 5)	2.7816	6	8
(1 3 1)	(1 5 3)	2.7990	5	9	(1 1 4)	(2 1 5)	2.8409	6	8
(1 3 1)	(3 5 1)	2.8771	5	9	(1 1 4)	(1 3 4)	2.7990	6	8
(1 3 1)	(5 3 1)	2.8911	5	9	(1 1 4)	(3 1 4)	2.8771	6	8
(1 3 1)	(1 4 4)	2.8039	5	9	(1 1 4)	(2 2 4)	2.9140	6	8
(1 3 1)	(4 4 1)	2.8879	5	9	(1 1 4)	(1 1 7)	2.7165	6	9
(1 3 1)	(2 5 2)	2.9140	5	9	(1 1 4)	(1 2 6)	2.7816	6	9
(1 3 1)	(3 4 2)	2.9526	5	9	(1 1 4)	(2 1 6)	2.8409	6	9
(1 3 1)	(2 4 3)	2.9344	5	9	(1 1 4)	(1 3 5)	2.7990	6	9
(1 3 1)	(2 3 4)	2.9404	5	9	(1 1 4)	(3 1 5)	2.8771	6	9
(1 3 1)	(4 3 2)	2.9641	5	9	(1 1 4)	(1 4 4)	2.8039	6	9
(1 3 1)	(3 3 3)	2.9740	5	9	(1 1 4)	(4 1 4)	2.8879	6	9
(1 1 3)	(1 1 3)	2.7165	5	5	(1 1 4)	(2 2 5)	2.9140	6	9
(1 1 3)	(1 1 4)	2.7165	5	6	(1 1 4)	(3 2 4)	2.9526	6	9
(1 1 3)	(1 2 3)	2.7816	5	6	(1 1 4)	(2 3 4)	2.9344	6	9
(1 1 3)	(2 1 3)	2.8409	5	6	(1 4 1)	(1 4 1)	2.7166	6	6
(1 1 3)	(1 1 5)	2.7165	5	7	(1 4 1)	(1 5 1)	2.7166	6	7
(1 1 3)	(1 2 4)	2.7816	5	7	(1 4 1)	(1 4 2)	2.7816	6	7
(1 1 3)	(2 1 4)	2.8409	5	7	(1 4 1)	(2 4 1)	2.8409	6	7
(1 1 3)	(1 3 3)	2.7990	5	7	(1 4 1)	(1 6 1)	2.7166	6	8
(1 1 3)	(3 1 3)	2.8771	5	7	(1 4 1)	(1 5 2)	2.7816	6	8
(1 1 3)	(2 2 3)	2.9140	5	7	(1 4 1)	(2 5 1)	2.8409	6	8
(1 1 3)	(1 1 6)	2.7165	5	8	(1 4 1)	(1 4 3)	2.7990	6	8
(1 1 3)	(1 2 5)	2.7816	5	8	(1 4 1)	(3 4 1)	2.8771	6	8
(1 1 3)	(2 1 5)	2.8409	5	8	(1 4 1)	(2 4 2)	2.9140	6	8
(1 1 3)	(1 3 4)	2.7990	5	8	(1 4 1)	(1 7 1)	2.7166	6	9
(1 1 3)	(1 4 3)	2.8039	5	8	(1 4 1)	(1 6 2)	2.7816	6	9
(1 1 3)	(3 1 4)	2.8771	5	8	(1 4 1)	(2 6 1)	2.8409	6	9
(1 1 3)	(4 1 3)	2.8879	5	8	(1 4 1)	(1 5 3)	2.7990	6	9
(1 1 3)	(2 2 4)	2.9140	5	8	(1 4 1)	(3 5 1)	2.8771	6	9
(1 1 3)	(3 2 3)	2.9526	5	8	(1 4 1)	(1 4 4)	2.8039	6	9
(1 1 3)	(2 3 3)	2.9344	5	8	(1 4 1)	(4 4 1)	2.8879	6	9
(1 1 3)	(1 1 7)	2.7165	5	9	(1 4 1)	(2 5 2)	2.9140	6	9
(1 1 3)	(1 2 6)	2.7816	5	9	(1 4 1)	(3 4 2)	2.9526	6	9
(1 1 3)	(2 1 6)	2.8409	5	9	(1 4 1)	(2 4 3)	2.9344	6	9
(1 1 3)	(1 3 5)	2.7990	5	9	(4 1 1)	(4 1 1)	2.7953	6	6
(1 1 3)	(1 5 3)	2.8052	5	9	(4 1 1)	(5 1 1)	2.7953	6	7
(1 1 3)	(3 1 5)	2.8771	5	9	(4 1 1)	(4 1 2)	2.8654	6	7
(1 1 3)	(5 1 3)	2.8911	5	9	(4 1 1)	(4 2 1)	2.8646	6	7
(1 1 3)	(1 4 4)	2.8039	5	9	(4 1 1)	(6 1 1)	2.7953	6	8
(1 1 3)	(4 1 4)	2.8879	5	9	(4 1 1)	(5 1 2)	2.8654	6	8
(1 1 3)	(2 2 5)	2.9140	5	9	(4 1 1)	(5 2 1)	2.8646	6	8
(1 1 3)	(3 2 4)	2.9526	5	9	(4 1 1)	(4 1 3)	2.8847	6	8
(1 1 3)	(2 4 3)	2.9404	5	9	(4 1 1)	(4 3 1)	2.8842	6	8
(1 1 3)	(2 3 4)	2.9344	5	9	(4 1 1)	(4 2 2)	2.9392	6	8
(1 1 3)	(4 2 3)	2.9641	5	9	(4 1 1)	(7 1 1)	2.7953	6	9
(1 1 3)	(3 3 3)	2.9740	5	9	(4 1 1)	(6 1 2)	2.8654	6	9
(1 1 4)	(1 1 4)	2.7165	6	6	(4 1 1)	(6 2 1)	2.8646	6	9
(1 1 4)	(1 1 5)	2.7165	6	7	(4 1 1)	(5 1 3)	2.8847	6	9
(1 1 4)	(1 2 4)	2.7816	6	7	(4 1 1)	(5 3 1)	2.8842	6	9

Table C.6: Complete enumeration, $N = 3$, max.$\sum c = 6$, max.$\sum K = 9$

c	K	θ	$\sum c$	$\sum K$	c	K	θ	$\sum c$	$\sum K$
(4 1 1)	(4 1 4)	2.8902	6	9	(2 3 1)	(3 3 2)	2.9687	6	8
(4 1 1)	(4 4 1)	2.8900	6	9	(2 3 1)	(2 3 3)	2.9886	6	8
(4 1 1)	(5 2 2)	2.9392	6	9	(2 3 1)	(2 6 1)	2.8904	6	9
(4 1 1)	(4 3 2)	2.9602	6	9	(2 3 1)	(3 5 1)	2.8922	6	9
(4 1 1)	(4 2 3)	2.9603	6	9	(2 3 1)	(5 3 1)	2.8925	6	9
(1 2 3)	(1 2 3)	2.8049	6	6	(2 3 1)	(4 4 1)	2.8924	6	9
(1 2 3)	(1 2 4)	2.8049	6	7	(2 3 1)	(2 5 2)	2.9669	6	9
(1 2 3)	(1 3 3)	2.8056	6	7	(2 3 1)	(3 4 2)	2.9687	6	9
(1 2 3)	(2 2 3)	2.9417	6	7	(2 3 1)	(2 4 3)	2.9886	6	9
(1 2 3)	(1 3 4)	2.8056	6	8	(2 3 1)	(2 3 4)	2.9950	6	9
(1 2 3)	(1 4 3)	2.8057	6	8	(2 3 1)	(4 3 2)	2.9690	6	9
(1 2 3)	(2 2 4)	2.9417	6	8	(2 3 1)	(3 3 3)	2.9905	6	9
(1 2 3)	(3 2 3)	2.9818	6	8	(2 1 3)	(2 1 3)	2.8904	6	6
(1 2 3)	(2 3 3)	2.9427	6	8	(2 1 3)	(2 1 4)	2.8904	6	7
(1 2 3)	(1 2 6)	2.8049	6	9	(2 1 3)	(3 1 3)	2.8922	6	7
(1 2 3)	(1 3 5)	2.8056	6	9	(2 1 3)	(2 2 3)	2.9669	6	7
(1 2 3)	(1 5 3)	2.8057	6	9	(2 1 3)	(2 1 5)	2.8904	6	8
(1 2 3)	(1 4 4)	2.8057	6	9	(2 1 3)	(3 1 4)	2.8922	6	8
(1 2 3)	(2 2 5)	2.9417	6	9	(2 1 3)	(4 1 3)	2.8924	6	8
(1 2 3)	(3 2 4)	2.9818	6	9	(2 1 3)	(2 2 4)	2.9669	6	8
(1 2 3)	(2 4 3)	2.9428	6	9	(2 1 3)	(3 2 3)	2.9687	6	8
(1 2 3)	(2 3 4)	2.9427	6	9	(2 1 3)	(2 3 3)	2.9886	6	8
(1 2 3)	(4 2 3)	2.9937	6	9	(2 1 3)	(2 1 6)	2.8904	6	9
(1 2 3)	(3 3 3)	2.9828	6	9	(2 1 3)	(3 1 5)	2.8922	6	9
(1 3 2)	(1 3 2)	2.8049	6	6	(2 1 3)	(5 1 3)	2.8925	6	9
(1 3 2)	(1 4 2)	2.8049	6	7	(2 1 3)	(4 1 4)	2.8924	6	9
(1 3 2)	(1 3 3)	2.8056	6	7	(2 1 3)	(2 2 5)	2.9669	6	9
(1 3 2)	(2 3 2)	2.9417	6	7	(2 1 3)	(3 2 4)	2.9687	6	9
(1 3 2)	(1 5 2)	2.8049	6	8	(2 1 3)	(2 4 3)	2.9950	6	9
(1 3 2)	(1 3 4)	2.8057	6	8	(2 1 3)	(2 3 4)	2.9886	6	9
(1 3 2)	(1 4 3)	2.8056	6	8	(2 1 3)	(4 2 3)	2.9690	6	9
(1 3 2)	(2 4 2)	2.9417	6	8	(2 1 3)	(3 3 3)	2.9905	6	9
(1 3 2)	(3 3 2)	2.9818	6	8	(3 1 2)	(3 1 2)	2.8914	6	6
(1 3 2)	(2 3 3)	2.9427	6	8	(3 1 2)	(4 1 2)	2.8914	6	7
(1 3 2)	(1 6 2)	2.8049	6	9	(3 1 2)	(3 1 3)	2.8923	6	7
(1 3 2)	(1 3 5)	2.8057	6	9	(3 1 2)	(3 2 2)	2.9679	6	7
(1 3 2)	(1 5 3)	2.8056	6	9	(3 1 2)	(5 1 2)	2.8914	6	8
(1 3 2)	(1 4 4)	2.8057	6	9	(3 1 2)	(3 1 4)	2.8924	6	8
(1 3 2)	(2 5 2)	2.9417	6	9	(3 1 2)	(4 1 3)	2.8923	6	8
(1 3 2)	(3 4 2)	2.9818	6	9	(3 1 2)	(4 2 2)	2.9679	6	8
(1 3 2)	(2 4 3)	2.9427	6	9	(3 1 2)	(3 2 3)	2.9689	6	8
(1 3 2)	(2 3 4)	2.9428	6	9	(3 1 2)	(3 3 2)	2.9896	6	8
(1 3 2)	(4 3 2)	2.9937	6	9	(3 1 2)	(6 1 2)	2.8914	6	9
(1 3 2)	(3 3 3)	2.9828	6	9	(3 1 2)	(3 1 5)	2.8925	6	9
(2 3 1)	(2 3 1)	2.8904	6	6	(3 1 2)	(5 1 3)	2.8923	6	9
(2 3 1)	(2 4 1)	2.8904	6	7	(3 1 2)	(4 1 4)	2.8924	6	9
(2 3 1)	(3 3 1)	2.8922	6	7	(3 1 2)	(5 2 2)	2.9679	6	9
(2 3 1)	(2 3 2)	2.9669	6	7	(3 1 2)	(3 2 4)	2.9690	6	9
(2 3 1)	(2 5 1)	2.8904	6	8	(3 1 2)	(3 4 2)	2.9960	6	9
(2 3 1)	(3 4 1)	2.8922	6	8	(3 1 2)	(4 3 2)	2.9896	6	9
(2 3 1)	(4 3 1)	2.8924	6	8	(3 1 2)	(4 2 3)	2.9689	6	9
(2 3 1)	(2 4 2)	2.9669	6	8	(3 1 2)	(3 3 3)	2.9906	6	9

Table C.7: Complete enumeration, $N = 3$, max.$\sum c = 6$, max.$\sum K = 9$

c	K	θ	$\sum c$	$\sum K$
(3 2 1)	(3 2 1)	2.8914	6	6
(3 2 1)	(4 2 1)	2.8914	6	7
(3 2 1)	(3 3 1)	2.8923	6	7
(3 2 1)	(3 2 2)	2.9679	6	7
(3 2 1)	(5 2 1)	2.8914	6	8
(3 2 1)	(3 4 1)	2.8924	6	8
(3 2 1)	(4 3 1)	2.8923	6	8
(3 2 1)	(4 2 2)	2.9679	6	8
(3 2 1)	(3 2 3)	2.9896	6	8
(3 2 1)	(3 3 2)	2.9689	6	8
(3 2 1)	(6 2 1)	2.8914	6	9
(3 2 1)	(3 5 1)	2.8925	6	9
(3 2 1)	(5 3 1)	2.8923	6	9
(3 2 1)	(4 4 1)	2.8924	6	9
(3 2 1)	(5 2 2)	2.9679	6	9
(3 2 1)	(3 2 4)	2.9960	6	9
(3 2 1)	(3 4 2)	2.9690	6	9
(3 2 1)	(4 3 2)	2.9689	6	9
(3 2 1)	(4 2 3)	2.9896	6	9
(3 2 1)	(3 3 3)	2.9906	6	9
(2 2 2)	**(2 2 2)**	**2.9954**	**6**	**6**
(2 2 2)	(2 2 3)	2.9964	6	7
(2 2 2)	(2 3 2)	2.9964	6	7
(2 2 2)	**(3 2 2)**	**2.9973**	**6**	**7**
(2 2 2)	(2 2 4)	2.9965	6	8
(2 2 2)	(2 4 2)	2.9965	6	8
(2 2 2)	(4 2 2)	2.9975	6	8
(2 2 2)	**(3 2 3)**	**2.9983**	**6**	**8**
(2 2 2)	(3 3 2)	2.9983	6	8
(2 2 2)	(2 3 3)	2.9974	6	8
(2 2 2)	(2 2 5)	2.9965	6	9
(2 2 2)	(2 5 2)	2.9965	6	9
(2 2 2)	(5 2 2)	2.9976	6	9
(2 2 2)	(3 2 4)	2.9984	6	9
(2 2 2)	(3 4 2)	2.9984	6	9
(2 2 2)	(2 4 3)	2.9975	6	9
(2 2 2)	(2 3 4)	2.9975	6	9
(2 2 2)	(4 3 2)	2.9986	6	9
(2 2 2)	(4 2 3)	2.9986	6	9
(2 2 2)	**(3 3 3)**	**2.9993**	**6**	**9**

Joint Buffer-Server Optimization in General Queueing Networks using the Generalized Expansion Method

R. Andriansyah
Dr.T.van.Woensel
Prof.Dr.ir.J.W.M. Bertrand

technische universiteit eindhoven

RESEARCH QUESTION:

"*Given a specific system configuration and a limited availability of budget in a production/manufacturing environment, how should one decide upon the number of buffers and servers to maximize productivity?"*

MATHEMATICAL FORMULATION:

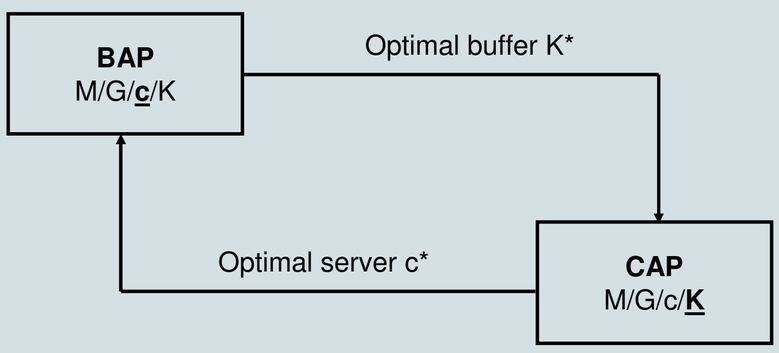
"Given a queueing network structure with N nodes, an external arrival rate of λ, and service rates of μ_i, what is the optimal number of servers (x_i) and buffers (y_i) at each node i *so as to maximize the corresponding throughput rate $\theta(x,y)$?"*

METHODOLOGIES

1st method: BAP-CAP interaction

• Using this method, we interact the two optimization methodologies that are available in the literature, namely the *Buffer Allocation Problem* (BAP) and the *Server Allocation Problem* (CAP).

• The BAP identifies the optimal buffer allocation, given a fixed server configuration. We will use the output from the BAP as the input for the CAP. The CAP then identifies the optimal server allocation, given the fixed buffer configuration from the BAP. We continue to do this until all dominated and *non-dominated* allocation of servers and buffers are obtained.



2nd method: Single objective BCAP

• The second method involves the simultaneous optimization of buffers (y_i) and servers (x_i). The following constrained objective function is used:

$$z = \min\left[\sum x_i + \sum y_i\right]$$ → Simple objective function

s.t.

$$\theta_{(x,y)} \geq \theta^{\tau}$$

$$y_i \geq x_i \qquad i = \{1,2,3,....,N\}$$ → Complicating constraints

• We relaxed the above objective function using *Lagrangian relaxation* to form a *unconstrained objective function* by including the two complicating constraints into the objective function. The unconstrained objective function is as follows:

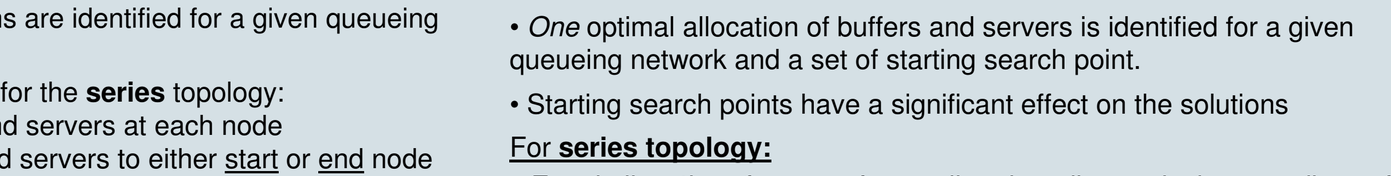
$$z = \min\left[\sum x_i + \sum y_i + \beta(\theta^{\tau} - \theta_{(x,y)}) + \gamma(x_i - y_i)\right]$$

• We also consider the case where the price of servers is not equal to that of buffers. The following objective function is used:

$$z = \min\left[\alpha\sum x_i + (2-\alpha)\sum y_i + \beta(\theta^{\tau} - \theta_{(x,y)}) + \gamma(x_i - y_i)\right]$$

3rd method: complete enumeration

• The complete enumeration is used to compare the performance of the two optimization methodologies for a simple queueing network structure as follows. The maximum number of servers Σc is 6, and the maximum number of buffers ΣK is 9



EXPERIMENTAL SETTINGS

• Three topologies of a queueing network are considered, namely *series, split,* and *merge* topologies. The network structure for each of the topology is depicted in the below figures.

• The arrival rate λ is set to **3, 7,** or **15**. The processing rate μ is set to 10 to all nodes. The squared coefficient of variation (CV) is set to **0.5, 1.0,** or **1.5**

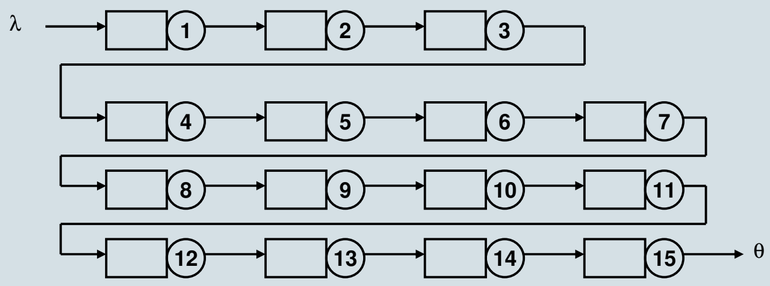


Figure 2: Series topology

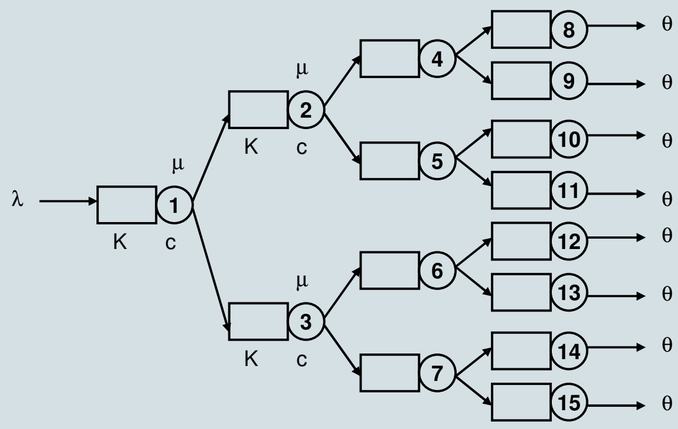


Figure 3: Split topology

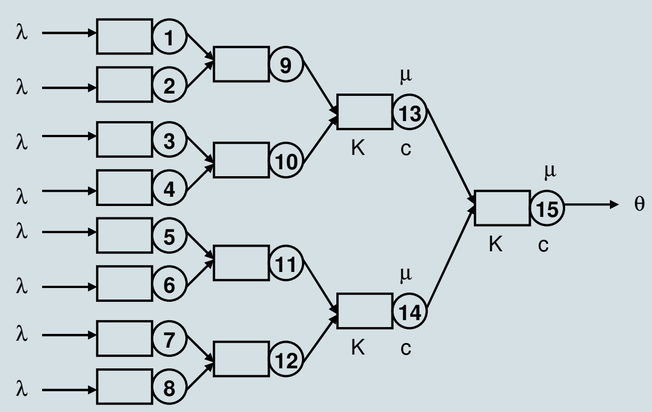


Figure 3: Merge topology

RESULTS

1st methodology: BAP-CAP interaction

• Several non-dominated solutions are identified for a given queueing network structure

• Buffer-server allocation pattern for the **series** topology:
(1) Equal allocation of buffers and servers at each node
(2) Extra allocation of buffers and servers to either start or end node

Table 1: Results for 1st method, series topology

λ	CV	c			K			Σ c	Σ K	Σ B	θ
		1	2	3	1	2	3				
3	1.0	2	2	2	3	3	3	6	9	3	2.9993
		2	2	3	3	3	3	7	9	2	3.0000
7	1.0	2	2	2	6	6	6	6	18	12	7.0000
15	1.0	3	2	2	23	10	10	7	43	36	15.000
		4	2	2	11	8	8	8	27	19	15.000
		3	4	4	7	7	7	11	21	10	14.998
	0.5	2	2	2	9	9	20	6	38	32	15.000
		2	2	3	9	9	9	7	27	20	14.995
		2	2	4	9	9	9	8	27	19	15.000
	1.5	2	2	3	11	11	11	7	33	26	14.997
		2	3	2	11	25	11	7	47	40	15.000

• Buffer-server allocation pattern for the **split** topology: more servers and buffers allocated to the *first* splitting node

• Buffer-server allocation pattern for the **merge** topology: more servers and buffers allocated to the *last* merging node

• Buffer allocation is more extreme than server allocation – additional servers increases processing rate, while additional buffers don't

• *Non-dominated* solutions are not necessarily optimal solutions

2nd methodology: Single objective BCAP

• *One* optimal allocation of buffers and servers is identified for a given queueing network and a set of starting search point.

• Starting search points have a significant effect on the solutions

For **series topology:**

• *Equal allocation* of servers for small and medium arrivals, regardless of CV

• Extra server allocated at either start or end node for high arrivals. This is consistent regardless of the network size

• Buffer allocation is more unpredictable

• Nodes located near or at the start/end of the line tend to receive more buffers. However, this behavior is less consistent across network size.

Table 2: Results for 2nd method, series topology

λ	CV	c							K							Σ c	Σ K	Σ B	θ
3	0.5	2	2	2	2	2	2	2	2	2	2	2	2	3	5	14	18	4	3.0000
	1	2	2	2	2	2	2	2	4	2	2	2	2	2	2	14	16	2	2.9998
	1.5	2	2	2	2	2	2	2	4	2	2	2	2	2	2	14	16	2	2.9997
7	0.5	2	2	2	2	2	2	2	7	2	2	7	2	2	3	14	25	11	6.9988
	1	2	2	2	2	2	2	2	8	2	2	3	2	2	6	14	25	11	6.9992
	1.5	3	3	3	3	3	3	3	3	3	3	3	3	3	3	21	21	0	6.9867
15	0.5	2	2	2	2	2	2	4	2	2	2	2	2	2	5	16	17	1	14.936
	1	3	2	2	2	2	2	2	3	3	2	2	2	2	3	15	17	2	15.000
	1.5	3	2	2	2	2	2	2	3	2	2	2	2	2	3	15	16	1	15.000

• For **split/merge** topology: more servers are allocated to the **splitting/ merging** node; priority is given to the **first splitting/last merging** node (except for 3-node network with low arrival)

3rd methodology: complete enumeration

• One optimal solution was found using the BAP-CAP interaction

• Two optimal solutions were found using the single objective BCAP

BAP-CAP interaction
c=(2 2 2) ; K=(3 3 3)

BCAP
c=(2 2 2) ; K=(3 2 2)
c=(2 2 2) ; K=(2 2 2)

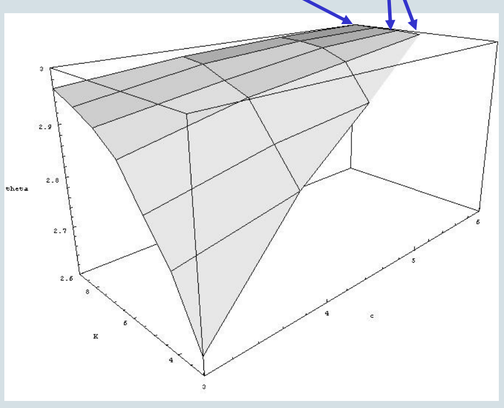


Figure 5: optimal results from complete enumeration

CONCLUSIONS

• The single objective BCAP is more preferable than the BAP-CAP interaction
• BAP-CAP interaction generates several *non-dominated* solutions.
• Single objective BCAP generates *one* solution for a given queueing network structure

CONTRIBUTIONS

• Joint buffer-server optimization in *M/G/c/K* queueing networks
• Arbitrary networks with 3 topologies: series, merge, and split
• Two optimization methodologies: BAP-CAP interaction, and single objective BCAP
• Comparison with complete enumeration
• Vast amount of data: 533 experiments from the optimization, 622 settings from the complete enumeration

/faculteit technologie management