Evaluation of optimization methods for the buffer allocation problem

Feddahi, J.

Award date:
2007

Evaluation of optimization methods for the Buffer Allocation Problem

Jamel Feddahi
0501233
Final version

Abstract

We have investigated whether a Decomposition and Column Generating (DCG) algorithm can offer benefits for the optimization of the Buffer Allocation Problem (BAP). In addition to this, a subgradient method has been applied in order to optimize the cost factor (or Lagrange multiplier) playing a key role in Lagrangian relaxation. Lagrangian relaxation has been applied to the BAP before, but only with a fixed cost factor. By applying this method we have been able to find tighter lower bounds for optimal performance than with this latter conventional method.

Summary

The Buffer Allocation Problem

This report is the result of a Master's project that has been carried out into optimization methods for the Buffer Allocation Problem (BAP). The Buffer Allocation Problem is concerned with the allocation of buffers to nodes in a finite queuing network. The trade-off underlying this problem is that costly buffers have to be minimized while at the same time throughput has to be maximized. This problem can thus be characterized as multi-objective. The objective function of the BAP is non-linear. Also, throughput depends on a number of network parameters and due to the complex way in which these are related to each other they have not yet been captured explicitly in a model.

Project assignment

The assignment carried out during this project has been formulated as follows:

Investigate the Decomposition and Column Generating (DCG) approach for its applicability to solve the Buffer Allocation Problem and (possibly) apply this method in order to evaluate its performance.
In addition to this, we would like to investigate the accuracy of Lagrangian relaxation as an optimization method for the BAP and improve the tightness of the lower bound on optimal performance that is generated by this method.

Applicability of the DCG for optimizing the BAP

A DCG approach has been applied recently for the optimization of spare parts inventory control problems under availability constraints. This DCG is very similar to Dantzig-Wolfe decomposition, which has been used to solve large-scale linear problems.
The nature and formulation of the problems to which the DCG has been applied shows some similarities to our BAP, e.g. the multi-objective and non-linear objective function. For the aforementioned (spare parts inventory control) problems the DCG manages to decompose the original multi-item problem into multiple single-item problems, making its optimization much more efficient.
Our BAP, however, differs in that it does not have a constraint per node and all jobs in the network have the same (relevant) characteristics. Therefore we cannot benefit from the same decomposition using the DCG. This decomposition from one multi-item problem into multiple one-item problems is the main benefit that the DCG has provided in solving a multi-objective, non-linear problem efficiently. We thus conclude that the differences between both types of models do not justify its application to the BAP.

Evaluation of Lagrangian relaxation applied to the BAP

Instead of using a fixed Lagrange multiplier we have applied a subgradient method for finding the Lagrange multiplier that results in the tightest bound on optimal performance of the BAP. This subgradient optimization method is an iterative procedure that has been effective in producing good multiplier values in a variety of Lagrangian-based optimization problems.

Enumeration has been used to find the buffer allocation for series networks. These have been compared to the buffer allocation resulting from our optimized Lagrange multiplier and the buffer allocation when a fixed Lagrange multiplier is chosen.

Results and conclusions

Using the subgradient method for setting the Lagrange multiplier we have gained some promising results on deriving a tight lower bound on optimal performance for the BAP. We have used the relative gap between this bound and the optimal value of the objective function for the BAP. For one setting we have seen a reduction of 78% in this gap due to the use of the subgradient method instead of using a fixed Lagrange multiplier of 1000. More testing and validation of our implementation of the subgradient method is required before a firmer conclusion can be drawn on the potential performance improvement.

Table of Contents

Preface

This thesis is the final project that I have carried out for my Master in Industrial Engineering and Management Science at the Eindhoven University of Technology. The past few months have allowed me to put the knowledge and energy that I have gathered during my studies into practice on some challenging research questions.

First of all, I would like to thank Tom van Woensel and Kai Huang for the opportunity to carry out this project and for their guidance and input to our discussions. They have always been very flexible and willing to find the time for our meetings and discussions.

I wouldn't have been able to carry out this project without the support from my wife, parents, and friends. They have supported me and helped me go on as difficult questions or hurdles arose along the project.

Jamel Feddahi

1. Introduction

This chapter contains a description and positioning of the Master's project. In addition to this, a research model describing the main stages of this project will be presented.

1.1 Project positioning

This Master's project has been carried out within the Operations Planning, Accounting and Control (OPAC) research group of the Technology Management faculty at the Eindhoven University of Technology. The objective of the OPAC research group is to develop scientific knowledge with respect to the design and control of operational processes in service and manufacturing industries, and to be acknowledged as leading in the area of Operations Management by the scientific community and the industrial community (research group website, http://www.tm.tue.nl/en/subdepartments/opac).

The project has been carried out under the supervision of dr. Tom van Woensel who has been doing research on the topic of this Master's project for some time now. Dr. Kai Huang has acted as the secondary supervisor.

1.2 Initial assignment formulation

In this Master's project we focus on evaluating the applicability of a relatively new optimization method for the Buffer Allocation Problem (BAP). In addition to this, Lagrangian relaxation, an optimization method which has already been applied to the BAP, will be evaluated for its performance. The BAP is concerned with the allocation of buffers in finite queueing networks in which blocking can influence network performance. More specifically we would like to:

Investigate the Decomposition and Column Generating (DCG) approach for its applicability to solve the Buffer Allocation Problem and (possibly) apply this method in order to evaluate its performance.
In addition to this, we would like to investigate the accuracy of Lagrangian relaxation as an optimization method for the BAP and improve the tightness of the lower bound on optimal performance that is generated by this method.

This project builds on work that has been done recently in this field and we aim at comparing the DCG approach with optimization methods applied therein. The BAP and recent research into its optimization will be presented in the next chapter.

1.3 Research model

We will use the research model of Verschuren and Dorewaard (2000) to describe the

structure of this project. Their model schematically depicts the research object (optimization approaches for the BAP), the research methods, confrontation between theory and practice (not within the scope of this project) and the result of the research project. Figure 1.1 on the next page shows the building blocks of this research project.

The first part of the assignment that we have formulated in the previous section will be carried out by studying literature on the Buffer Allocation Problem and describing optimization methods that have been used to optimize the BAP. Next, we will describe the characteristics of problems to which the DCG has already been applied and compare these problems to the BAP in order to determine whether applying it to solve the BAP is useful.

In the last part of this project we will apply literature on the subgradient method and combine this with literature on Lagrangian optimization in order to optimize the Lagrangian multiplier (which can be considered as a cost factor for deviating from the threshold throughput in a network). By deriving the optimal solution via enumeration we will try to asses the accuracy of the bounds generated using the subgradient method.

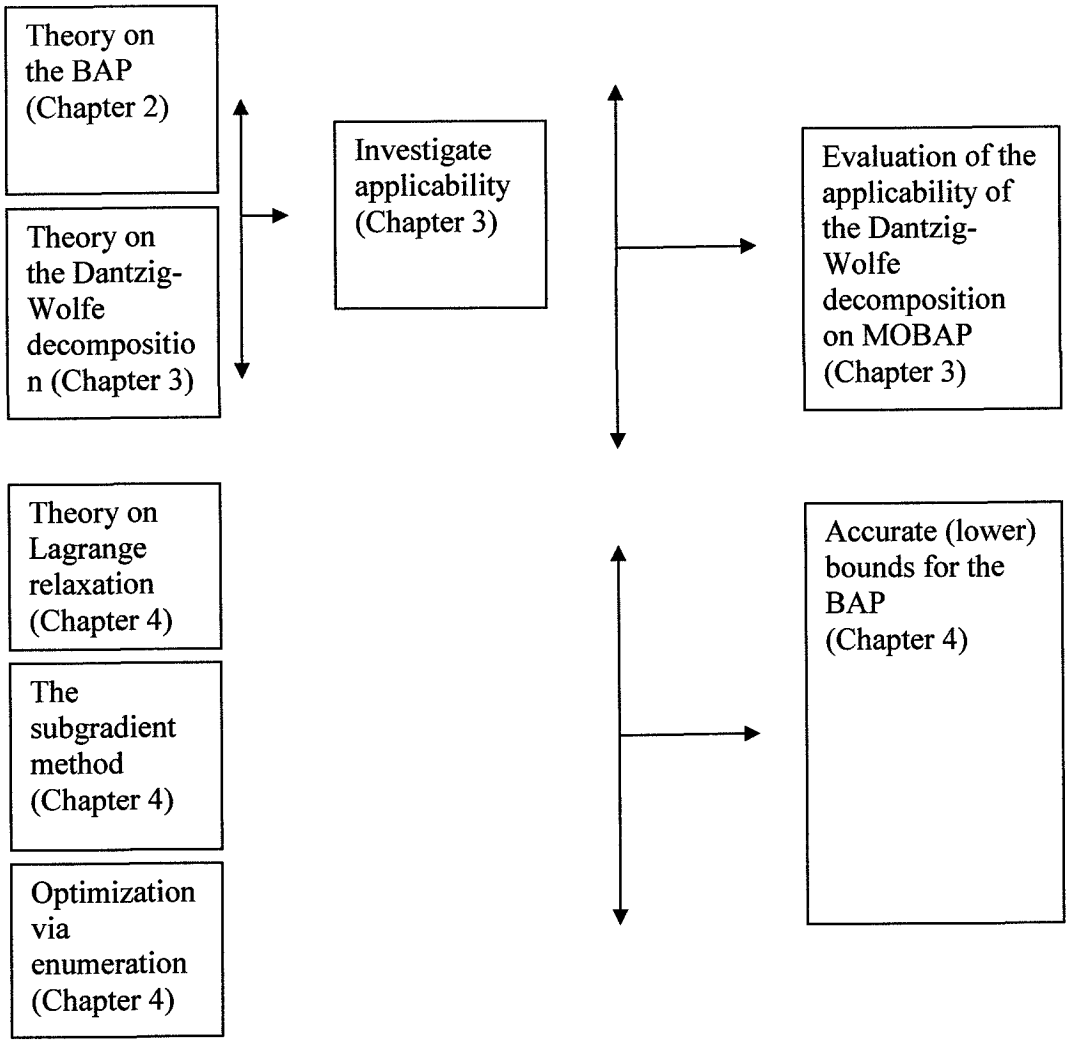

Figure 1.1: research model for the Master thesis

2. The Buffer Allocation Problem

This chapter contains a description of the Buffer Allocation Problem, and approaches used for generating optimal solutions and performance evaluation (Generalized Expansion Method).

2.1 Relevance

The Buffer Allocation Problem has been studied extensively in scientific research (see section 2.2 for an overview of publications). The main focus in these articles is on finding the allocation of buffers in finite queuing networks that optimizes some performance measure, usually the minimization of costs associated with buffers (subject to a realizing a minimum throughput).
Buffers play an important role in many processes and applications. By decoupling multiple stages in a process (or nodes in a network) via buffers each stage (or node) can operate without directly being affected by two well-known disruptions: starvation and blocking (Macgregor Smith and Chikhale, 1995). Starvation is the result of one stage still handling a job while the next (idle) stage is ready to process this job. Blocking occurs when jobs that have been processed at one stage cannot proceed to the next stage because stations or servers at the next stage are being occupied by another job. Both types of disruptions result in the network being unable to realize the throughput that it would have been had it been decoupled by one or more buffers.
Increasing the number of buffers can result in an increase in the number of jobs that can be handled in the network as stages can continue working without directly being affected by the aforementioned disruptions. There is a limitation, however, to how much throughput can increase as the number of buffers increases. Adding more buffers above a certain threshold does not lead to a higher throughput. Also, as more buffers are added the increase in throughput normally decreases; there is thus no linear relationship between the number of buffers and the throughput in a queuing network.

Incorporating buffers in, for example, a production line, or communications network, however, does not go without costs. These costs can be considerable and have to be traded off against the additional increase in throughput that can be realized with more buffers. Macgregor Smith and Chikhale (1995) argue that the costs of buffers consist of:

- indirect costs that are proportional to the time a customer spends in the queuing system, and
- costs due to the occupation of space

2.2 Literature review

In the study of finite queueing networks in which blocking can occur we can distinguish two directions of research. In one of these researchers are mainly concerned with the optimization of the Buffer Allocation Problem. Another branch of research that can be distinguished tries to better understand and model the relationships between the relevant parameters in a finite queueing network. These two branches are closely related and lately we can see that more and more these are being combined (see e.g. Smith et al., 2006).
Our main focus in this project is on the application and evaluation of optimization approaches and we will restrict to giving an overview of articles that deal with this.

Optimization methods used to solve the BAP can be divided in four categories (Cruz et al., 2007):
- Simulation
- Metaheuristics
- Dynamic programming
- Search methods

Figure 2.1 links some of the numerous articles that have appeared in this area of research to these optimization methods.

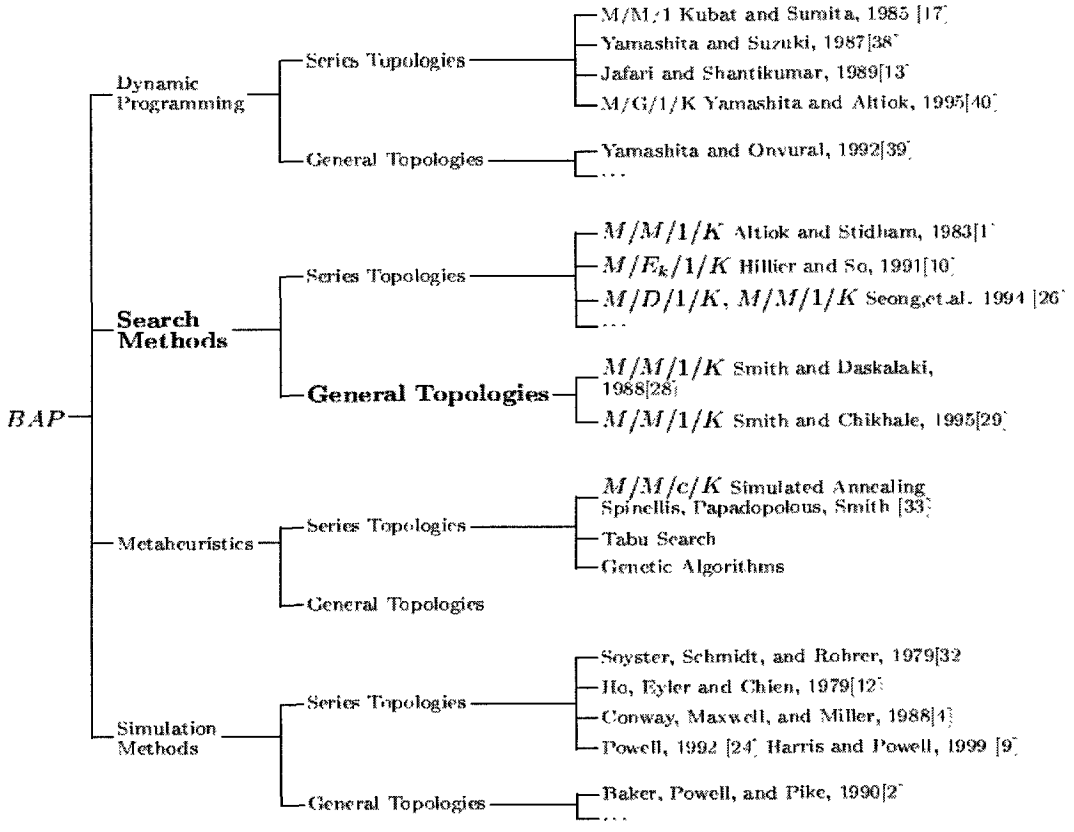

Figure 2.1: diagram of optimization approaches applied to the BAP

Next, we will give a short description of each type of method.

Simulation methods make use of robust assumptions to represent a real system. This is done by using general probability distributions to model the various aspects of the system, such as interarrival times, batch size of the arrivals, and service times. Simulation methods are usually very general and efficient, but they require great computational effort that may reduce the size of treatable instances. Simulation has been applied by a number of researchers, among who we can mention Soyster et al. (1979) for series queuing networks, and Baker et al. (1990) for general topologies.
Metaheuristics are very popular methods nowadays, mainly because of the increasing computational capacity available. Typical techniques that fall into this area include simulated annealing, tabu search, and more recently, generic algorithms. The advantages of metaheuristics are the absence of restrictive assumptions usually required by the traditional methods and the ability of avoiding local optima traps in the search for the global optimum. The disadvantage is that usually the metaheuristics must be tailored to the special structure of the problem. Spinellis et al. (2000), e.g., have reported successful results for optimization with simulated annealing.
Dynamic programming is another powerful and reasonable approach for the BAP. Usually the exponential space complexity of dynamic programming methods reduces their applicability to instances of a very small size. Dynamic programming has been applied successfully to the BAP by e.g. Kubat and Sumita (1985) for M/M/1 queueing networks and by Yamashita and Onvural (1998) for general topologies.
Finally, there are the *search methods*, which try to solve the problem avoiding the combinatorial explosion of possible solutions by choosing those solutions that are close to the optimum results. Their main disadvantage is their restrictive assumptions, such as the concavity and convexity that may limit the applicability. Three closely related search methods (two for a single-objective formulation of the BAP and one for a multi-objective formulation of the BAP) will be described in more detail in section 2.4.

2.3 Model formulation

In essence the Buffer Allocation Problem is about trading off the costs of extra buffers against the revenues as a result of (a possible) increase in throughput. As mentioned in section 2.1 the relationship between the number of buffers and the throughput in a network is non-linear. Another characteristic of the BAP is that the throughput depends on a number of factors (service times, interarrival times, etc), making it hard to model the relationships between the relevant variables explicitly. With this description of the BAP we can describe this problem from a modeling point of view as a multi-objective, non-linear, integer optimization problem.

Previous research has focused on a single-object formulation, where the number of buffers is minimized subject to realizing some minimum throughput constraint. This single-objective formulation can be formalized as follows:

$$\min Z = \sum_i x_i \quad (1)$$

s.t.

$$\Theta(x) \geq \Theta^{\min} \quad (2)$$

$$x_i \in \{1,2,3,...\}, \forall i \quad (3)$$

Where the symbols used will be defined as:
x_i := buffer i in the queueing network
Z := the total number of buffers in the queuing network
$\Theta(x)$:= the throughput in the network when the total number of buffers in the network equals x
$\Theta^{\min}$:= the threshold throughput which should be minimally achieved in the network

Cruz et al (2007) are the first to have investigated the simultaneous trade-off between the number of buffers and throughput by solving the multi-objective Buffer Allocation Problem (MOBAP). The formulation for the MOBAP is a follows:

$$Optimize\, Z = \{f_1(x), f_2(x)\} \quad (4)$$

s.t.

$$x_i \in \{1,2,3,...\}, \forall i \quad (5)$$

In which

$$f_1(x) = \sum_i x_i \quad (6)$$

$$f_2(x) = \Theta(x) \quad (7)$$

In this report we will focus on the single-objective formulation.

2.4 Relevant optimization approaches

In this section we will describe the search algorithms applied in MacGregor Smith et al. (2006), Cruz et al. (2007) and in Cruz et al. (2007) in some more detail as they are closely related to the assignments that we would like to carry out in this project.
The first two publications focus on the single-objective BAP and describe algorithms that are capable of generating accurate solutions efficiently. Both algorithms make use of Lagrangian relaxation in order to solve the BAP and a derivative-free search algorithm to generate the optimal vector of buffers in a network with general service times. They

differ, however, in the structure of their search algorithm, where the latter has proven to be more efficient.
Cruz et al. (2007) deal with an application of a Genetic Algorithm for optimizing the Multi-Objective Buffer Allocation Problem. They provide a Genetic Algorithm that is capable of generating the set of pareto-optimal buffer allocations.

Lagrangian relaxation
Lagrangian relaxation is a technique that has been applied frequently to the BAP (see e.g. MacGregor Smith et al. (2006), Cruz et al. (2007)) . It consists in relaxing the complicating constraint and including it in the objective function as a penalty factor. In the case of the BAP, for the problem consisting of (1), (2), and (3), which we will repeat first:

$$\min Z = \sum_i x_i \tag{1}$$

s.t.

$$\Theta(x) \geq \Theta^{\min} \tag{2}$$

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

a dual variable α is defined and (1) and (2) are replaced by the Lagrangian:

$$L(\alpha) = \min\left[\sum_i x_i + \alpha\left(\Theta^{\min} - \Theta(x)\right)\right]$$

By first setting the threshold throughput exactly to the total external arrival rate, and setting an acceptable difference between $\Theta(x)$ and Θ^r the problem is solved.

Search algorithm in MacGregor Smith (2006): Powell's algorithm
Assuming poisson arrival processes and blocking after service, Smith et al. (2006) have used Powel's method to search for the optimal buffer vector(s) in general service, finite queuing networks. Powel's method locates the minimum of f(x) of a non-linear function by successive unidimensional searches from an initial starting point x(o) along a set of conjugate directions. Powel's method is based on the idea that if a minimum of a non-linear function f(x) is found along p conjugate directions in a stage of the search, and an appropriate step is made in each direction, the overall step from the beginning to the p^{th} step is conjugate to all of the p subdirections of the search.

Search algorithm in Cruz et al. (2007)
This search algorithm is a classical derivative-free direct search method: it starts by reading all relevant variables of the queuing network, and proceeds to optimize $L(\alpha)$ only in relation to the first coordinate of vector x, while the remaining coordinates are kept fixed. The process is repeated for the second coordinate and so on, until the last coordinate is reached. A completely new vector is obtained and compared with the

previous vector. This process is continued until a convergence is reached or the difference between two consecutive vectors is less than a pre-prespecified value.

The genetic algorithm (MOBAP)
Cruz et al. (2007) have used a genetic algorithm to solve the multi-objective BAP. GA's have proven to be very efficient in solving multi-objective problems in general. The GA's are optimization algorithms that perform an approximate global search relying on the information obtained from the evaluation of several points in the search space and obtaining a population of these points that converges to the optimum through the application of the genetic operators mutation, crossover, selection, and elitism (Takahashi et al., 2003). Each one of these operators may be implemented in several different ways, each on of them characterizing an instance of the GA.

In the special case of the multi-objective optimization problems, the operators selection and elitism must be specially structured to correctly identify the individuals. Operators mutation and crossover are independent on the multi-objective nature of the problem (Takahashi et al., 2004). The multi-objective GA evolves the whole population toward the Pareto set, instead of a single point and in a single run the whole Pareto set, or a large portion of it, will be found. This explains the superiority of the GA's over the deterministic algorithms, which are able to find only one Pareto point for each run.

2.5 Performance evaluation: Generalized Expansion Method

Due to the complex relationship between the throughput, and the parameters of a general queuing network (arrival rate, number of buffers, service rates, etc.) there is no general closed-form method for computing this performance measure. Several composition and decomposition methods for estimating the throughput have been proposed, of which Kerbache and Smith (1986) provide a short overview. These researchers have developed an expansion method that can be characterized as a decomposition method in which every node of a network is evaluated and at the end all feedback loops are eliminated. This Generalized Expansion Method (GEM) has proven to be an accurate and efficient method (Kerbache and Smith,1986). Their expansion method was initially developed based on the assumption that interarrival and service times in the network have an exponential distribution. In a later article this assumption has been relaxed and their expansion method extended to incorporate general service times. The only constraint that the GEM poses on the service time distributions is that they have to be renewal processes.

Due to its efficacy the GEM is frequently used for evaluating the performance of a queuing network. Within the GEM we can distinguish three stages:

Stage 1: Network reconfiguration
Stage 2: Parameter estimation
Stage 3: Feedback elimination

In the first stage, the network is expanded by adding an artificial queue node after every finite node (except the final node). This is shown in figure 2.2 for a series network and in figure 2.3 for a network with splitting.

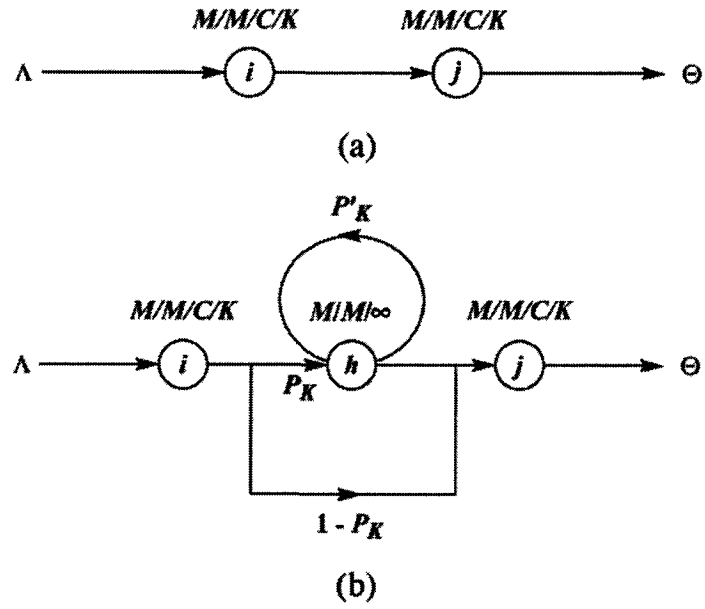

Figure 2.2: expansion with the GEM in a series network

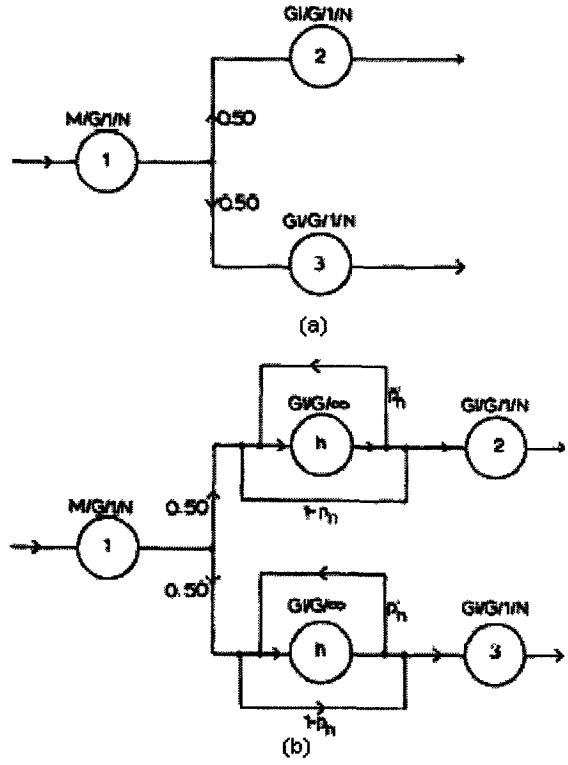

Figure 2.3: expansion with the GEM in a network with splitting

This artificial node is modeled as a M/G/∞ queue in order to capture the waiting time of customers or jobs that are blocked in the network. If the buffer before a subsequent node is full, a job is rerouted to the artificial node and held there until a new place becomes available in the buffer; thus blocking of the job has occurred. After each service period at the artificial node (there is no waiting time due to the infinite number of servers) the jobs or customers proceed to the next node or when the buffers at that node are still occupied are sent back to the artificial node. This creates a feedback loop at the artificial node.

In the second stage, a system of non-linear equations is set up containing the unknown parameters that arise as a consequence of the network expansion. These parameters are:

- the blocking probability of the finite queue of size K
- the feedback blocking probability in the Generalized Expansion method, i.e. the probability that a job finds the next queue occupied after having just been processed in the artificial node
- the remaining service time distribution of a job at a finite queue. This is equal to the time that a blocked job remains in service at an artificial job

In the third stage, the feedback loops at all the artificial nodes are eliminated by recalculating a few parameters (the effect of the feedback loop is incorporated in the parameters of the originating "real" node). With these stages completed one solves a nonlinear system and calculates the desired performance measures of the open finite queuing network under study.

2.6 Research questions

Recent publications inventory management for spare parts have shown some promising techniques that might be applicable to the Buffer Allocation Problem: a decomposition and column generating (DCG) approach and use of a subgradient method for finding tight lower bounds to optimal performance.
In this project we will investigate and (potentially) apply these methods to the BAP. This assignment was formulated in chapter 1 and we will translate it into the following three research questions that we would like to find answers to in this project:

1. Is it possible to use a decomposition and column generating (DCG) approach to solve the nonlinear multi-objective Buffer Allocation Problem and if so, how does this improve the performance of the optimization procedure?

2. How can we determine the Lagrange multiplier for the Lagrange relaxation that yields a tight lower bound for optimal performance of the BAP?

3. How close is this bound to the buffer allocation that results in to optimal performance for the BAP?

3. Decomposition and column generating approach

In this chapter we will discuss an approach similar to DW-decomposition that has been applied to non-linear problems: decomposition and column generating (DCG) approach. We will discuss whether the DCG approach is also suitable for optimizing the Buffer Allocation Problem.

3.1 Introduction

The DCG approach has been developed and applied in e.g. Kranenburg (2006), and Kranenburg and Van Houtum(2007). This approach is very similar to the Dantzig-Wolfe (DW) decomposition algorithm which can be used to solve large-scale linear programming (LP) problems (Dantzig and Wolfe, 1960).
In Kranenburg and Van Houtum (2007) the DCG has been applied to a class of problems which is fairly similar to ours. They investigate the optimization of inventory control policies for (multi-echelon, multi-item) spare parts inventory systems. An important characteristic of their work, which the BAP shares, is that the objective function and (aggregate) constraints for their model are non-linear and not available in closed-form expressions. We will investigate whether this model is close enough to ours to justify the application of the DCG approach for the BAP. The basic concepts and procedures involved in the DW-algorithm have been described in Appendix A. Below we will start with a concise description of the DCG and then discuss the two different models in order to assess its applicability in this project.

3.2 The decomposition and column generating approach

Similar to DW, the DCG consists of transforming an original problem into a so-called Master Problem and solving a subproblem – called a Restricted Master Problem (RMP) in Kranenburg and Van Houtum (2007) - via column generation. This RMP contains only a subset of all columns in the Master Problem and is thus much easier to solve using the revised simplex method. We will illustrate these concepts using the model in Kranenburg and Van Houtum (2007).

The objective function of the Master Problem contains all possible values for variables of the original problem and for each value a binary variable indicating whether that value is chosen or not. (Further on, the integrality constraint is relaxed, which means that the "binary" variable can assume fractional values and thus randomized solutions are allowed.) In Kranenburg and Van Houtum (2007) the decision variables are base-stock policies for multiple items. The objective is to minimize total relevant costs (inventory holding and transportation costs for regular and emergency shipments), subject to the constraint that the average waiting time of a group of items does not exceed the specified threshold waiting time for that group.

The RMP for the Master Problem in Kranenburg (2006) contains for each item a small subset of all possible base-stock policies (initially only one policy per item; an easy rule is provided for selecting this initial policy such that it provides a feasible solution to the RMP, but for the sake of clarity we will omit this here). Then for each item a so-called column generating subproblem is solved. This generates a policy with the lowest reduced cost coefficient. The lowest reduced cost coefficient can be interpreted as the degree of violation from the corresponding constraint in the Master Problem. As long as the column generating subproblem finds policies for an item with a negative reduced cost coefficient these columns are added to the RMP and the RMP is solved (this is similar to the working of the simplex method for solving linear problems, which determines which column to choose as the entering column by finding the variable in the objective function with the most negative coefficient).

3.3 Applicability of the DCG to the BAP

As was mentioned in section 3.1 the problems in Kranenburg (2006), and Kranenburg and Van Houtum (2007) show some similarity with the BAP that we are investigating. Both have a non-linear objective function and constraint(s). Both are of a multi-objective nature and dealing with this has only been possible with heuristics. This has motivated finding an answer to whether the novel DCG might offer benefits in solving the BAP.

Although there are many similarities, there are also some very important differences. The DCG allows the problems in e.g. Kranenburg (2006), and Kranenburg and Van Houtum (2007) to be solved efficiently. This is realized by the decomposition of the original multi-item problem into single-item problems. In our problem, however this is not possible. There is a (throughput) constraint for the overall network consisting of a number of nodes in a certain topology. Furthermore, the jobs in a node all have the same characteristics (e.g. service time, arrival time) so in essence we are already dealing with a "single-item" problem as we can compare the level of a node to the level of a group (of different items) in Kranenburg and Van Houtum(2007) or a (stocking)location in Kranenburg (2006).
A more similar problem to the ones in Kranenburg (2006), and Kranenburg and Van Houtum (2007) might be obtained if we would have a (throughput) constraint for each buffer and jobs would have different characteristics. That, however, would lead to a completely different problem than the one we are addressing and that does not lie within the scope of this project. We thus end this investigation with the conclusion that the differences between the problems under study are too large to continue with the implementation of the DCG for optimizing the BAP. We do not expect that this will result in major benefits or new insights at the moment. This conclusion has been verified by experts that have experience with the DCG.

We therefore choose to focus on the derivation of good and reliable (lower and upper) bounds for this problem and on calculating optimal performance for the Buffer Allocation Problem (in relatively small queueing networks). This allows us to say something about the performance of methods that have already been used, but that have been subject to the

arbitrary selection of one key parameter of Lagrange relaxation: the value of the Lagrange multiplier. These two subjects will be the subject of the next chapter.

4. Optimal performance and a lower bound for the Buffer Allocation Problem

In this chapter we will focus on deriving tight lower bounds for the performance of the (single-objective) BAP.

As was described in section 2.3.2 recent work has primarily made use of Lagrange relaxation in order to deal with the complexity of the multi-objective nature of the BAP. Optimization has been achieved by relaxing the throughput constraint and including this as an additional term in the objective function (now called the relaxed objective function). Also, a cost variable is associated with this term such that deviating from the threshold throughput is penalized. The penalty variable is also called a Lagrange multiplier. This transformation results in the value of the relaxed objective function always being lower than the value in the original problem for the same buffer allocation. The solution to the relaxed problem thus forms a lower bound for the optimal performance of the BAP.
Being a part of the objective function, the Lagrange multiplier also influences the buffer allocation; with a very high penalty factor a buffer allocation resulting in a throughput very close to the threshold throughput will be favored.
In recent work (e.g. Smith et al, 2006) setting the Lagrange multiplier is driven by having an accurate outcome. This will be described in more detail in section 4.1. Section 4.2 will discuss a commonly used method for finding optimal Lagrange multiplier(s) resulting in the tightest lower bound for a multi-objective problem. This method has been applied in e.g. Wong et al (2006). In section 4.3 we will look at how we can determine the optimal performance of the BAP via enumeration in order to asses how accurate our bound is. Section 4.4 will present some (preliminary) results and conclusions.

4.1 Lower bound for optimal performance

4.1.1 Lagrange multiplier for the BAP

As was mentioned in section 2.4 Lagrange relaxation is used for the optimization of the BAP in recent work on which we are building (e.g. Smith et al., 2006 and Cruz et al., 2007). In this work the Lagrange multiplier is selected such that the single objective BAP is solved. In order to clarify this, we will repeat the original single objective BAP formulation, (1)-(3), and the relaxed objective function after Lagrange relaxation, (4):

$$\min Z = \sum_i x_i \qquad (1)$$

s.t.

$$\Theta(x) \geq \Theta^{\min} \qquad (2)$$

$$x_i \in \{1,2,3,...\}, \forall i \quad (3)$$

In order to apply Lagrange relaxation a dual variable α is defined and (1) and (2) are replaced by the Lagrangian:

$$L(\alpha) = \min\left[\sum_i x_i + \alpha * \left(\Theta^{\min} - \Theta(x)\right)\right] \quad (4)$$

To complete the relaxed formulation of the original problem we add the constraint:

$$\alpha \geq 0 \quad (5)$$

The best possible bound for the single objective formulation is the one where the threshold throughput is achieved (or, due to the integrality constraint, just exceeded), i.e. there are no more buffers than required to meet this constraint. This can be achieved by setting the Lagrange multiplier to infinity; the large (negative) penalty factor and the difference between the throughput and the minimum threshold throughput becoming negative results in a positive term being added to the Lagrangian $L(\alpha)$ when more buffers are added than necessary. Adding more buffers than needed to realize the threshold throughput is thus avoided with an infinite Lagrange multiplier.

This, however, is a non-practical value for the Lagrange Multiplier as minimizing the Lagrangian in equation (4) with an infinite penalty variable requires the number of buffers to be infinite also. This is solved in the abovementioned work of Smith et al., (2006) and Cruz et al. (2007) by allowing a minimal difference between the threshold throughput and the resulting throughput from the optimal buffer allocation. This difference then determines the size of the Lagrange multiplier.

The choice for the difference between the threshold throughput and the resulting throughput allowed is determined very arbitrarily. This way of setting the Lagrange multiplier ignores the influence that the Lagrange multiplier has on the objective function and thus on the tightness of the lower bound for optimal performance. This will be the topic of the remainder of this chapter.

4.2 Optimization of the Lagrange multiplier

In Wong et al. (2006) Lagrange relaxation is used to optimize, a multi-item continuous review model of a two-location inventory systems for repairable spare parts. For given Lagrange multipliers they give a procedure for optimizing the relaxed problem formulation. Their model is very similar to the BAP in that the objective function is also non-linear. The constraints in their model are also non-linear and the decision variables have integer values.

They suggest a method which is commonly used for finding Lagrange multipliers that result in solutions for the relaxed problem that are close to optimal results for the original problem: a subgradient method. This subgradient optimization method is an iterative procedure that has been effective in producing good multiplier values in a variety of Lagrangian-based optimization problems (Fisher, 1985).

In the next section we will describe the subgradient method in more detail.

4.2.1 Subgradient method

As we mentioned in the previous section, the solution to the relaxed problem,

$$L(\alpha)=\min\left[\sum_i x_i+\alpha\left(\Theta^{\min}-\Theta(x)\right)\right] \tag{8}$$

forms an lower bound for our original minimization problem,

$$\min Z=\sum_i x_i \tag{1}$$

s.t.

$$\Theta(x)\geq\Theta^{\min} \tag{2}$$

$$x_i\in\{1,2,3,\ldots\},\forall i \tag{3}$$

This is readily observable when we look at the objective function of the dual problem. For a feasible solution to our original problem, i.e. $\Theta(x)\geq\Theta^{\min}$ the term that is added to the original objective function will always be smaller than or equal to zero.

The problem that we aim to solve with the subgradient method in order to find the tightest lower bound to the optimal solution can be formulated as:

$$Z_D(\alpha)=\underset{\alpha\geq 0}{\text{Max}}\ L(\alpha)=\min\left[\sum_i x_i+\alpha\left(\Theta^{\min}-\Theta(x)\right)\right]$$

For each set of values for x_i we get a linear function in α and we can thus construct a family of equations for all feasible buffer allocations. Minimizing the relaxed objective function means that for a particular value of α, $Z_D(\alpha)$ is equal to the smallest of these functions. The linear equations that satisfy this criterion form a piecewise linear function. We will construct this graph for the BAP using the settings and results in table 4.1 below. The results have been obtained by running a tool that evaluates the throughput of a finite queuing network given a buffer allocation.

Topology: 3 nodes in series External arrival rate: 2.0 Service rate for each node: 10.0 Threshold throughput: 1.8			
Total number of buffers x_i	Throughput $\Theta(x)$	Slope $\Theta^{min} - \Theta(x)$	Objective function $Z_D(\alpha)$
0	1.64678	0.1532	$0.1532*\alpha$
3	1.96407	-0.1641	$3 - 0.1641*\alpha$
6	1.99639	-0.1964	$6 - 0.1964*\alpha$
9	1.99964	-0.1996	$9 - 0.1996*\alpha$
12	1.99996	-0.2000	$12 - 0.2000*\alpha$

Table 4.1 data for piecewise linear graph of $Z_D(\alpha)$

In figure 4.1 we have plotted the family of linear equations of $Z_D(\alpha)$ when the total number of buffers is fixed. When trying to find the tightest lower bound, we are in fact looking for the maximum value of the lower envelop in figure 4.1 (Fisher, 1985).

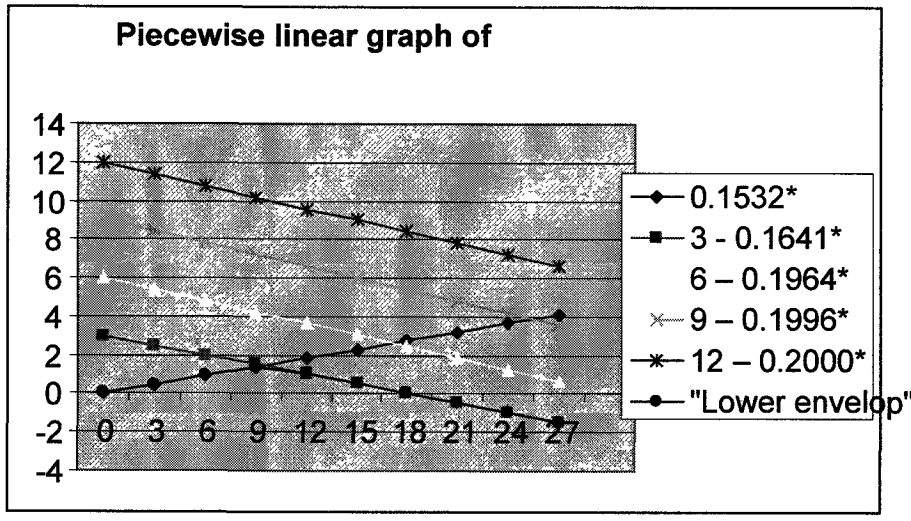

Figure 4.1 Piecewise linear graph of $Z_D(\alpha)$ (horizontal axis is α, vertical axis is $Z_D(\alpha)$)

$Z_D(\alpha)$ is convex and differentiable except at points where the Lagrangian problem has multiple solutions. The subgradient method applies a gradient method to the minimization of $L(\alpha)$ except at those point where $L(\alpha)$ is non-differentiable. At these points it chooses from the set of alternative optimal Lagrangian solutions and uses the coefficient next to the Lagrange multiplier in the Lagrangian as though it were the gradient of $L(\alpha)$. In our case this is $\left(\Theta^{min} - \Theta(x)\right)$.

The steps that the subgradient method goes through are as follows. Starting with an initial Lagrange multiplier α^0 (for which normally and also in our case the value 0 is chosen) a sequence of Lagrange multipliers is determined with the following formula:

$$\alpha^{k+1} = \max(0, \alpha^k - t_k(\Theta^{\min} - \Theta(x^k))),$$

(4)

Where t_k is a scalar stepsize and x^k is an optimal solution to $L(\alpha)$ with dual variables set to α^k.

Whether the optimal solution is actually reached with the subgradient method also depends on the stepsize faction t_k; steps of equal size can result in oscillating behavior (the procedure bounces between two solutions and thus does not converge to the optimal solution), and too large steps can result in the procedure converging to a solution that is not optimal (Fisher, 1985). A formula for t_k that has proven effective in practice (Held et al., 1984) and which is also used in Wong et al (2006) is,

$$t_k = \frac{\lambda_k Z_D(\alpha^k) - Z^*}{(\Theta^{\min} - \Theta(x))^2} \qquad (5)$$

In this formula, Z* is the objective value of the best known feasible solution to the original problem and λ_k is a scalar chosen between 0 and 2. Usually, the sequence λ_k is determined by starting with λ_k=2 and reducing λ_k by a factor 2 whenever $Z_D(\alpha)$ has failed to decrease in a specified number of iterations. Usage of this formula has been justified in Held, Wolfe and Crowder (1974). The feasible value Z* initially can be set to 0 and then updated using the solutions that are obtained on those iterations in which the Lagrangian problem solution turns out to be feasible in the original problem. That is also the starting point and method for Z* that we will apply in our implementation of the stepsize function.

Appendix C describes how the subgradient method has been added to existing software programmed in Fortran for optimizing the Buffer Allocation Problem.

4.3 Performance bounds: optimization of the Lagrange multiplier

We have used enumeration to derive the optimal buffer allocation for the BAP. This can only be done for small instances of the problem as the state space for larger instances quickly becomes too large to handle in a reasonable amount of time. Appendix B shows the code that has been written in VBA in order to generate all possible buffer allocations for a specified queueing network and to export these to separate input files for the evaluation program of the BAP.

4.4 Results and conclusions

4.4.1 Experiment setup

We have run our implementation of the BAP, containing the subgradient method for setting the Lagrange multiplier, (BAP_opt_evaluation) for different networks and with different parameter settings. We will compare the results generated in these runs with:

(1) The implementation of the BAP in Cruz et al. (2007) and Smith et al. (2006) which have a fixed Lagrange multiplier/cost factor α of 1000.
(2) Optimal performance as generated by our enumeration and evaluation tools (BAP_enum_evaluation). These will only be generated for the networks with 3 and 5 nodes due to the large computational time that this requires.

For different configurations of the series topology we will look at the following (performance) indicators:

1) The optimal number of buffers (the value of the objective function of the original problem formulation, equation (1) in section 4.1)
2) $L(\alpha)$ (the value of the objective function of the relaxed problem, equation (4) in section 4.1)
3) The throughput of the network

The BAP_opt_evaluation and the original BAP program produce symmetric buffer allocations, i.e. each node has the same number of buffers. The evaluation tool, however, evaluates all possible allocations over the specified number of nodes and can thus find an optimal allocation (including non-symmetrical ones). For the optimization programs we will thus only denote the total number of buffers.

4.4.2 Results

4.4.2.1 Series topologies with 3 nodes

Arrival rate = 2.0
Service rate: 10.0

		BAP_opt_evaluation			Original BAP		
s^2	Total # buffers	Buffer allocation	Through put	$L(\alpha^*)$	Total # buffers	Through put	$L(\alpha^*)$
0.5	6	2, 2, 2	1.9900	3.849	9	1.9980	1000
1.0	6	2, 2, 2	1.9870	-7.594	9	1.9970	1000
1.5	6	2, 2, 2	1.9840	2.653	9	1.9970	1000

Arrival rate = 4.0
Service rate: 10.0

		BAP_opt_evaluation			BAP		
s^2	Total # buffers	Buffer allocation	Through put	$L(\alpha^*)$	Total # buffers	Through put	$L(\alpha^*)$
0.5	6	2, 2, 2	3.7960	2.584	9	3.9960	1000
1.0	6	2, 2, 2	3.6830	5.257	9	3.9940	1000
1.5	6	2, 2, 2	3.5830	3.427	9	3.9910	1000

Arrival rate = 8.0
Service rate: 10.0

		BAP_opt_evaluation			BAP		
s^2	Total # buffers	Buffer allocation	Through put	$L(\alpha^*)$	Total # buffers	Through put	$L(\alpha^*)$
0.5	6	2, 2, 2	5.878	-2.396	9	7.9880	1000
1.0	6	2, 2, 2	5.262	-1.966	9	7.9810	1000
1.5	6	2, 2, 2	4.8500	5.181	9	7.9740	1000

4.4.3.1 Series topologies with 5 nodes

Arrival rate = 2.0
Service rate: 10.0

		BAP_opt_evaluation			BAP		
s^2	Total # buffers	Buffer allocation	Through put	$L(\alpha^*)$	Total # buffers	Through put	$L(\alpha^*)$
0.5	6	2, 2, 2, 2, 2	1.9830	9.359	15	1.9970	1000
1.0	6	2, 2, 2, 2, 2	1.9790	5.420	15	1.9960	1000
1.5	6	2, 2, 2, 2, 2	1.9740	4.361	15	1.9940	1000

Arrival rate = 4.0
Service rate: 10.0

		BAP_opt_evaluation			BAP		
s^2	Total # buffers	Buffer allocation	Through put	$L(\alpha^*)$	Total # buffers	Through put	$L(\alpha^*)$
0.5	6	2, 2, 2, 2, 2	3.6830	7.787	9	3.9940	1000
1.0	6	2, 2, 2, 2, 2	3.6230	8.663	9	3.9960	1000
1.5	6	2, 2, 2, 2, 2	3.5670	7.968	9	3.9920	1000

Arrival rate = 8.0
Service rate: 10.0

		BAP_opt_evaluation			BAP		
s^2	Total # buffers	Buffer allocation	Through put	$L(\alpha^*)$	Total # buffers	Through put	$L(\alpha^*)$
0.5	6	2, 2, 2, 2, 2	5.2620	-1.833	9	7.9810	1000
1.0	6	2, 2, 2, 2, 2	5.0570	9.000	9	7.9740	1000
1.5	6	2, 2, 2, 2, 2	4.8860	-6.089	9	7.9730	1000

4.4.3.1 Series topologies with 7 nodes

Arrival rate = 2.0
Service rate: 10.0

		BAP_opt_evaluation			BAP		
s^2	Total # buffers	Buffer allocation	Through put	$L(\alpha^*)$	Total # buffers	Through put	$L(\alpha^*)$
0.5	6	2, 2, 2, 2, 2, 2, 2	1.9770	10.605	9	1.9960	1000
1.0	6	2, 2, 2, 2, 2, 2, 2	1.9710	9.687	9	1.9940	1000
1.5	6	2, 2, 2, 2, 2, 2, 2	1.9650	-9.848	9	1.9920	1000

Arrival rate = 4.0
Service rate: 10.0

		BAP_opt_evaluation			BAP		
s^2	Total # buffers	Buffer allocation	Through put	$L(\alpha^*)$	Total # buffers	Through put	$L(\alpha^*)$
0.5	6	2, 2, 2, 2, 2, 2, 2	3.5830	11.807	9	3.9910	1000
1.0	6	2, 2, 2, 2, 2, 2, 2	3.5090	10.678	9	3.9940	1000
1.5	6	2, 2, 2, 2, 2, 2, 2	3.5670	-9.515	9	3.9890	1000

Arrival rate = 8.0
Service rate: 10.0

		BAP_opt_evaluation			BAP		
s^2	Total # buffers	Buffer allocation	Through put	$L(\alpha^*)$	Total # buffers	Through put	$L(\alpha^*)$
0.5	6	2, 2, 2, 2, 2, 2, 2	4.8500	-4.496	9	7.9740	1000
1.0	6	2, 2, 2, 2, 2, 2, 2	4.6400	-11.132	9	7.9640	1000
1.5	6	2, 2, 2, 2, 2, 2, 2	6.4400	12.021	9	7.9620	1000

4.4.4 Conclusions

The buffer allocation models compared

The buffer allocations produced by our adjusted BAP program (BAP_opt_evaluation) are indeed symmetric. Furthermore, we also see that the relationship between throughput and the other relevant variables is in both models the same:

- an increase of the variation in the service time in both models results in a lower throughput time (with the buffer allocations remaining practically unchanged).
- another observation (albeit much more visible for the BAP_opt_evaluation model) is that as the utilization rate increases the deviation from the threshold throughput also increases

These results verify that the underlying (Powell) optimization and evaluation algorithms work the same in both models. For more on the relationships between these we refer to Smith et al (2006) and Cruz et al (2007). We will focus on explaining the difference between our BAP programs as a result of the difference in how the Lagrange multiplier is set.

Lagrangian relaxation and the tightness of the lower bound

The main difference between our two models for optimizing the BAP is that in the original program used in Smith et al. (2006) and Cruz et al. (2007) the Lagrange multiplier is fixed, whereas in our adaptation to this model we have optimized the Lagrange multiplier using the subgradient method as discussed in Fisher (1985). We will compare the results from both programs with optimal results as has been derived via enumeration for one setting. Figure 4.2 shows the results for a network with 3 nodes, a squared coefficient of variation of 0.5 for the service time, and a service rate of 2.0.

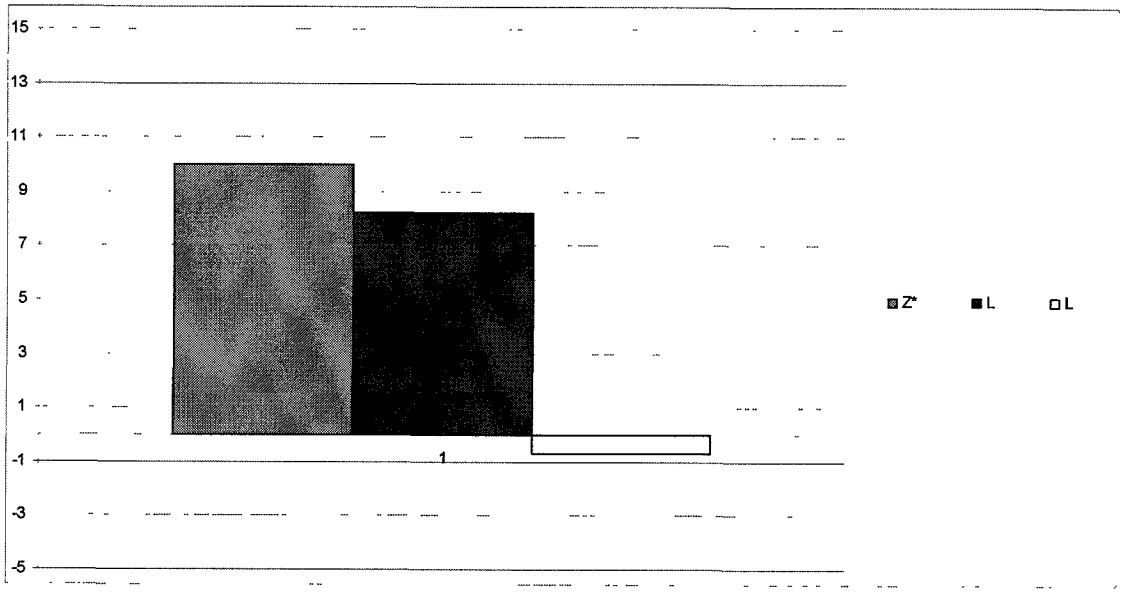

Figure 4.2 comparison of BAP and optimal results for the 5 node series network (s^2=1.0, service rate = 2.0)

The column to the left depicts the minimum number of buffers (the optimal value of the objective function of the original value) derived via enumeration that satisfies the threshold throughput (1.98; choosing a lower threshold results in a very negative value for the BAP with a Lagrange multiplier of 1000).
The bar in the middle shows $L(\alpha^*)$ (the value of the objective function of the dual problem), when the Lagrange multiplier is derived with the subgradient method. And the bar to the right shows $L(\alpha)$ when we have a Lagrange multiplier of 1000 as in the original BAP program. From these results we can clearly conclude the BAP program with the subgradient method for setting the Lagrange multiplier produces a much tighter lower bound for optimal performance.

In order to quantify increase in tightness of the lower bound due to the use of the subgradient method, we will look at the relative gap between the lower bound and the results from enumeration. We will use the following formula to calculate this gap:

$$\text{Relative gap} = \frac{optimal\ \#buffers - L(\alpha)}{optimal\ \#buffers} * 100$$

A small relative gap means that the value of the Lagrangian is close to the optimal value of the objective function of the BAP. We will calculate this relative gap again using the results obtained with both methods for setting the Lagrange multiplier for the 3 node network. When this multiplier is set with the subgradient method, we get a (average) relative gap of

$$\frac{10-7{,}7}{10} * 100 = 23$$

When we use a fixed Lagrange multiplier of 1000, our (average) relative gap equals

$$\frac{10-(-0.9)}{10} * 100 = 109$$

Using the relative gap as a performance measure we can draw the careful conclusion that using the subgradient method for optimizing the Lagrange multiplier has resulted in a 78% reduction of our gap. More testing and model validation is required in order to further gain insight into the performance improvement due to the subgradient method.

5. Conclusions

In chapter 2 we formulated the following three research questions that we would address in this project:

1. Is it possible to use a decomposition and column generating (DCG) approach to solve the nonlinear multi-objective Buffer Allocation Problem and if so, how does this improve the performance of the optimization procedure?

2. How can we determine the Lagrange multiplier for the Lagrange relaxation that yields a tight lower bound for optimal performance of the BAP?

3. How close is this bound to the buffer allocation that results in to optimal performance for the BAP?

For each question we will now evaluate to which extent we have been able to provide an answer.

1. Is it possible to use a decomposition and column generating (DCG) approach to solve the nonlinear multi-objective Buffer Allocation Problem and if so, how does this improve the performance of the optimization procedure?

In chapter 3 we argued that, although the BAP and models for which DW-like decomposition has been used show similarities we do not expect major benefits for the BAP. The main focus for this project was therefore shifted to investigating how existing optimization methods based on Lagrange relaxation could be further improved.

2. How can we determine the Lagrange multiplier for the Lagrange relaxation that yields a tight lower bound for optimal performance of the BAP?

We have used a subgradient method to find tight bounds for optimal performance of the BAP. The subgradient optimization method is an iterative procedure for producing good Lagrange multipliers. It find the Lagrange multiplier that optimizes the following

problem. $Z_D(\alpha) = \underset{\alpha \geq 0}{\text{Max}}\ L(\alpha) = \min\left[\sum_i x_i + \alpha\left(\Theta^{\min} - \Theta(x)\right)\right]$

3. How close is this bound to the buffer allocation that results in optimal performance of the BAP?

We have used the relative gap as a performance measure and based on the results generated we carefully conclude that that using the subgradient method for optimizing the Lagrange multiplier has resulted in a 77% reduction of our gap. More testing and

validation of the model is, however, required before a firmer conclusion can be drawn about the improvement potential caused by the subgradient method.

REFERENCES

Verschuren, O. & Doorewaard, H. (1995), "Het ontwerpen van een onderzoek" ("The design of a research project"). Utrecht: Lemma

Smith, J. MacGregor and Chikhale, N. (1995), " Buffer allocation for a class of nonlinear stochastic knapsack problems", Annals of Operations Research 58, 232-360

Cruz F.R.B., Duczmal L., Woensel T. van, MacGregor Smith J., (2007) "A Multi-objective Approach for Buffer Allocation in General Queueing Networks" short paper

Cruz, F. R. B., A.R. Duarte, T. van Woensel (2007), "Buffer allocation in general single-server queueing networks", Computers & Operations Research 1-16 (in press)

Smith, J. MacGregor, Cruz, F. R. B., T. van Woensel (2006), "Topological Network Design of General, Finite, Multi-Server Queueing Networks.

Takahashi, R.H.C., J.A. Vasconcelos, J.A. Ramirez, L. Krahenbuhl (2003). "A multi-objective methodology for evaluating genetic operators." IEEE Transactions on Magnetics 39 (3) 1321-1324

Takahashi, R.H.C., R.M. Palhares, D.A. Dutra, L.P.S. Gonalves (2004). "Estimation of Pareto sets in the mixed h_2/h_∞ control problem" International Journal of Systems Science 35 55-67

Fisher, M.L. (1985). An applications Oriented Guide to Lagrangian Relaxation. Interfaces 15. 10-21

Held, M. Wolfe, P. Crowder. H,. (1974). "Validation of Subgradient Optimisation". Mathematical Programming" 6. 62-88

Wong, H., Houtum, G.J. van., Cattrysse, D., Oudheusden, D. van,. (2006) "Multi-item spare parts systems with lateral transshipments and waiting time constraints." European Journal of Operational Research 171. 1071-1093

Kranenburg, A.A., Van Houtum, G.J., (2007) "Effect of commonality on spare parts Provisioning costs for capital goods." Int. J. Production Economics, in press

Kranenburg A.A., (2006), "Spare parts inventory control under system availability constraints" Technische Universiteit Eindhoven, dissertation

Operations, Planning, Accounting, and Control (OPAC) research group website (goals statement), "http://w3.tm.tue.nl/en/subdepartments/opac/research/introduction/"

Dantzig G.B., Wolfe P. (1960), "Decomposition principles for linear programs" Operations Research 8, 101-111

Kerbache, L., MacGregor Smith, J. (1986) "The generalized expansion method for open finite queueing networks" European Journal of Operational Research 32. 448-461

List of figures and tables

Figures

Tables

Appendix A Dantzig-Wolfe decomposition

This appendix contains a description of the DW-algorithm, which can be used to solve large-scale LP-problems.

We will use matrix formulation to describe the goal-function and constraints in an LP problem. A normal variable will be distinguished from a matrix by writing the latter in bold.

The Dantzig-Wolfe decomposition approach is based on the transformation of an original LP problem of the form

Maximize $\mathbf{cx}$

subject to $\mathbf{Ax} = \mathbf{b}$

$1<\mathbf{x}<\mathbf{u}$

Into an equivalent master-problem

Maximize $\tilde{\mathbf{c}}\tilde{\mathbf{x}}$

subject to $\tilde{\mathbf{A}}\tilde{\mathbf{x}} = \tilde{\mathbf{b}}$

$1<\mathbf{x}<\mathbf{u}$

With **A** having fewer rows but typically many more columns.

Constructing the master problem can be a tedious task for large-scale problems. By using a technique called delayed column generation, the complete matrices $\tilde{\mathbf{A}}$ and $\tilde{\mathbf{c}}$ no longer have to be known explicitly in order to solve the original problem with the DW algorithm. This technique will be described in the next section.

A.1 The algorithm

A.1.1 Delayed column generation

The master problem is solved by applying the revised simplex method to only a part of the master problem. This part consists of a matrix **B**, a row vector $\tilde{\mathbf{c}}_B$ which is made up of the elements in the objective function corresponding to the row numbers in matrix **B**, and finally, column vector $\tilde{\mathbf{x}}_B$. The column vector $\tilde{\mathbf{x}}_B$ satisfies the equation $\mathbf{B}\tilde{\mathbf{x}}_B = \tilde{\mathbf{b}}$. The matrix B is nonsingular and consists of m'+1 columns of $\tilde{\mathbf{A}}$.

With only the knowledge of the above part of the master problem we now need to find and construct the entering column from the (still unknown) remaining part. Delayed column generation is used to do this.

Step 1
The first step of delayed column generation is to solve the system

$$\mathbf{yB} = \tilde{\mathbf{c}}_{\mathbf{B}}$$

Step 2
In step 2 we calculate

c-y'A', where **y'** is obtained by deleting the last component from **y**, $y_{m'+1}$. **A'** is obtained by splitting the constraint set **A** of the original problem into a part **A'** and a second part **A''**. This rewriting of the constraints of the original problem is done before applying the algorithm.

Next, we solve the subproblem

(c-y'A')x

subject to **Ax = b**

1<x<u

Solving the subproblem can result in three different outcomes that determine the next step in the algorithm:

1. The subproblem has an optimal solution **x*** such that **(c-y'A')x*>** $y_{m'+1}$.
2. The subproblem is unbounded.
3 The subproblem has an optimal solution **x*** such that **(c-y'A')x*>** $y_{m'+1}$.

In the first case a normal basic feasible solution **v** is found and the entering column is described by

$$\tilde{\mathbf{a}} = \begin{bmatrix} \mathbf{A'v} \\ 1 \end{bmatrix}$$

of $\tilde{\mathbf{A}}$, with the corresponding component of $\tilde{\mathbf{c}}$ equal to **cv.**

In the second case a basic feasible direction **w** is found, giving rise to the entering column

$$\tilde{\mathbf{a}} = \begin{bmatrix} \mathbf{A'w} \\ 0 \end{bmatrix}$$

In the third case the solution found to the subproblem is optimal and the algorithm stops. In the first and second case the algorithm proceeds to the next step of finding the leaving column.

Step 3
In order to find the leaving column, first the system $\mathbf{Bd}=\tilde{\mathbf{a}}$ is solved

Step 4
The column vector $\tilde{\mathbf{x}}_B$ is divided by $\mathbf{d}^T$. The column with the number corresponding to the number of the element with the lowest ration is selected to be the leaving column.

Step 5
A new $\mathbf{B}$, $\tilde{\mathbf{c}}_B$, and $\tilde{\mathbf{x}}_B$ are the result of the previous steps and a new iteration is started.

A.1.2 The decomposition algorithm

The above steps form the decomposition algorithm.

A.1.3 Initialization

The initial $\mathbf{B}$ and $\tilde{\mathbf{c}}_B$ can be found by applying the two-phase simplex method to the master problem.

Appendix B VBA code for enumeration

Figure B.1 shows the VBA code that has been used to program the enumeration procedure for a network configuration with three nodes. Using this code all possible buffer allocations are generated for the specified network. Enumeration has also been carried out for network configurations with five, and seven nodes. The network structure can easily be adapted by adding or deleting variables and updating the summation formula (which is used to guarantee that the allocation does not exceed the maximum allowed number of buffers).

```
Public Sub Enumeration()
Dim i As Integer
Dim j As Integer
Dim k As Integer
Dim l As Integer
Dim m As Integer
Dim intMaxValue As Integer
MaxValue = ActiveSheet.Cells(1, 2).Value
m = 3
For i = 0 To MaxValue
 For j = 0 To MaxValue
  For k = 0 To MaxValue
   For l = 0 To MaxValue
   If i + j + k + l < MaxValue Then
    With ActiveSheet
      .Cells(m, 1).Value = i
      .Cells(m, 2).Value = j
       .Cells(m, 3).Value = k
      .Cells(m, 4).Value = l
      .Cells(m, 5).Value = i + j + k + l
    End With
    m = m + 1
    End If
   Next l
  Next k
 Next j
Next i
End Sub
```

Figure B.1 VBA code for enumeration for a network with three nodes

Figure B.2 shows the code that has been used to export each buffer allocation generated by the enumeration program to a separate input file (.txt format) and create a batch file (.bat format) that contains a list of commands with which a large number of evaluations can be performed easily.

```
Sub OneRowPerCellPerrow()
  Dim newrange As Range
  Dim cell As Range
  Dim filename As Variant
  Dim filenam As Variant
  Dim retVal As Variant
  Dim suffix As String
  suffix = ""
  Dim irow As Long
  Dim row_id As String
  Dim rowRange As Range
  For irow = 19 To ActiveSheet.UsedRange.Rows.Count
     Set newrange = Intersect(Cells.Rows(irow), ActiveSheet.UsedRange)
    row_id = Left(Cells(irow, 1).Address(0, 0), _
       Len(Cells(irow, 1).Address(0, 0)))
    filename = "N:\Afstudeerproject 2\enumeration\input\input_" & row_id & ".txt"
    filenam = "N:\Afstudeerproject 2\enumeration\input\series.bat"
    If UCase(Right(filename, 4)) = ".HTM" Then suffix = "<br>"
    Close #1
    Close #2
    Open filename For Output As 1 'open the input file for each buffer allocation
    Open filenam For Append As 2 'open the batch file
    Print #1, "NUMBER OF QUEUES NQ"
    Print #1, Cells(2, 5).Text
    Print #1, "NUMBER OF ARCS NARCS"
    Print #1, Cells(3, 5).Text
    Print #1, "STARTING NODE AND ENDING NODE FOR EACH ARC NS(I) NF(I)"
    Print #1, Cells(4, 5).Text
    Print #1, Cells(5, 5).Text
    Print #1, "ROUTING PROBABILITY ALONG EACH ARC RP(I)"
    Print #1, Cells(6, 5).Text
    Print #1, "NUMBER OF SOURCE NODES NSNOD"
    Print #1, Cells(7, 5).Text
    Print #1, "SOURCE NODES SOUR(I)"
    Print #1, Cells(8, 5).Text
    Print #1, "ARRIVAL RATE TO QUEUE # SLAM(J)"
    Print #1, Cells(9, 5).Text
    Print #1, "NUMBER OF END NODES NEND"
    Print #1, Cells(10, 5).Text
    Print #1, "END NODES ENDD"
    Print #1, Cells(11, 5).Text
    Print #1, "SERVICE RATE OF EACH QUEUE RATE(I)"
    Print #1, Cells(12, 5).Text
    Print #1, "VECTOR FOR PARALLEL SERVERS"
    Print #1, Cells(13, 5).Text
    Print #1, "FINITE BUFFER VECTOR"
```

```
    For Each cell In newrange
      If Trim(cell.Text) <> "" Then
        Print #1, cell.Text & suffix & " ";
      End If
    Next cell
    Print #1, ""
    Print #1, ""
    Close #1
    Print #2, "gen_tput5 < input_" & row_id & ".txt" & " >> series.txt"
    Close #2
  Next irow

End Sub
```

Figure B.2 VBA code for exporting input files

Appendix C Extension of the BAP with the subgradient method for optimization of the Lagrange multiplier

This appendix describes the iterations of the subgradient method that have been added to an existing Fortran program for the BAP. (Comments corresponding to this description have been added to the actual code in order to make the programming code more understandable).

We can distinguish three different parts in our implementation of the subgradient method. The first part ensures that the initial values for the (starting) Lagrange multiplier, scalar in the stepsize updating function, and the "first best-known solution to the original BAP problem" are set according to the description of the subgradient method in section 4.2. The second part updates the Lagrange multiplier (using updating formula (4) on page 25) as long as each new iteration (i.e. each minimization cycle with a new Lagrange multiplier) results in a tighter bound.
The third and last part adjusts the scalar in the formula for the stepsize (formula (5) on page 25) when an update of the Lagrange multiplier does not result in a tighter bound. After each adjustment the scalar is reduced by a factor 2 according to the method in Held (1974), which has been shown to perform well in achieving a convergence to the optimal Lagrange multiplier. This part (and application of the subgradient method) terminates when a continuation will result in the scalar becoming smaller than a specified threshold value (in our case a value of 0.1).

The code that has been added to the BAP program using Fortran as a programming language is shown in figure C.1.

```
C
C         HERE WE RE-INITIALIZE THE STARTING VECTOR
C         WITH THE NUMBER OF RUNS VARIABLE INCREMENT
C
          DO 90 I=1, NQ
              W(I) = WINT(I) + IJ
              V(I) = 0.0
   90     CONTINUE
  100 CONTINUE
C
C     OPTIMIZATION OF LAGRANGE MULTIPLIERS, MAY 24TH 2007 BY JAMEL FEDDAHI
C     SUBGRADIENT METHOD START AFTER DOING ONE POWELL OPTIMIZATION ITERATION
C     (WITH THE INITIAL LAGRANGE MULTIPLIER/COST FACTOR SET TO 0, ACCORDING TO
C     THE GUIDELINES IN FISHER (1986)) THE FIRST IF PART IS RUN AFTER THIS FIRST
C     ITERATION  IT RECORDS THE BEST SOLUTION BELONGING TO THE INITIAL ITERATION,
C     INITIALLY ASSIGNS THE VALUE 2.0 TO THE SCALAR IN THE STEPSIZE FUNCTION,
C     UPDATES THE LAGRANGE MULTIPLIER AND REINITIALIZES POWELL

          IF (AMU  EQ. 0 ) THEN
          BESTFXOLD = BESTFX
          SZSCAL = 2.0
          STPSZ = SZSCAL*(SUM-0)/(PENALTY**2)
          AMU = MAX(0.0, AMU-STPSZ*PENALTY)
          GO TO 207

C     THE SECOND IF PART IS RUN AFTER THE PREVIOUS INITIALIZATION STEPS· IT CHECKS WHETHER
C     THE PREVIOUS ITERATION RESULTED IN A TIGHTER BOUND (i e  THE OBJECTIVE FUNCTION
C     VALUE AFTER RUNNING POWELL IS NOW HIGHER THAN IN THE PREVIOUS ITERATION),
C     IF IT DID IT UPDATES THE LAGRANGEMULTIPLIER ACCORDING TO THE RULE IN FISHER (1986)

          ELSE IF ((BESTFX-BESTFXOLD) .GT  0 ) THEN
          BESTFXOLD = BESTFX
          STPSZ = SZSCAL*(SUM-(SUM + AMU*PENALTY - nserv*cserv))/(PENALTY**2)
          AMU = MAX(0 0, AMU-STPSZ*PENALTY)
          GO TO 207

C     THE LAST IF PART IS RUN WHEN THE PREVIOUS ITERATION DID NOT RESULT IN A TIGHTER BOUND
C     THE SCALAR IN THE STEPSIZE FUNCTION IS REDUCED BY A FACTOR 2 AND THE LAGRANGE
C     MULTIPLIER IS UPDATED AS BEFORE THIS IS DONE UNTIL THE SCALAR WOULD BECOME
C     SMALLER THAN 0.1 (AN ARBITRARY SELECTED VALUE) TO PREVENT THIS FROM RUNNING
C     FOR EVER  AFTER THIS HAPPENS THE PROGRAM CONTINUES TO EXPORTING THE OUTPUT

          ELSE IF (SZSCAL .GT. 0.1) THEN
          BESTFXOLD = BESTFX
          SZSCAL = SZSCAL/2.0
          STPSZ = SZSCAL*(SUM-(SUM + AMU*PENALTY - nserv*cserv))/(PENALTY**2)
          AMU = MAX(0.0, AMU-STPSZ*PENALTY)
          GO TO 207

          ENDIF

C
C

  106     WRITE(4,*) 'BEST SOLUTION:'
          WRITE(4,*) 'QUEUE VECTOR'
          WRITE(4,110) (NINT(BESTW(I)),I=1,NQ)
  110     FORMAT(10(I2,','))
```

Figure C.1 Fortran code for the subgradient method extension to the BAP