A Study of Knapsack-based Admission and Allocation Techniques

by

Rafael Parra Hernández
B.Ind.Eng., Instituto Tecnológico de Veracruz (México), 1993
M.Sc., Cenidet (México), 1996

A Dissertation Submitted in Partial Fulfillment of the
Requirements for the Degree of

DOCTOR OF PHILOSOPHY

in the Department of Electrical and Computer Engineering

Supervisor: Dr. Nikitas J. Dimopoulos

ABSTRACT

The allocation of resources among various project, units, or users is accomplished through the use of a systematic mechanism called resource allocation. The types of resources vary, depending upon the system under consideration. For instance, frequency spectrum and transmitter power might be the resources needed to allocate in an efficient manner on a cellular network system, so that the number of mobile users attended is maximized. On a Grid computing system, one needs to allocate resources such as processors, memory, disk space, and so on, in order that computational tasks run in the most efficient manner.

In order to evaluate resource allocation techniques performances, we first need to evaluate whether a particular resource allocation problem can be cast in the mathematical formulation we are exploring. We also need to decide which mechanism will be used, or if a new one needs to be constructed to solve the particular formulation. Finally, we need to evaluate whether the solution obtained is better than those obtained from other techniques that might express and solve the allocation problem in different ways.

In this dissertation, we propose a new resource allocation technique for a system described only by a formulation known as the Multichoice Multidimensional Knapsack Problem, or MMKP. We also propose and evaluate resource allocation techniques on two other systems: a cellular network and a Grid computing system; in this regard, the resource allocation problem is not expressed as an MMKP, although the formulations used are particular cases of the MMKP. The MMKP formulation is not applied because its use would not have allowed us to make a fair performance comparison with other more commonly used allocation techniques. However, we believe that as more complex tasks are demanded from systems where resource allocation mechanisms are needed, an MMKP formulation could more suitably represent the allocation problem.

Numerical results indicate that the resource allocation techniques explored in this work present a better performance than previous techniques. Numerical results also indicate that the use of the proposed techniques and the use of suitable optimization criteria can be used to achieve a number of resource allocation goals.

Supervisor: Dr. N. J. Dimopoulos, (Department of Electrical and Computer Engineering)

Table of Contents

List of Tables

List of Figures

Acknowledgement

Beginnings and endings are inseparable. My adventure of going after a doctorate degree began years ago. As it ends, I wish to acknowledge those who contributed most to my achieving this goal.

First of all, I want to express my sincere appreciation to my supervisor, Professor Dr. Nikitas J. Dimopoulos, for his complete support and guidance throughout this research. This project would not have been possible without you. Thank you.

I am also indebted to the graduate students and faculty members with whom I have become friends, and shared discussions about our research. Thank you, especially the fellows at LAPIS. I will miss you all.

I would also like to thank the members of my dissertation committee, Dr. Panajotis Agathoklis, Dr. Koen Bertels, Dr. Kin F. Li, Dr. Wu-Sheng Lu and Dr. Gholamali C. Shoja, for taking time from their busy schedules to evaluate this work, and for their useful comments.

This research would not have been possible without the financial support of the National Sciences and Engineering Research Council of Canada.

Finally, I thank my family for their unconditional love and support during all these years.

Dedication

To my parents:

Leonor and Rafael

Chapter 1

Introduction

1.1 Resource Allocation

Allocating resources among various projects, units, or users, is accomplished through a systematic procedure called resource allocation. Although the resources to be allocated vary depending on the system under consideration, if optimal allocations are sought efficient resource allocation procedures are needed. For instance, assume that a group is working on a manufacturing design project, that the project contains n tasks, and that the number of persons working on the project equals m. Let us further assume that each of the n tasks can be assigned to a subset of persons simultaneously working on that particular task; for example, task 1 could be handled by persons a and b, or by persons a, b and c, or by a single individual. The processing time for each task will depend on the number of persons assigned. Assigning n tasks to m individuals in order to minimize the project finishing time is known as the multiprocessor task scheduling problem. Scheduling problems are concerned with how to allocate resources to tasks over time.

Another example of resources needing to be allocated in an efficient manner is found in the bin packing problem. Here we have a group of items with different volumes that must be packed into a finite number of bins with capacity V, and in such a way that the least number of bins is used (see Figure 1.1).

The examples given above are nondeterministic polynomial (NP) problems. That is,

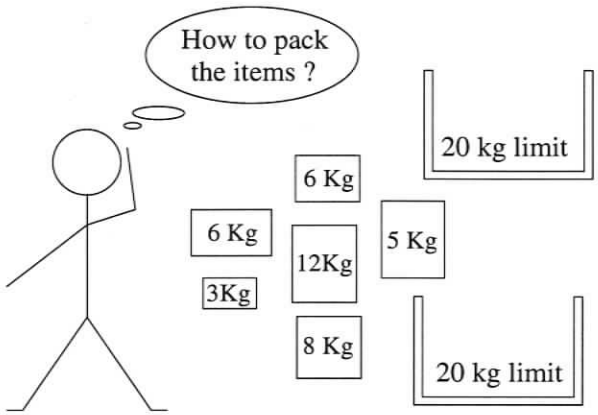

Figure 1.1. *Bin packing problem*

although the solution to an NP problem can be verified in polynomial time, no particular rule is followed to construct such a solution. There is no known algorithm that solves to optimality such difficult problems in polynomial time. To solve such problems in short periods of time we can use either an approximation or a heuristic algorithm; however, optimality can not be assured.

In the examples described above, it is clear that an efficient way to allocate people to perform a task, or bins to pack our items, will allow us to use the resources at hand in an optimal manner. That is to say, an optimal allocation increases the performance of a system (manufacturing, packing, and so on). The purpose of this dissertation is to study resource allocation techniques on the set of systems briefly described below. The resource allocation problems in these systems are NP, and heuristic approaches are used to solve them.

1.1.1 On a General System

In this case, the system is described only by a mathematical formulation; that is, we assume that a mapping mechanism is used to map the resource allocation problem from an ad-hoc system into a formulation known as the Multichoice Multidimensional Knapsack Problem (MMKP). The MMKP is solved by using the new heuristic technique presented

in this dissertation. Simulation results indicate that this technique produces better solutions than previous ones, although the price for obtaining a better performance is a higher computational complexity.

As in any general optimization problem, the MMKP has an objective function that needs to be maximized, and a set of equality and inequality constraints that need to be satisfied. The solution value for each variable is bounded, with either 0 or 1 being the only possible outcomes. The MMKP could be represented as

$$\begin{array}{ll} \text{maximize} & \text{OBJECTIVE FUNCTION} \\ \text{subject to:} & \text{INEQUALITY CONSTRAINTS} \\ & \text{EQUALITY CONSTRAINTS} \\ & \text{VARIABLES} \in \{0,1\} \end{array}$$

1.1.2 On a Cellular System

In this case, the system is a cellular network with power control capability. A cellular network is a mobile network on which resources are managed in cells. Each cell is a bounded area, and is served by an antenna or base station. The base station provides frequency channels to mobile stations. In a cellular system with power control capability, power is the medium used to control the interference among mobile stations, so that, in principle, all channels are available for use at all base stations. Let us say that we have the following situation in the cellular system,

- User α is already allocated channel 1 (*Ch_1*) at base station 1 (*BS_1*).
- *Ch_2* is available for use at *BS_1*.
- At *BS_2* (next to *BS_1*) user β needs to be allocated a channel. *Ch_1* is the only available channel at *BS_2*.

Regarding user β, let us say that we notice that it cannot be allocated *Ch_1* because the amount of power needed to carry the transmission on (and to avoid interference with user

α, which is using the same channel at *BS_1*) is greater than the amount of available power. Having user α on *Ch_1* might not be the best allocation; perhaps, if user α were allocated *Ch_2* (at *BS_1*), then user β could be allocated *Ch_1* (at *BS_2*).

On a cellular network system with power control capability, we could express the resource allocation problems as

maximize OBJECTIVE FUNCTION

subject to: FINITE_FREQUENCY_SPECTRUM
FINITE_TRANSMITTER_POWER

where OBJECTIVE FUNCTION describes a predefined metric such as the total number of users, the probability of a successful user allocation, and so on.

1.1.3 On a Grid System

In this case, the system is a Grid computing system. A Grid can be defined as a hardware and software infrastructure that clusters and integrates high-end computers, networks, databases, and scientific instruments from multiple sources in order to form a virtual network on which users can work collaboratively. Computational tasks need to be allocated resources to run on the Grid. A well known allocation strategy is first-come first-served, where tasks are allocated as they arrived on the Grid. If a task cannot run because resources are busy, then that task, and all later ones, will be put in a queue and will remain there until resources are found to allocate them in the order given by their position in the queue. The allocation problem on the Grid could be described as

maximize OBJECTIVE FUNCTION

subject to: GRID AVAILABLE RESOURCES

where OBJECTIVE FUNCTION describes a predefined metric such as the total computation time of the tasks, the total number of tasks, the total value of the tasks, and so on.

1.2 Research Work

To propose a variety of allocation mechanisms (that is, algorithms) and to evaluate their performance based on their ability to optimize predefined metrics is in itself a goal worth pursuing. Nevertheless, we think that the presentation of good engineering examples where such algorithms can be evaluated is equally important. Thus we have decided to explore the systems outlined above. In these systems, it is clear that potential benefits can be obtained if, for example, the number of users in a cellular system is maximized, or if the quality of service a task receives when running in a Grid is increased.

In the initial stages of this research, we explored allocation techniques based on computational intelligence [46, 49]. Although these techniques produced very good allocations, they were not explored further because of the high computational time they require to construct such allocations. This fact forced us to look for strategies that were less computationally expensive and only these mechanisms are reported in this dissertation.

We use variants of the MMKP formulation to express the resource allocation problem on a cellular network system with power control capability and on a Grid computing network system. The variants are particular cases of the MMKP. A new heuristic for the MMKP is also part of the dissertation. The MMKP formulation was not applied on the cellular and Grid systems because it would not have allowed us to make a fair performance comparison with other more commonly used standard allocation techniques.

We believe that, as the complexity of tasks requiring resource allocation mechanisms increases, an MMKP formulation could more suitably represent the allocation problem.

1.3 Dissertation Overview

This document is organized as follows:

Chapter 2 provides background material for the dissertation.

Chapter 3 presents a new heuristic for solving the Multichoice Multidimensional Knapsack Problem (MMKP). Here, the MMKP is reduced to a Multidimensional Knapsack Problem (MKP), then a linear programming relaxation of the MKP is solved, and a pseudo-utility value is computed for each variable in the MMKP. These values and constraint coefficients are used in order to obtain a feasible solution (that is, a solution that satisfies all the constraints). Finally, the quality of the feasible solution is improved, if possible, using a mechanism based on the pseudo-utility values and on the coefficients of the objective function. Numerical results show that the performance of the proposed approach is better than that obtained by previous techniques.

Chapter 4 presents a resource allocation strategy on a cellular network system with power control capability. The resource allocation problem is expressed as a variant of the MMKP. This variant is relaxed into a constrained linear programming problem. A series of pseudo-utility values is obtained from the solution of that problem. These values are used to establish an order by which a series of trials are attempted to evaluate whether a user can be allocated on the system.

Chapter 5 presents a resource allocation strategy on a Grid system. This work formulates the resource allocation problem on Grids as a variant of the MMKP. A notion of utility on a Grid system is also introduced, and is used to effect allocation policies on the Grid. Simulation results indicate that a knapsack formulation can be a more suitable way to express the resource allocation problem.

Chapter 6 brings the dissertation to a conclusion by summarizing our contributions and by providing direction for future work.

Chapter 2

Knapsack Background

2.1 Introduction

This chapter presents background information about an important class of combinatorial optimization problems known as knapsack problems. We first describe different knapsack formulations that are related to one that we are particularly interested in, called the multichoice multidimensional knapsack problem. We then provide a short description of applications, and examples, of where this knapsack formulation has been used. Finally, we briefly describe how some of the problems that especially interest us can be cast using this formulation. The information introduced in this chapter assists in the presentation of Chapters 3–5.

The elements in a knapsack problem are a knapsack of fixed volume, and objects, or items, of different volumes and values. We need to find the most valuable set of items that fit into the knapsack. This problem can be formulated by numbering the items from 1 to l, and by introducing a binary variable to denote whether an item is selected for the knapsack or not, that is,

$$x_j = \begin{cases} 1 & \text{if item } j \text{ is selected;} \\ 0 & \text{otherwise} \end{cases} \tag{2.1}$$

If we also establish that v_j represents the value of item j, r_j its weight or volume,

and b the volume of the knapsack, then from all the possible sets of items that satisfy the constraint

$$\sum_{j=1}^{l} r_j x_j \leq b,$$

we choose the one that maximizes the objective function

$$\sum_{j=1}^{l} v_j x_j$$

Knapsack problems have been intensively studied (e.g., see [41, 52] and references within). From a practical point of view, knapsack formulations can model problems such as cargo loading, stock cutting, capital budgeting, resource allocation, menu planning, and so on.

In this dissertation we use knapsack formulations to cast the resource allocation problem on three systems. We need to point out that expressing a problem in a particular formulation has no purpose if the formulation does not provide a good understanding of the problem, or if the quality of the solution using the formulation is no better than ones obtained by techniques already in use. For systems similar to the ones simulated in this work, we believe a better resource management can be achieved if knapsack formulations are used. This is supported by results presented in this dissertation. We also believe that such formulations could applied to other problems. Although this better resource management demands more computational time, it does fall within the real-time requirements of the systems studied.

2.2 Particular Knapsack Problems

The *0−1 Knapsack Problem* involves selecting a subset of the total l items such that the corresponding value sum is maximized without having the weight sum to exceed the knapsack

size b. A formulation of the $0-1$ Knapsack Problem is given by

$$\begin{aligned} &\max \sum_{j=1}^{l} v_j x_j \\ &\text{subject to:} \quad \sum_{j=1}^{l} r_j x_j \leq b, \\ &\qquad x_j \in \{0,1\} \end{aligned} \tag{2.2}$$

The *Multichoice Knapsack Problem* is a generalization of the $0-1$ Knapsack Problem. This generalization occurs when the item set is partitioned into subsets N_i, $i = 1, ..., n$, and is required to select only one item j from each subset N_i such that the objective function is maximized. The *Multichoice Knapsack Problem* can be formulated as

$$\begin{aligned} &\max \sum_{j=1}^{l} v_j x_j \\ &\text{subject to:} \quad \sum_{j=1}^{l} r_j x_j \leq b, \\ &\qquad \sum_{j \in N_i} x_j = 1, \quad i = 1, ..., n \\ &\qquad x_j \in \{0,1\} \qquad j \in N = \{1, ..., l\} = \bigcup_{i=1}^{n} N_i \end{aligned} \tag{2.3}$$

assuming

$$N_g \bigcap N_h = \emptyset \quad \text{for all } g \neq h$$

The most commonly used form of the Knapsack Problem is the *Multidimensional Knapsack Problem* (MKP). The MKP can be described by

$$\max \sum_{j=1}^{l} v_j x_j \tag{2.4}$$

$$\text{subject to: } \sum_{j=1}^{l} r_{kj} x_j \leq b_k, \quad k = 1, \ldots, m,$$

$$x_j \in \{0, 1\}$$

A combination of the multichoice knapsack problem and the multidimensional knapsack problem generates the *Multichoice Multidimensional Knapsack Problem* (MMKP).

In the MMKP there are n sets of items. There are m inequality constraints and n equality constraints that must be satisfied. The MMKP is to select *only* one item from each set and the goal is to maximize the total value of the selection subject to resource constraints. In this knapsack problem, and the others given above, we assume that v_j, r_{kj} and b_k are positive integers. The MMKP can be formulated as

$$\max \sum_{j=1}^{l} v_j x_j \tag{2.5}$$

$$\begin{aligned} \text{subject to: } & \sum_{j=1}^{l} r_{kj} x_j \leq b_k, && k = 1, ..., m \\ & \sum_{j \in N_i} x_j = 1, && i = 1, ..., n \\ & x_j \in \{0, 1\} && j \in N = \{1, ..., l\} = \bigcup_{i=1}^{n} N_i \end{aligned}$$

assuming

$$N_g \bigcap N_h = \emptyset \quad \text{for all } g \neq h \tag{2.6}$$

The MMKP can be expressed in a different formulation if we introduce a subset-index for the variables; for instance, x_{ij} will represent the j-variable in the i-subset, v_{ij} will

represent the value of the j-variable in the i-subset, and so on. We can also renumber the variables in each subset and denote by l_i the number of variables in the i-subset. Then, the MMKP can be described by

$$\max \sum_{i=1}^{n} \sum_{j=1}^{l_i} v_{ij} x_{ij} \tag{2.7a}$$

$$\text{subject to: } \sum_{i=1}^{n} \sum_{j=1}^{l_i} r_{kij} x_{ij} \leq b_k, \qquad k = 1, ..., m \tag{2.7b}$$

$$\sum_{j=1}^{l_i} x_{ij} = 1, \qquad i = 1, ..., n \tag{2.7c}$$

$$x_{ij} \in \{0, 1\} \tag{2.7d}$$

2.3 Examples using an MMKP Formulation

The MMKP formulation is useful to describe configuration problems that involve options [42, 12, 32, 59]. In [42], the MMKP is used to represent a model for a reconfigurable service of a single session multimedia service. The reconfiguration is carried out by selecting an implementation, among a set of implementations, of the elementary services (audio presentation, video generation, and so on) of a distributed multimedia service, where video conference and video-on-demand are the multimedia applications cited. Each of the implementations requires a number of resources and provides a certain Quality of Service (QoS). When the amount of resources available in the system changes, a different implementation of equal QoS may be selected. If this action is not possible, then an implementation of lower QoS may be selected; in this case, the implementation selected is the one that provides the closest QoS to the desired QoS (given by the user's preferences). The selection of an implementation for each elementary service is represented by the equality constraints in the MMKP formulation. The fact that the amount of resources demanded by the implementations selected must be less than, or equal to, the amount of available resources is expressed by the set of inequality constraints in the MMKP formulation. In [42], the

reconfigurable service model proposed is called the optimally graceful QoS degradation model.

In [32], the MMKP is used to represent a model for a multisession adaptive multimedia system. In this representation, each session provides a quality profile, and the multimedia system has to select the dynamic QoS of each session in order to maximize a predefined metric or utility function. If the amount of resources available in the system changes, then one or more sessions could experience a change in their QoS. In [32], the multisession adaptive multimedia system model is called the Utility Model (UM). Applications of the UM are presented in [12] and [59]. The application described in [12] deals with the problem of bandwidth sharing among applications during real-time multimedia transport. In a specific manner, a media source is encoded into several signal layers, each of which is sent over a distinct network channel. A receiver could select a subset of layers to receive and decode; the application of the UM allows the selection of the subset of layers to be received and decoded based on the available reception bandwidth. On the other hand, in [59] the UM is used to map a model of a packet-switched network. In this case, the resources are the bandwidths and latencies of the network's links and switches. In this application, the UM is used in the design of a network admission controller for the admission or change of traffic flow associated with a service level agreement.

Variants of the MMKP are also used to describe configuration problems. For example, a relaxation of the MMKP, named MRMD (Multiple Resources and Multiple QoS Dimensions problem), is used in [35] to represent the problem of allocating multiple resources in order to satisfy the QoS requirements of different applications, or tasks, along multiple QoS dimensions. In this work, the set of equality constraints is relaxed; that is to say, the allocation mechanism may or may not accept a particular task for processing.

A minimization variant of the MMKP also appears in the Nursing Personnel Scheduling Problem [58], which is defined as the identification of a staffing pattern that

- states the number of nurses of a certain skill class to be scheduled on wards and nursing shifts,
- fulfills the total nursing personnel capacity, integral assignments, minimum staffing requirements, and other constraints needed for the scheduling period,
- permits substitution of tasks among nursing skill classes, and
- minimizes the cost of the nursing care shortage.

This problem is first cast as a mixed-integer quadratic programming problem and then decomposed into a multiple-choice programming master problem (a minimization version of the MMKP), with a number of quadratic programming subproblems. Each subproblem is solved for all feasible combinations of 8-hour nurse shifts, skill classes, and specific wards and shifts. The solutions, and their associated feasible nurse-shift combinations, are used to generate the minimization version of the MMKP. The equality constraints are constructed from the fact that one of the feasible nurse-shift combinations should be selected for a particular ward and for a specific shift. Although the problem is expressed as variant of an MMKP, a relaxation is performed and the problem is solved as a linear programming problem. Later, a post-optimal search is conducted when noninteger solutions are obtained.

We will now proceed to describe two other examples where the MMKP, or variants thereof, could be used to cast the resource allocation problem.

2.3.1 The MMKP Formulation on a Cellular Network System

A cellular network is a mobile network on which resources are managed in cells. Each cell is a bounded area and is served by an antenna or base station (BS). Cell size and shape depend on signal strength, the presence of obstacles to signal propagation, customer capacity, and cost constraints. The cellular concept was proposed in [16]; inside a cell, mobile stations (MS) are able to start/receive communication while moving inside the cellular network. The BS provides frequency channels to the MS. When an active user reaches the

boundary of the cell, that user needs to change its current frequency channel to one belonging to the neighboring cell; this procedure is known as handoff.

Because of the increasing demand for mobile communications services, it is important to maximize the use of limited transmission power and spectrum.

The channel resource allocation/reallocation problem can be expressed as a variant of the MMKP. If a knapsack fashion is used, then a mechanism can be performed to obtain an order by which trials are attempted to evaluate whether a MS can be allocated on the system. Numerical results, from simulations, indicate that this allocation mechanism improves a system's performance when compared with a more commonly used strategy. In general, the resource allocation problem on a cellular system can be expressed as follows:

maximize OBJECTIVE FUNCTION

subject to: FINITE_FREQUENCY_SPECTRUM
FINITE_TRANSMITTER_POWER

Utility values can be used as coefficients in an objective function to describe the different allocation options for a set of users. The utility values can represent the relative merits of each option, and depend on other metrics associated with each option. The values can guide trials that attempt to evaluate whether a MS can be allocated on the network.

It is possible to compute utility values from a MS perspective, or a system perspective, or both. Let us say that we decide to compute the utility values from the system's perspective. That is, if an option is associated with positive utility values, this means that the system perceives a gain; if the association is negative, then a loss. We can establish that when a MS requests a service, the requested service could be subject to any of the following options:

1. REJECTED. The new request is denied. In this case, one can associate a non-positive utility value with this option; if negative, then it represents a penalty the system pays

for not being able to provide the requested service. Under this option the resources consumed are zero.

2. ADMITTED. The request is granted by the corresponding BS. In this case, a positive utility value is associated with the option. The amount of resources to be consumed is greater than zero; that is, actual resources are consumed when a service is provided.
3. QUEUED. In this case, the request is sent to a queue and it will remain there until a final decision (REJECTED or ADMITTED) is taken. Therefore the queue state is a temporary one. If a request is queued, a negative utility value can be associated with this option because there is a consumption of resources by the BS through a control channel (to keep communication with the user). The QUEUED option is a problematic one since it is highly likely that mobile stations on a cellular system will not be willing to spend time in a queue.

On the other hand, when providing a service to a MS, we can assume that the service being provided can keep its current status (MAINTAINED option), or be subject to any of the following options:

1. DROPPED. After a request has been granted, the BS makes an effort to provide that service. Under extreme conditions of overload, the network may decide to drop the service in order to free resources for some other purpose. In such a case, the option can be associated with a non-positive utility value, representing the penalty value the BS needs to pay because of its inability to continue providing the service originally granted.
2. DEGRADED. Under certain load conditions, the BS might need to reduce the quality of the service given to a particular user. In this case, the option can be associated with a positive utility value, representing the service given to the user, albeit lower in quality.
3. UPGRADED. Under other load conditions, the BS may be able to increase the quality

of the service given to a particular user. In this case, the option can be associated with a positive utility value, representing a higher-quality service for the user.

4. REALLOCATED. The services granted are provided through channels. It might be the case that, due to load conditions, the BS decides to reallocate or change the channel of a user. In this case, the option can be associated with a positive utility value, representing the service given to the user.

The options described above can form a set, or group, related to a specific user. One needs to select an option for each user, and the options selected for all the users must satisfy the resource constraints. These are the elements of an MMKP formulation. In Chapter 4, we present a resource allocation technique on a cellular network system with power control capability.

2.3.2 The MMKP Formulation on a Grid Computing System

A Grid could be defined as a hardware and software infrastructure that clusters and integrates high-end computers, networks, databases and scientific instruments from multiple sources, in order to form a virtual network on which users can work collaboratively. A computational Grid is formed by Clients, Servers and a Resource Allocation Centre. Clients generate computational tasks that require resources for their execution; Servers provide these resources. The Clients-Servers resource allocation is performed at the Resource Allocation Centre (RAC).

A task is described by the set of resources needed to conclude its computation (the number of computational units (processors), storage requirements, length of computation, and so on). Along with these resource requirements, each task may also list a number of options, each requiring a different amount of resources. With each option, one can associate a set of metrics to compute a utility value for the particular option; this value can be used as a guide in a task-option selection process.

As a feasible task arrives at the RAC, it can be either:

1. ADMITTED. The computational task runs immediately on the Grid.
2. QUEUED. The task will run when the requested resources become available.

If a task has been ADMITTED (is already running), then it can experience any of the following options:

1. MAINTAINED. The task keeps is current resource allocation.
2. DEGRADED. The computational resources given to the task are reduced.
3. UPGRADED. The computational resources given to the task are increased. This might be the case when the original requested resources were not all granted the moment the task was allowed to run on the Grid.
4. REALLOCATED. The task is reallocated to run on another part of the Grid.

As in the cellular system example, we have here the elements to construct an MMKP; that is, we have sets, or groups, of items represented by the options available for each computational task. One needs to select an option for each task and all options selected must satisfy the resource constraints. In Chapter 5, we present a resource allocation mechanism on a Grid.

In the knapsack problem, coefficients v, r and b are assumed to be positive integers [41]. However, there are formulations where the use of negative coefficients is more suitable (strictly speaking, the use of negative coefficients is not allowed in the knapsack formulation). In particular cases, formulations using negative coefficients could be mapped into standard knapsack formulations, or formulations that resemble the knapsack structure. In the following section, we present a scenario where the resources demanded by the available options, and the value assigned to such options, results in the used of negative coefficients in the objective function. This formulation is mapped into a knapsack problem.

2.4 On the MMKP Formulation

In the options described above one finds that it is possible to have negative coefficients in the objective function (DROPPED and REJECTED options). A knapsack formulation can still be used if we proceed as explained below.

We can assume that no users (such, mobile users or computational tasks) are on the system. If users are being attended, it is as if they were not allocated on the system; therefore, the resources to be considered available in the formulation are the total amount of resources in the system represented by b_k, where $k = 1, ..., m$, denotes different types of resources. A DROPPED option consumes zero resources and may be associated with a negative utility value. The other options (MAINTAINED, REALLOCATED, DEGRADED and UPGRADED) represent a service to the user, and are associated with positive utility values, and implied a consumption of resources.

Let us say that the following equation,

$$
\begin{array}{lcccccccccccl}
 & \multicolumn{5}{c}{\overbrace{\phantom{v_{11}x_{11} + v_{12}x_{12} + v_{13}x_{13}}}^{\textit{Options for User } \alpha}} & & & & & & & \\
 & \multicolumn{5}{c}{\overbrace{v_{11}x_{11}}^{\text{DROPPED}} + \overbrace{v_{12}x_{12}}^{\text{REALLOCATED}} + \overbrace{v_{13}x_{13}}^{\text{DEGRADED}}} & + & v_{21}x_{21} & + & v_{22}x_{22} & + & v_{23}x_{23} & \\
\max f(x) = & & & & & & & & & & & & & \\
\text{subject to:} & & & & & & & & & & & & & \\
 & r_{111}x_{11} & + & & + & r_{113}x_{13} & + & r_{121}x_{21} & & & & & & \leq b_1 \\
 & & + & r_{212}x_{12} & & & & & + & r_{222}x_{22} & & & & \leq b_2 \\
 & & + & r_{312}x_{12} & + & r_{313}x_{13} & & & & & & & & \leq b_3 \\
 & & & & & & & & & & + & r_{423}x_{23} & & \leq b_4 \\
 & x_{11} & + & x_{12} & + & x_{13} & & & & & & & & = 1 \\
 & & & & & & + & \underbrace{x_{21}}_{\textit{Opt. 1}} & + & \underbrace{x_{22}}_{\textit{Opt. 2}} & + & \underbrace{x_{23}}_{\textit{Opt. 3}} & & = 1 \\
 & & & & & & & \multicolumn{5}{c}{\underbrace{\phantom{x_{21} + x_{22} + x_{23}}}_{\textit{Options for User } \beta}} & & &
\end{array}
$$

$$x_{ij} \in \{0, 1\} \tag{2.8}$$

Note: $v_{11} < 0$ and $r_{111} = 0$

where x_{ij} represents the j-option for the i-user, describes the following situation on a system. Assume that at time t the system is already attending a user (*User* α using $[r_{111}]$ resources) and the system receives a request to attend a new user (*User* β). Due to load conditions, the resource allocation mechanism in the system estimates that

- *User* α could be
  1. DROPPED; that is, $[r_{111}]$ resources will be made available ($r_{111} = 0$).
  2. REALLOCATED on $[r_{212}\ r_{312}]^T$ resources.
  3. DEGRADED or experience a reduction in the quality of the service by being reallocated on $[r_{113}\ r_{313}]^T$ resources.
- *User* β could be attended through
  1. Option 1 by allocating $[r_{121}]$ resources to the user.
  2. Option 2 by allocating $[r_{222}]$ resources to the user.
  3. Option 3 by allocating $[r_{423}]$ resources to the user.

As can been seen, the first equality constraint in Equation 2.8 expresses that if an option such as REALLOCATED or DEGRADED is not possible, then the system needs to pay a v_{11} penalty for dropping *User* α. If an option to be attended is found, then no penalty is paid. The problem represents a modified MMKP; the negative coefficients can be eliminated, and heuristics used to solve the MMKP can be modified to solve this modified MMKP. An example follows.

Let us say that the coefficients that represent a particular resource allocation problem are those in Equation 2.9. Two users need to be allocated. The DROPPED option consumes zero resources and has a negative coefficient (penalty) value in the objective function. Four options are available for *User* α and two options for *User* β.

To eliminate the negative coefficient, we solve the equality constraint (first one) for variable x_{11} and substitute its value on the optimization problem:

$$
\begin{array}{lcccccccccccl}
 & \overbrace{\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad}^{\textit{Options for User } \alpha} & & & & & & & & & & & \\
 & \overbrace{\text{DROPPED}}^{} & & \overbrace{\text{REALLOCATED}}^{} & & \overbrace{\text{DEGRADED}}^{} & & \overbrace{\text{MAINTAINED}}^{} & & & & & \\
\max f(x) = & -6x_{11} & + & 5x_{12} & + & 2x_{13} & + & 5x_{14} & + & 2x_{21} & + & 3x_{22} & \\
\text{subject to:} & & & & & & & & & & & & \\
 & 0x_{11} & & & + & 2x_{13} & + & 3x_{14} & & & & & \leq 3 \\
 & & + & 2x_{12} & & & & & + & 4x_{21} & & & \leq 4 \\
 & 0x_{11} & + & 3x_{12} & + & 2x_{13} & + & 4x_{14} & & & + & 2x_{22} & \leq 5 \\
 & x_{11} & + & x_{12} & + & x_{13} & + & x_{14} & & & & & = 1 \\
 & & & & & & & & + & \underbrace{x_{21}}_{\textit{Opt. 1}} & + & \underbrace{x_{22}}_{\textit{Opt. 2}} & = 1 \\
 & & & & & & & & & \underbrace{\qquad\qquad\qquad\qquad}_{\textit{Options for User } \beta} & & &
\end{array}
$$

$$
x_{11}, x_{12}, x_{13}, x_{14}, x_{21}, x_{22} \in \{0, 1\} \tag{2.9}
$$

- for $x_{11} = 1 - x_{12} - x_{13} - x_{14}$, the objective function becomes max $-6 + 11x_{12} + 8x_{13} + 11x_{14} + 2x_{21} + 3x_{22}$;
- the inequality constraints do not change since the resource coefficient values associated with x_{11} equal zero;
- the first equality constraint disappears since it was used to solve variable x_{11}. The second equality constraint remains unchanged since x_{11} is not present, and
- the variables constraints change to

$$
(1 - x_{12} - x_{13} - x_{14}), x_{12}, x_{13}, x_{14}, x_{21}, x_{22} \in \{0, 1\}
$$

Constraint $(1 - x_{12} - x_{13} - x_{14}) \in \{0, 1\}$ is equivalent to constraint $0 \leq (1 - x_{12} - x_{13} - x_{14}) \leq 1$ when *in* Equation 2.9. The lower limit $1 - x_{12} - x_{13} - x_{14} \geq 0$ expresses the inequality $x_{12} + x_{13} + x_{14} \leq 1$; this inequality should be included with the problem's other inequalities. The upper limit $1 - x_{12} - x_{13} - x_{14} \leq 1$ expresses the inequality $x_{12} + x_{13} + x_{14} \geq 0$; this inequality is redundant since the minimum value of x_{ij} equals zero ($x_{ij} \in \{0, 1\}$).

Thus the problem to solve is expressed by Equation 2.10. The value of the $f(x)$ is given by $f(x) = \tilde{f}(x) - 6$ and x_{11} is given by $x_{11} = 1 - x_{12} - x_{13} - x_{14}$.

$$
\begin{array}{lccccccccccl}
\max \tilde{f}(x) = & 11x_{12} & + & 8x_{13} & + & 11x_{14} & + & 2x_{21} & + & 3x_{22} & & \\
\text{subject to:} & & & & & & & & & & & \\
 & & + & 2x_{13} & + & 3x_{14} & & & & & & \leq 3 \\
 & 2x_{12} & & & & & + & 4x_{21} & & & & \leq 4 \\
 & 3x_{12} & + & 2x_{13} & + & 4x_{14} & & & + & 2x_{22} & & \leq 5 \\
 & x_{12} & + & x_{13} & + & x_{14} & & & & & & \leq 1 \\
 & & & & & & + & x_{21} & + & x_{22} & & = 1 \\
 & x_{12}, x_{13}, x_{14}, x_{21}, x_{22} \in \{0,1\} & & & & & & & & & & (2.10)
\end{array}
$$

The procedure used in the previous example can be described as follows:

Step1 Create an empty penalty set N^-.

Step2 For each subset of options N_i $(i = 1, \ldots, n)$

- if there exists an element d such that $v_{id} < 0$ and $r_{kid} = 0 \ \forall \ k$, then form the subset N_i^+ with the elements of N_i less element d, otherwise form the subset $N_i^+ = N_i$. Add element d to penalty set N^-.
- then

$$
\begin{cases}
\text{if } N_i^+ \neq N_i \text{ then } & \tilde{v}_{ij} = v_{ij} - v_{id} \quad \text{for } j \in N_i^+ \\
\text{if } N_i^+ = N_i \text{ then } & \tilde{v}_{ij} = v_{ij} \quad\quad\quad\ \ \text{for } j \in N_i^+
\end{cases}
$$

Step3 Form the subset indexes $I^- = \{i : \text{ if } N_i^+ = N_i\}$ and $I^+ = \{i : \text{ if } N_i^+ \neq N_i\}$

Generate the residual problem

$$\max \quad \tilde{f}(x) = \sum_{i=1}^{n} \sum_{j \in N_i^+} \tilde{v}_{ij} x_{ij} \tag{2.11}$$

$$\begin{aligned}
\text{subject to: } & \sum_{i=1}^{n} \sum_{j \in N_i^+} r_{kij} x_{ij} \leq b_k, && k = 1, ..., m \\
& \sum_{j \in N_i^+} x_{ij} \leq 1, && i \in I^+ \\
& \sum_{j \in N_i^+} x_{ij} = 1, && i \in I^- \\
& x_{ij} \in \{0, 1\}, && i = 1, ..., n, \quad j \in N_i^+
\end{aligned}$$

where $\tilde{v}_{ij}$, r_{kij}, and b_k are positive integer numbers. If $f(x^*)$ represents the system's utility at optimal allocation x^*, then

$$f(x^*) = \tilde{f}(x^*) + Penalty \tag{2.12}$$

where $Penalty = \sum_{N^-} v_{ij}$; that is, $Penalty$ equals the sum of all the coefficients of the original variables that do not appear in the residual problem.

Chapter 3

A New Heuristic for Solving the MMKP

Heuristics are educated guesses that limit the search for solutions in difficult domains. Different heuristics will demand different computational resources in order to operate. They are also likely to generate different solutions for an instance of a given problem. Although a number of heuristics for the solution of the Multichoice Multidimensional Knapsack Problem (MMKP) have been proposed [43, 32, 31], increasing the number of good heuristics is always beneficial. From these, we can select the one that satisfies a particular criteria, such as speed in constructing a solution, closeness of the solution to predefined bounds, rate of failure, and so on. To that end, a new heuristic for solving the MMKP is presented in this chapter. The proposed heuristic is described below.

The MMKP is first relaxed into a Multidimensional Knapsack Problem (MKP) and then into a linear programming problem. The linear problem is solved and from the solution a series of pseudo-utility values is computed for the variables. The pseudo-utility values and the constraints' coefficients are used in order to obtain a feasible solution[1] for the MMKP. Finally, the quality of the feasible solution is improved using the pseudo-utility values and the coefficient values of the objective function. Numerical results show that the performance of this approach is superior to that of earlier techniques. The work presented in this chapter is also described in [47].

[1] A feasible solution is one that does not violate the set of constraints

3.1 Introduction

The MMKP is a combinatorial optimization problem. A set of variables is divided into subsets or groups. Each variable has a value associated with it, as well as weights or volumes, henceforth called resources, for the different dimensions, known as constraints. The MMKP can be expressed as

$$\max \sum_{i=1}^{n} \sum_{j=1}^{l_i} v_{ij} x_{ij} \tag{3.1a}$$

$$\text{subject to: } \sum_{i=1}^{n} \sum_{j=1}^{l_i} r_{kij} x_{ij} \leq b_k, \qquad k = 1, ..., m \tag{3.1b}$$

$$\sum_{j=1}^{l_i} x_{ij} = 1, \qquad i = 1, ..., n \tag{3.1c}$$

$$x_{ij} \in \{0, 1\} \tag{3.1d}$$

where v_{ij}, r_{kij} represent non-negative numbers.

In the MMKP, there are a total of $\sum_{i=1}^{n} l_i$ variables, which are divided into n groups. There are m inequality constraints and n equality constraints that must be satisfied. The MMKP is to select only *one* variable from each group (Equation 3.1c). The value of the variable, x_{ij}, is denoted by v_{ij}, and the resources are denoted by r_{kij}. The goal is to maximize the total value of the selection (Equation 3.1a), subject to resource constraints (Equations 3.1b−3.1d).

It is well known that *single* knapsack problems, such as the

- $0-1$ knapsack given by
  $\max \sum_{j=1}^{l} v_j x_j \quad \text{subject to: } \sum_{j=1}^{l} r_j x_j \leq b, \quad x_j \in \{0, 1\},$
- subset-sum, a particular case of the $0-1$ knapsack that appears when $v_j = r_j$, i.e.,
  $\max \sum_{j=1}^{l} r_j x_j \quad \text{subject to: } \sum_{j=1}^{l} r_j x_j \leq b, \quad x_j \in \{0, 1\},$

- bounded knapsack described by
  $$\max \sum_{j=1}^{l} v_j x_j \quad \text{subject to:} \sum_{j=1}^{l} r_j x_j \leq b, \quad 0 \leq x_j \leq c_j, \quad x_j \text{ integer},$$
- change-making given by
  $$\max \sum_{j=1}^{l} x_j \quad \text{subject to:} \sum_{j=1}^{l} r_j x_j = b, \quad 0 \leq x_j \leq c_j, \quad x_j \text{ integer},$$

and other single knapsack problems are NP-hard [22, 41]. However, because they are not strongly NP-hard, they can be solved in pseudo-polynomial time by dynamic programming (see [15, 34, 41]). Polynomial time and fully polynomial time approximation schemes have also been obtained for the single knapsack problems (see [54, 28] for approximation schemes for the $0-1$ knapsack problem).

On the other hand, for the *multiple* problems such as the

- $0-1$ multiple knapsack given by
  $$\max \sum_{i=1}^{m} \sum_{j=1}^{l} v_j x_{ij} \quad \text{subject to:} \sum_{j=1}^{l} r_j x_{ij} \leq b_i, \quad \sum_{i=1}^{m} x_{ij} \leq 1, \quad x_{ij} \in \{0,1\},$$
- $0-1$ multidimensional knapsack described by
  $$\max \sum_{j=1}^{l} v_j x_j \quad \text{subject to:} \sum_{j=1}^{l} r_{kj} x_j \leq b_k \ (k = 1, \ldots, m,), \quad x_j \in \{0,1\},$$
- bin packing expressed by
  $$\max \sum_{i=1}^{m} y_i \quad \text{subject to:} \sum_{j=1}^{l} r_j x_{ij} \leq b(1 - y_i), \quad \sum_{i=1}^{m} x_{ij} = 1, \quad y_i, x_{ij} \in \{0,1\},$$

and other multiple problems, no pseudo-polynomial algorithm can exist because the multiple problems are NP-hard in the strong sense [21, 41, 52] (dynamic programming will lead to strictly exponential computational times). Also, we can not expect to find fully polynomial approximation schemes because this would imply the existence of a pseudo-polynomial algorithm for the optimal solution of the multiple problems.

As a multiple problem, the MMKP is NP-hard in the strong sense. This can be proved by showing that the MMKP contains a NP-hard problem as a subproblem. In this case, the MMKP contains the $0-1$ multidimensional knapsack problem as a subproblem, which can

be seen as follows [43]:

- divide the total number of variables, say q, into q groups, placing one variable in each group.
- add to each group a Null variable with both a value v_{ij} and resources r_{kij} equal to 0.

If one selects the Null variable, in the $0-1$ multidimensional knapsack problem it corresponds to not placing the variable in the knapsack. If one selects the non-Null variable, it corresponds to placing the variable in the knapsack. This argument proves that the MMKP is NP-hard in the strong sense, and it can be expected that any general algorithm that solves it exactly, consumes computational resources exponentially with increasing problem size q. It is possible that a reasonable amount of computational effort may not produce an optimal solution; therefore, heuristic approaches are needed for solving problems where optimal solution procedures are not useful.

3.1.1 Heuristic Techniques

A heuristic is a rule of thumb, or educated guess, that limits the search for solutions in difficult domains. Heuristics do not guarantee optimal, or even feasible, solutions. Different rules will generate different solutions for an instance of a given problem. Different heuristics will demand different computational resources in order to operate. Heuristic approaches for the solution of the MMKP have been proposed in [43, 32, 31], in [32] a Branch and Bound algorithm for the MMKP is presented as well. Having a number of heuristics at our disposal, allows us to select the one that satisfies our criteria, or constraints such as the time it takes to reach a solution, "closeness" to optimality, rate of failure, and so on. Therefore, a wide selection of heuristics is desirable, if we are to find the one that satisfies our criteria. In this chapter, a new heuristic approach for solving the MMKP is presented. Numerical results show that the solution quality is, by in large, better than those obtained by previous techniques.

As pointed out in [43], for the $0-1$ knapsack problem and the $0-1$ multidimensional knapsack problem, a non-trivial feasible solution can be found, if one exists, by selecting only a single variable. For the multichoice knapsack problem, a feasible solution, if one exists, can be obtained by choosing the lightest variable from each group. However, the combination of more than one constraint (inequalities) and group constraints (equalities) requires the testing of variable combinations. In the worst case, all possible variable combinations must be tested in order to find a feasible solution; in this extreme instance, finding a feasible solution is equivalent to solving the MMKP.

It is important to point out that even if a feasible solution does exist for an instance of the MMKP, the use of any of the heuristics in the above-mentioned works, and the one presented in this work, do not guarantee finding a feasible solution. However, it is desirable to minimize the frequency of failure in finding a feasible solution, if one exits, when a heuristic is used. The computational results show that the rate of failure in finding a feasible solution is lower for the approach presented here than the rate obtained by previous techniques.

This chapter is organized as follows. In section 3.2, the MMKP is relaxed to a Multidimensional Knapsack Problem (MKP), and the reason for the relaxation is stated in that section. In section 3.3, the heuristic for solving the MMKP is presented. A computational complexity analysis of the proposed heuristic is given in section 3.4. The performance of the heuristic is compared with those produced by approaches proposed in [43] and [31], and the results obtained are reported in section 3.5. The performance is reported in terms of solution quality (how close we are to the optimal point) and execution time needed for the heuristics. Finally, conclusions are given in section 3.6.

3.2 The MMKP and MKP

As mentioned, a heuristic is a rule of thumb. The heuristics that have been proposed for the MMKP are based on a variety of rules. For instance, the heuristic in [43] relates to the MKP heuristic in [40], which is a dual heuristic. A dual heuristic starts with the all-ones solution, then variables are set to zero, according to predefined rules, until a feasible solution is obtained. A later step tries to improve this feasible solution. On the other hand, the heuristic in [32] uses rules related to the heuristic in [56], which is a primal heuristic for the zero-one programming problem that includes, but is not limited to, the MKP. A primal heuristic starts with a zero solution, then variables are set to one, according to predefined rules, as long as the solution remains feasible. Later, steps are followed to improve the feasible solution.

Naturally, heuristics for the MMKP have evolved from ideas previously established for the solution of related problems. The heuristic presented in this chapter relates to the heuristic in [51], which is a primal heuristic for the MKP. We first performed the steps necessary to cast the MMKP as an MKP in order to use ideas presented in [51]; then, we performed the steps needed in order to solve the MMKP.

The use of the technique in [51] as a foundation for our work is based on the fact that extensive computational experimentation carried out in [13] shows that the approach in [51] has a superior performance compared to others, including heuristics based on [40] and [56]. Therefore, we postulated that the extension of that approach to solve the MMKP would generate a heuristic with a performance superior to those based on the other cited rules. Our expectations were validated when the computational results of our approach were compared with those of previous techniques.

Expressing the MMKP as an MKP is accomplished by choosing indices and relaxing the equality constraints as shown below. This resulting MKP is a relaxation of the MMKP. The MKP solution serves as a basis for constructing the MMKP solution. In the following

steps, we assume that groups of variables are numbered sequentially and that variables within a group are as well.

- Map into a single index named q, the indices i and j that are used to denote the group number and the variable number inside a group. To that end, define

$$q = L_n, \quad \text{and } L_h = \sum_{i=1}^{h} l_i, \quad \text{and } L_0 = 0$$

and use them to renumber and rename variables and coefficients. For example, let us assume that we have two groups of variables containing three and two variables respectively: x_{11}, x_{12}, x_{13}, x_{21} and x_{21} (i.e., $l_1 = 3$ and $l_2 = 2$). These variables will be renumbered and renamed as: x_1, x_2, x_3, x_4 and x_5 respectively. That is, variable x_{ab} is renumbered and renamed as variable x_z, where $z = L_{a-1} + b$. Likewise, coefficients r_{kab} are renumbered and renamed as coefficients r_{kz}.

- Express the coefficients of all the constraints in the same fashion. That is,
  - the coefficients for the equality constraints should be named explicitly, and
  - the equality constraints themselves should be numbered sequentially, following the inequality constraints.

  Thus, $r_{(h+m)z}$ denotes the coefficients used in the equality constraints. If, in the h-th equality constraint, the new variable x_z belongs to the h-th group, then the coefficient $r_{(h+m)z}$ equals one; otherwise, the coefficient equals zero.

The MMKP is thus expressed as

$$\max \sum_{z=1}^{q} v_z x_z \tag{3.2a}$$

$$\text{subject to:} \quad \sum_{z=1}^{q} r_{kz} x_z \leq b_k, \qquad k = 1, ..., m \tag{3.2b}$$

$$\sum_{z=1}^{q} r_{(h+m)z} x_z = 1, \qquad h = 1, ..., n \tag{3.2c}$$

$$x_z \in \{0, 1\} \tag{3.2d}$$

If a relaxation of Equation 3.2c is performed, that is,

$$\sum_{z=1}^{q} r_{(h+m)z}x_z \leq 1, \qquad h = 1, ..., n \tag{3.3}$$

and if a single index is used to denote the inequality constraints, then we obtained

$$\max \sum_{z=1}^{q} v_z x_z \tag{3.4a}$$

$$\text{subject to:} \quad \sum_{z=1}^{q} r_{kz}x_z \leq b_k, \qquad k = 1, ..., m+n \tag{3.4b}$$

$$x_z \in \{0, 1\} \tag{3.4c}$$

The MMKP has been relaxed to a MKP; due to the relaxation of the equality constraints, the original number of inequality constraints k has increased from m to $m + n$. Each one of the b_k coefficients for $k \geq m$ equals one (the value of the right-hand side coefficient of Equation 3.3).

There are exact and heuristic approaches for solving the MKP; some of them are given in [10], [23] and [51]. A helpful review of exact and heuristic approaches for the MKP is given in [13].

3.3 A New Heuristic for the MMKP (Hmmkp)

A new heuristic for solving the MMKP is presented in this section. The heuristic is called Hmmkp and is related to the heuristic given in [51] (MKHEUR). MKHEUR is a heuristic for solving the MKP. Computational results of heuristics for the MKP presented, or evaluated, in [10, 13, 23, 38, 40, 51] and our own computational experience indicate that MKHEUR offers one of the best trade-offs between computational complexity and performance, and as such was used as a basis for developing Hmmkp. A brief explanation of the MKHEUR heuristic is given, and then the Hmmkp heuristic is presented.

3.3.1 The MKHEUR Background

The technique for the MKP described in this section is based on the use of *pseudo-utility values* (denoted by u_j) for solving the $0-1$ knapsack problem. The values are ratios of the objective function coefficients v_j to the constraint coefficients r_j, that is, $u_j = v_j/r_j$. The greater the ratio, the higher the probability that the corresponding variable will be equal to one in the solution. A high ratio means that either the v_j value of such a variable is high, or that the amount of r_j resource demanded by that variable is low. That is, we try to select high-value low-resource-consumption variables. However, it is not clear how these ratios can be computed in the presence of multiple resource constraints. Several approaches have been proposed to compute the pseudo-values for the MKP, including [51, 38, 56].

In [51], the MKP is relaxed into a "$0-1$ knapsack problem". The relaxation used is a surrogate that allows for a reduction in the hardness, or severity of a problem, and is performed for the purpose of discarding some information, expanding the set of feasible solutions, and thus obtaining a less difficult problem to solve.

The relaxation of the MKP into a single knapsack problem eliminates the presence of multiple resource constraints and, therefore, allows the use of the notion of pseudo values, as in the case of the $0-1$ knapsack problem.

The surrogate problem agglomerates the constraints into a single one named the surrogate constraint; that is, the surrogate constraint is a linear combination of the component constraints that associates a multiplier w_k with each component constraint. The surrogate problem of the MKP is defined by

$$\max \sum_{z=1}^{q} v_z x_z \tag{3.5a}$$

$$\text{subject to: } \sum_{z=1}^{q} \left(\sum_{k} w_k r_{kz} \right) x_z \leq \sum_{k} w_k b_k \tag{3.5b}$$

$$x_z \in \{0, 1\} \tag{3.5c}$$

where:

w_k non-negative multipliers (surrogate multipliers)

As in the $0-1$ knapsack problem, the pseudo-utility values for the surrogate relaxation of the MKP are ratios of the objective function coefficients to the constraint coefficients, that is

$$u_z = \frac{v_z}{\sum_k w_k r_{kz}} \tag{3.6}$$

Not just any value for the surrogate multipliers is allowed. There are surrogate conditions that must be satisfied (such as the fact that the surrogate multipliers should be non-negatives and that solution x must be optimal for the surrogate problem).

The effectiveness of the heuristic, based on the surrogate problem, depends on the ability of the surrogate constraint (Equation 3.5b) to capture the aggregate consumption levels of resources. Finding the solution to the surrogate problem is a difficult task because the feasible region of the problem is not convex. In [51], a method to compute the surrogate multipliers is outlined, and as well an alternative method to find the multipliers is suggested. In this alternative method, the Lagrange multipliers from the solution of the Linear Programming Relaxation of the MKP (LPR-MKP) are used as the surrogate multipliers. In the LPR-MKP, the x_z variables are continuous (i.e., $0 \leq x_z \leq 1$). In [51], it was observed that there is no significant difference in performance when either multipliers are used.

In contrast to the surrogate relaxation, which replaces the original constraints by a new single constraint, the Lagrange relaxation (also known as the Lagrangian relaxation) absorbs the constraints into the objective function. That is, the Lagrange relaxation of the LPR-MKP is given by

$$\max \sum_{z=1}^{q} v_z x_z + \sum_{k=1}^{m} \lambda_k (b_k - \sum_{z=1}^{q} r_{kz} x_z) \tag{3.7a}$$

$$\text{subject to:} \quad 0 \leq x_z \leq 1 \tag{3.7b}$$

where:

λ_k non-negative multipliers (Lagrange multipliers)

We can observe that a multiplier λ_i represents the value of the partial derivative of $\sum_{z=1}^{q} v_z x_z + \sum_{k=1}^{m} \lambda_k (b_k - \sum_{z=1}^{q} r_{kz} x_z)$ with respect to the constraint $(b_i - \sum_{z=1}^{q} r_{iz} x_z)$. That is, the multiplier represents the rate at which we could increase the total value of the selected variables, if we were to raise the target of constraint i. In other words, the multiplier would represent the amount by which profit would be rise if one were allowed one more unit of resource i. To efficiently use the available resources, one might want to select variables according to ratios of their value to their weighted consumption of resources (weighted by the Lagrange multipliers). That is, one might prefer the selection of high-value low-resources-consumption variables.

The advantage of using a Lagrange relaxation over a surrogate one, is the availability of powerful linear programming methods that solve for the Lagrange multipliers. The purpose of this work is not to focus on how multipliers are computed, but rather their use and evaluation as a tool for the solution of the MMKP. Methods to obtain the Lagrange multipliers include the solution of the dual problem[2] per se and the use of methods (such as primal-dual interior-point), where both the primal and the dual problems are an integral part and solved simultaneously [53, 61, 9].

In this work, as a part of a heuristic for the MMKP, a primal-dual linear programming algorithm is used to solve for $\mathbf{x}$ and λ simultaneously; the Lagrange multipliers are then used as the surrogate multipliers, as suggested in [51].

The MKHEUR procedure is as follows.

[2] In optimization, the concept of duality refers to a problem transformation. The original problem, referred to as the primal problem, is transformed into a problem, called the dual problem, in which the parameters are the Lagrange multipliers of the primal.

1. Determine the set of multipliers.
2. Calculate the pseudo-utility values. Sort and renumber variables according to the decreasing order of these values.
3. On the MKP, set variables equal to one, according to the order determined in Step 2. If setting a variable equal to one causes the violation of one of the constraints, set that variable to zero and continue. Denote the feasible solution determined in this step as X_0.
4. For each variable set equal to one in X_0, set the variable equal to zero and repeat Step 3 to define a new feasible solution. Denote these feasible solutions as X_p.

As pointed out in [51], the first solution obtained in Step 3 (X_0) is generally not optimal. This observation, and the observation that optimal solutions differ from X_0 by only a few variables, led to Step 4. This last step attempts to capture optimal solutions by forcing variables which have values one in X_0, to zero. The feasible solution $X_i = [x_1^i, ..., x_q^i]$ $(i = 0, ..., P)$ that

$$\max_i \{\sum_{z=1}^{q} v_z x_z^i\} \tag{3.8}$$

is *the solution* under MKHEUR.

3.3.2 The Hmmkp

The idea of the MHEUR approach was extended so that the group of equality constraints is satisfied as well. This new heuristic is called Hmmkp (Heuristic for the MMKP). In the Hmmkp heuristic, relaxations to the MMKP are performed, the relaxed problem is solved and the Lagrange multipliers are obtained; the multipliers are then used to compute the pseudo-utility values needed for the Hmmkp.

The MMKP relaxation into an MKP was shown in section 3.2. After the relaxation, a LPR of this MKP is performed; that is, $x_z \in \{0, 1\}$ becomes $0 \leq x_{ij} \leq 1$. The LPR-MKP

is solved and the Lagrange multipliers obtained. Pseudo-values are then computed and used in the Hmmkp.

For convenience, we again write the relaxed MMKP formulation,

$$\max \sum_{z=1}^{q} v_z x_z \tag{3.9a}$$

$$\text{subject to:} \quad \sum_{z=1}^{q} r_{kz} x_z \leq b_k, \qquad k = 1, ..., m+n \tag{3.9b}$$

$$x_z \in \{0, 1\} \tag{3.9c}$$

where:

$$q = \sum_{i=1}^{n} l_i$$

We need to remember that, although a single index is used, variables are still divided into groups. Variables x_z, $z = 1 + L_{h-1}, ..., L_h$, belong to group h ($L_h = \sum_{i=1}^{h} l_i$ and $L_0 = 0$ have been defined in Section 3.2).

The proposed Hmmkp heuristic works as follows. First, an LPR of the relaxed MMKP described by Equations 3.9a−3.9c is solved and the Lagrange multipliers obtained. Then a series of steps are performed in order to obtain, if possible, a solution for the MMKP. A brief description of these steps follows. Multipliers are used to calculate the pseudo-utility values (Step 1). Based on the pseudo-utility values, the approach creates a first solution (Step 2). The heuristic selects one variable per group; the variable selected is the one with the highest pseudo-utility value. Then a series of solutions derived from the first one are generated (Step 3). The feasible solution, if any, is improved, if possible, in Steps 6−7. If no feasible solution is found in Step 3, then the heuristic tries to create one using the Euclidean norms of the ratio of coefficients *r's* and coefficients *b's*. The Euclidean norm for each variable, called the resource coefficient value, is calculated in Step 1. Steps 4 and 5 attempt to create a feasible solution based on the resource value coefficients. First, Step 4, the variable with the lowest resource value coefficient in each group, is selected. If the

solution is feasible, then Steps 6 and 7 are performed. If the solution is not feasible, then Step 5 follows. In Step 5, a reduction in the number of violated constraints is attempted.

After step 3, feasible solutions, if any, are of good quality. However, when failing to find a feasible solution, one needs to concentrate on the immediate problem of finding one. The problem of *finding a feasible solution* is not the same as the problem of *finding a feasible solution of good quality*. In the former, information about the actual value or pseudo value of the variables to be selected is irrelevant; what matters is the amount of resources the variables consume and the amount of resources available. That is why, in the Hmmkp, steps that concentrate only on coefficients *r's* and coefficients *b's* are used when failing to find a feasible solution using the pseudo-value approach. Different rules on how to relate the mentioned coefficients were evaluated. It was observed in the problems tested that the Euclidean norm is an efficient tool in constructing a feasible solution. However, as expected, such a solution is not of a good quality and requires improvements, if possible. Steps for improvement are also provided in the Hmmkp. A detailed description of the Hmmpk follows below.

The Hmmkp procedure:

1. Index Generation.
    - Relax the MMKP into an MKP, Equations 3.9a−3.9c. Solve the LPR of this relaxed problem and obtain the Lagrange multipliers related to the inequality constraints of the LPR problem. In the MMKP, there are n groups of variables and q total variables. There are also m original inequality constraints (OICs) and n new inequality constraints because of the MMKP relaxation. The coefficients of the inequality constraints, r_{kz}, form a matrix of dimension $(m+n)q$. For each variable:
        - Calculate the pseudo-utility value

$$u_z = \frac{v_z}{\sum_{k=1}^{m+n} \lambda_k r_{kz}} \tag{3.10}$$

The coefficients λ_k are the Lagrange multipliers obtained.

- Calculate the resource value coefficient, res_z, by

$$res_z = \sqrt{\sum_{k=1}^{m} (r_{kz}/b_k)^2} \tag{3.11}$$

- Generate a utility-index table containing the order of the variables (per group) according to the pseudo-utility values obtained (from highest to lowest utility value). Then, reorder the groups in the utility-index table, according to the pseudo-utility values of the variables with the highest pseudo-utility value in each group (from highest to lowest pseudo-utility value of the variables with the highest pseudo-utility value in each group).
- Generate a resource-index table containing the order of the variables (per group) according to the resource value coefficients obtained (from lowest to highest resource value). Then, reorder the groups in the resource-index table, according to the resource values of the variables with the lowest resource value in each group (from lowest to highest resource value of the variables with the lowest resource value in each group).

The goal of the following steps is the construction of a feasible solution for the original MMKP. Although variables used a single index, they are still divided into groups. Variables x_z, $z = 1 + L_{h-1}, ..., L_h$, belong to group h.

2. Set variables equal to one, according to the order determined in the utility-index table. From the first group, set equal to one the variable with the highest pseudo-utility value. Next, set equal to zero the rest of the variables belonging to that same group. From the second group, set equal to one the variable with the highest pseudo-utility value and, again, set to zero the remaining variables belonging to the second

group. If setting a variable equal to one causes the violation of one of the OICs, set that variable to zero and explore whether the variable (from the same group) with the next highest pseudo-utility value can be set to one. Continue until all groups have been evaluated or until no variable (from the ones still available) can be selected because it violates an OIC. Denote the solution obtained in this step as X_0. At this step, the solution can be either a feasible or an infeasible solution.

3. For each group of variables and in reference to solution X_0, as obtained at the end of Step 2:
   - If the group contains a variable that is equal to one, set this variable to zero. This variable will not be allowed to participate in the following selection process.
     - Set to zero all variables in all other groups.
     - Use the procedure outlined in Step 2 to obtain a new solution (feasible or infeasible). Name this solution $X_p = [x_1^p, ..., x_q^p]$.
   - If the group does not contain a variable that is equal to one, do nothing.

   At the end of Step 3, we have a collection of solutions $X_p; p = 0, 1, ..., P$ where P is the number of groups of variables, each group containing a variable set to one in the original solution X_0. Some, or all, of the solutions maybe infeasible (that is, a solution contains at least one group of variables all of which are zero)
   If no feasible solutions exist in the collection of $P+1$ solutions identified previously, continue with Step 4. If there are feasible solutions, name as $\tilde{X}$ the feasible solution that maximizes the objective function, that is, $\tilde{X} = X_{p^*}$, where

$$p^* = \max_p \{\sum_{z=1}^{q} v_z x_z^p\} \tag{3.12}$$

   Go to Step 7.

4. Try to create a feasible solution using the resource-index table. The variables preferred are the ones with the lowest resource value coefficient in each group. For example, from the first group, set equal to one the variable with the lowest resource

value coefficient, and then set to zero the rest of the variables that belong to that group. Do the same for the remaining groups. If setting a variable equal to one causes the violation of one of the OICs, set that variable to zero and explore the variable with the next lowest resource value coefficient in the group.

If at any time, the solution being created becomes infeasible (all the variables in a group have been set to zero), then STOP and go to Step 5. If a feasible solution is found, denote that solution as $\tilde{X}$ and go to Step 6.

5. Using the resource-index table, in *every* group set to one the variable with the lowest resource value coefficient and denote the solution as X_{old}.

   Calculate the amount of resources

$$used_k^{old} = \sum_i r_{ki}, \qquad k = 1, ..., m$$

   (i represents the selected variables) consumed by X_{old}. Identify the set of violated OICs V_{old} (the set holds the indices j of the OICs that are violated by solution X_{old}). Then, estimate the average normalized resource-usage coefficient

$$anruc_{old} = \frac{\sum_{j \in V_{old}} (used_j^{old}/b_j)}{\tau}$$

   where:

   - τ represents the total number of OICs violated
   - j represents the indices of violated OICs

   Using the resource-index table, explore whether it is possible to reduce $anruc_{old}$. In group one, set the variable that equals one to zero, and then set to one any of the variables in the same group. Denote this solution as X_{new}. Evaluate the resources used, $used_k^{new}$, by X_{new} and identify a new set, V_{new}, of violated OICs. If V_{new} is contained in V_{old}, then calculate $anruc_{new}$. The point is not to violate any non-violated OIC. If $anruc_{new} < anruc_{old}$, then $X_{old} = X_{new}$, $anruc_{new} = anruc_{old}$

and $V_{old} = V_{new}$. Perform this exploration for every variable in the first group. Then, according to the resource-index table, repeat the process with each of the other groups of variables. This is an iterative process. An iteration is performed when all groups have been evaluated. A maximum number of l iterations is allowed to be performed in this step.

The idea is to empty the set of violated OICs through the reduction of the average normalized resource-usage coefficient. If at any time all OIC are non-violated, then stop Step 5, denote the feasible solution, X_{old}, as $\tilde{X}$ and go to Step 6. On the other hand, if either the maximum number of iterations, l, has been reached or an iteration has been performed and no reduction in $anruc_{old}$ is achieved, then STOP and end the Hmmkp heuristic. There is no feasible solution under Hmmkp.

6. Improvement of the feasible solution by pseudo-utility.

   Perform the following iterative procedure.

   - Iterative procedure begins: Iterate from $t = 1$ to a maximum number, that equals the number of variables in the largest group, minus one.
     - For each group, calculate the improvement factor

       $$imp_i^t = h_i^t - s_{iz} \quad (i = 1, ..., n)$$

       (h_i^t is the t^{th} highest pseudo-utility value in group i; for example, in iteration $t = 2$, h_i^2 is the second highest pseudo-utility value in group i. s_{iz} is the pseudo-utility value of the variable z that is set to one in group i in the actual feasible solution $\tilde{X}$).
     - Select the group with the highest imp_i^t and explore if, while maintaining feasibility, an improvement is possible. For example, set to zero the variable that equals one in group i, and set to one the variable with the h_i^t value. If an improvement is possible, update $\tilde{X}$. Proceed to the group with the second highest improvement factor, and so on.

       Note: Groups with $imp_i^t \leq 0$ do not need to be explored. This situation

occurs when the pseudo-utility value of the variable set to one in group i has a higher value than h_i^t, which is the t^{th} highest pseudo-utility value in group i. That is, there is no possible improvement in group i but only a reduction in its pseudo-value, which is not the goal.

- Iterative procedure ends.

The goal of the iterative procedure is to improve, as many times as possible, the feasible solution $\tilde{X}$. As mentioned, if $imp_i^t \leq 0$, the i group does not need to be evaluated in subsequent iterations. The maximum number of iterations equals the number of variables in the biggest group, minus one.

7. Improvement of feasible solution by v_z coefficients.
   Perform the iterative procedure described in Step 6, but now h_i^1 is the highest coefficient v_z in the group i, h_i^2 is the second highest coefficient v_z in group i, and so on. In group i, s_{iz} is the v_z coefficient of the variable set to one in the feasible solution $\tilde{X}$.
   Then return $\tilde{X}$; this is the best solution found under Hmmkp.
   End of the Hmmkp procedure.

3.4 Computational Complexity

Not taking into account the computational complexity of the linear programming algorithm, the complexity of Hmmkp is computed as follows. For simplicity, let us assume that the number of variables per group is a constant l [3], and that there are m resources and n groups. At each step of the Hmmkp heuristic the complexity is:

- In Step 1, a utility-index table and a resource-index table are generated. For the

[3] If the number of variables in the groups is not constant, then l equals the number of elements in the biggest group. Computational Complexity is related to the efficiency of algorithms, a measure of this efficiency is the number of computer operations it takes to solve the problem in the worst case. This worst case is considered by making l equal to the number of elements in the biggest group.

utility-index table, the pseudo-utility values from Equation 3.10 are calculated; this operation has a computational complexity of $\mathcal{O}(ln(m+n))$. Next, variables are sorted in each group, $\mathcal{O}(ln \log l)$, and then the groups are sorted, $\mathcal{O}(n \log n)$. For the resource-index table, the resource value coefficients (Equation 3.11) are computed; this operation has complexity of $\mathcal{O}(lnm)$. Next, the variables are sorted in each group, $\mathcal{O}(ln \log l)$, and then the groups are sorted as well, $\mathcal{O}(n \log n)$. Thus, the computational complexity of Step 1 is $\mathcal{O}(\max\{ln(m+n), ln \log l\})$.

- The computational complexity of Step 2 equals $\mathcal{O}(lmn)$, which means it is possible that every variable in every group may need to be evaluated in order to create a solution (with up to n variables), whether or not the solution is feasible.
- In Step 3, each of the variables, up to n, in the solution from Step 2 is set to zero individually. The remaining $(ln-1)$ variables are explored when a call is made to Step 2. Therefore, Step 3 has a computational complexity equal to $\mathcal{O}(mn(ln-1))$.
- The computational complexity of Step 4 equals $\mathcal{O}(lmn)$. This step is similar to Step 2, that is, it is possible that all the ln variables may need to be tested in order to try to create a feasible solution, whether the final solution in the step is feasible or not.
- In Step 5, variables are selected using the resource-index table, and an initial solution is constructed. This operation has a complexity of $\mathcal{O}(n)$. Next, the amount of resources consumed by the solution is computed, $\mathcal{O}(mn)$, the set of violated OICs is identified, $\mathcal{O}(m)$, and the average normalized resource-usage coefficient ($anruc$) is computed, $\mathcal{O}(m)$. Next, the remaining $n(l-1)$ variables are explored during one iteration in order to evaluate whether or not a new solution (obtained by changing a selected variable for one of the remaining variables) will reduce the $anruc$. The maximum number of iterations is set equal to l. Therefore, the complexity of Step 5 is $\mathcal{O}(lmn^2(l-1))$
- In Step 6, the improvement factor imp_i^t is computed for each group, $\mathcal{O}(n)$. The group with the highest improvement factor is selected, $\mathcal{O}(n)$, a new solution is then

constructed $\mathcal{O}(mn)$, and feasibility is evaluated $\mathcal{O}(m)$. Next, the group with the second highest improvement factor is selected, and so on. There are n groups to be evaluated and the maximum number of iterations to be performed equals $(l-1)$. Therefore, Step 6 has a computational complexity of $\mathcal{O}(mn^2(l-1))$.

- Step 7 has the same complexity as Step 6.

The complexity of Steps 1–7 is $\mathcal{O}(mn^2(l^2-l))$ (in the MMKP $m, n, l > 1$). However, 95% of the time used by Hmmkp is consumed by the linear programming interior-point algorithm that has a complexity of $\mathcal{O}((ln)^{3.5}t)$, where t can be interpreted as the number of significant figures required in the solution, and assuming that in the linear programming problem the total number of variables is equal to, or greater than, the total number of constraints ($ln \geq (m+n)$) [29, 61]. Therefore, the computational complexity of Hmmkp is given by $\mathcal{O}((ln)^{3.5}t)$. It is expected that both heuristics MOSER and HEUR will run faster, since their complexity is given by $\mathcal{O}(mn^2(l-1)^2)$ [43, 31].

3.5 Comparison of Performance

The performance of the Hmmkp was compared to the ones obtained using [43] and [31] (MOSER and HEU). The MOSER approach is related to the heuristic given in [40], but is organized in such a way that the equality constraints of the MMKP are preserved. In [40] and [43], the idea of generalized Lagrange multipliers as presented in [17] is used.

On the other hand, the HEU approach uses an iterative improvement procedure, based on the concepts of savings in aggregate resource and value gain per unit of extra aggregate resource. The concept of aggregate resource is presented in [56]. The heuristic starts searching for a feasible solution. First, HEU tries a solution where, in each group, the variable with the smallest coefficient v_j is selected. If this solution is not feasible, HEU tries an iterative procedure in order to find one. This procedure is similar to Step 5 of Hmmkp, however, in HEU the criterion used (for exchanging variables) is the decrement

in the amount of resources consumed by the most violated constraint. The variable to be exchanged does not violate any non-violated constraint. If a feasible solution is found, then HEU performs a series of upgrades and downgrades to improve the quality of that solution.

The code PCx [14] was used to solve the LPR of the relaxed MMKP. PCx is a primal-dual interior-point code for linear programming and is written in C. The code was extended in order to include the steps needed for solving the MMKP. The free software LIPSOL [62] is also an implementation of a primal-dual interior-point algorithm for large-scale linear programming. However, LIPSOL works under the MATLAB[4] environment; therefore, it does not performs as fast as PCx. We decided to modify PCx in order to include the changes observed in LIPSOL that produced a better performance, specifically, the heuristic for estimating the initial values of the Lagrange multipliers used in [62]. This initial point heuristic is a modified version of the one presented in [39]. Modifications were carried out in the PCs as well, in order to include the heuristic used in the centering parameter estimation step and in the updating step, from [62].

Using the code provided by the author of [31], 24 random problems were generated. The performance in terms of solution quality and execution time is given in Table 3.1. The number of groups, variables per group, and inequality constraints is given by parameters n, l and m, respectively. The column *OPT* shows the optimum value of the problem evaluated. The heuristics were tested in an AMD 1.4 GHz Personal Computer, with 264 MB of RAM, and running Mandrake Linux OS, kernel 2.4.21-0.13mdk.

As can be observed from Table 3.1, Hmmkp is the heuristic that is the the most time consuming because of the solution to the linear programming problem. As mentioned, the linear programming problem solution takes approximately 95% of the time used by Hmmkp. However, the Hmmkp solution quality is better than that obtained using either MOSER or HEU.

[4] MATLAB is a registered trademark of The Math Works,Inc.

Table 3.1. *Performance comparison of MOSER, HEU and Hmmkp*

MMKP parameters			Solution quality				Execution time (msec)		
n	l	m	OPT.	MOSER	HEU	Hmmkp	MOSER	HEU	Hmmkp
5	5	5	108	105	108	105	0	0	2
5	7	5	116	116	108	116	0	0	3
5	9	5	121	121	121	121	0	0	3
10	5	5	232	0	232	232	0.5	0.1	6.6
10	7	5	237	236	235	236	0	0.1	7.1
10	9	5	241	223	238	238	0.5	0.2	8.4
15	5	5	351	328	342	348	0.2	0.2	6.9
15	7	5	338	324	320	338	0.5	0.2	7
15	9	5	386	373	386	384	0.7	0.4	8.4
20	5	5	434	0	423	434	0.6	0.2	7
20	7	5	493	472	466	493	0.9	0.5	8.7
20	9	5	456	412	415	456	1.8	0.7	10.6
25	5	5	528	474	491	505	0.7	0.3	9.2
25	7	5	613	600	613	613	1.6	0.8	9.4
25	9	5	586	520	572	584	2.6	1.1	10.2
30	5	5	667	606	659	667	1.2	0.7	10.4
30	7	5	698	627	677	697	2.4	1.3	10.5
30	9	5	701	564	682	696	3.9	1.5	11
35	5	5	774	693	721	761	1.1	0.7	12
35	7	5	870	797	830	870	3.3	1.3	13.5
35	9	5	896	808	844	894	3.4	2.2	13.8
40	5	5	822	714	778	805	1.7	1.1	12
40	7	5	867	703	777	865	3.7	1.8	12.2
40	9	5	1009	902	962	1004	6.2	3.2	14.8
	$\sum$		12 544	10 718	12 000	12 462	37.5	18.6	217.7
	perf. %		100%	85.4%	95.6%	99.3%			

Hmmkp and the heuristics evaluated here may fail to find a feasible solution (provided there is one). As the amount of available resources (b_k in Equation 3.1b) decreases, the harder it is to solve the MMKP. In order to better test the effectiveness of the heuristics, standard MMKP test problems should be used. Insofar as we are aware, there is no such set of test problems. Nevertheless, publicly-available standard MKP test problems exist (see OR-Library at www.ms.ic.ac.uk/info.html). In order to create an MMKP from an MKP, we only need to define the number of i-groups and the number of l_i-variables per group. This definition creates the set of equality constraints (Equation 3.1c). In this work, the number of variables in each group equals five. The first five variables of an MKP form the first group, the next five variables form the second, and so on.

The OR-Library test problems we evaluated are contained in the files mknapcb7, mknapcb8, and mknapcb9. The first problems in each file are the most difficult to solve, and they were used for testing MOSER, HEU and Hmmkp. Only the first ten problems from each of the above-mentioned files were tested.

The available resources, coefficients b_k, were decreased by a factor f. First, no decrement is performed ($f = 1.0$); then, a decrement of 10% ($f = 0.9$) is allowed; finally, a decrement is done, so that at least one of the heuristics fails in finding a feasible solution. Because of the decrement factor f used, a total of 90 problems were evaluated. The results are given in Appendix A through Tables A.1−A.3.

The reported results are representative of the performance produced by the heuristics. These results show a similar behaviour as that reported in Table 3.1; that is, Hmmkp produces better solutions and requires a higher computational time. A statistical analysis of the solutions generated by Hmmkp supports that fact that Hmmkp produces better solutions than either MOSER or HEU. This analysis is provided in Appendix A, Section A.1.2. Numerical results also show that Hmmkp is able to find a feasible solution when either MOSER or HEU fail. Out of the 204 problems reported in this work, we did not find a single instance where the Hmmkp heuristic fails and the other heuristics do not. Extensive

computational evaluations show that, although Hmmkp does fail in finding a feasible solution in some instances of the MMKP, its rate of failure is lower than that provided by other techniques. An instance where Hmmkp, HEU and MOSER do fail, and where there exists a feasible solution, occurs when trying to solve problem number nine from file mknapcb7 and when using a decrement factor $f = 0.83$.

Additionally, we have tested the above heuristics on further synthetic MMKP problems derived from the standard MKP test problems as discussed above, but these synthetic problems include groups of variables of varying size. The same behaviour was observed: Hmmkp found better quality solutions than MOSER or HEU. A detailed discussion can be found in Appendix A section A.1.1.

Although Hmmkp has seven steps, not all of them are used. In fact, in more than 90% of the cases (where f equals either 1.0 or 0.9) a feasible solution is found in Step 2 or 3.

3.6 Conclusion

In this chapter, a heuristic called Hmmkp is presented for the solution of the MMKP. The heuristic needs to solve the LPR of a relaxation of the MMKP.

Based on the computational results, Hmmkp is shown to have two important characteristics. First, Hmmkp produces a better solution than at least two other heuristics presented elsewhere in the literature. Second, Hmmkp has the highly desirable quality of being able to find a feasible solution where other heuristics fail; that is, the Hmmkp rate of failure in finding a feasible solution will be lower. However, the use of Hmmkp, or any of the heuristics mentioned in this work, does not guarantee finding a feasible solution, if one exits. Although the time Hmmkp needs is greater than that of the other heuristics, it is small (in the order of milliseconds) and is useful in real time application. We consider this increased computational complexity as well invested because of the overall performance obtained.

Chapter 4

Resource Allocation on a Cellular System

4.1 Introduction

In this chapter, we use a knapsack formulation to express a resource allocation problem on a mobile system. The knapsack problem is relaxed and then solved. From the solution, a series of pseudo-utility values is obtained. These values are used to establish an order by which a series of trials are attempted to evaluate whether mobile stations can be allocated channels. The work presented in this chapter is also described in [48].

4.2 Mobile Systems

A mobile communication system is one that allows mobility in communications by transmitting data via radio waves. There are several examples of mobile communication systems, including the following.

- **Cellular radio networks.** In this system, the communication network is divided into cells; each cell is served by a base station. The communication is duplex; that is, the device used to communicate can send/receive a communication anywhere inside the cellular systems.

- **Communication satellites.** This is a broadcast system. The satellites are broadcast devices, transponders that listen to a particular radio frequency, amplify the signal, and rebroadcast it at a different frequency. Point-to-Point satellite-based voice and data systems are also a possibility (e.g., the Iridium Satellite System).
- **Paging.** This is a one-way communication system. An antenna, or satellite, broadcasts short messages to subscribers. The receivers are usually devices such as beepers, which display messages on a screen.
- **Personal handyphone system.** This is similar to a cellular radio network system; however, the communication devices can also communicate directly with one another when in range. This is an advantage over cellular communication devices, which can only communicate with one another via a base station.

In this chapter, a resource allocation approach on a cellular radio network is proposed. In the following section, we briefly described how a cellular system works.

4.3 Cellular Radio Networks

A cellular radio network is a mobile network on which resources are managed in bounded areas called cells. Each cell is served by an antenna or base station (BS) [16] (see Figure 4.1). Cell size and shape depend on signal strength, the presence of obstacles to signal propagation, customer capacity, and cost constraints. A cellular network consists of land-based and radio-based sections. The network is composed of several elements, including the following:

- **Mobile Station** (MS): A device used to communicate over the cellular network.
- **Base Station** (BS): This is a transmitter/receiver of signals over the radio interface section of the network.
- **Mobile Switching Centre** (MSC): This is the network core, which sets up and maintains calls made over the network.

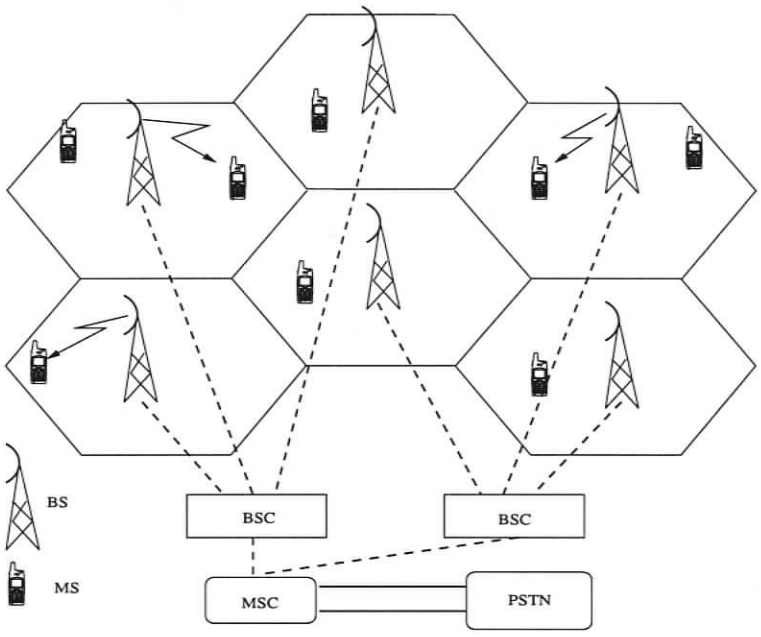

Figure 4.1. *A cellular radio network*

- **Base Station Controller** (BSC): Controls communication between a group of BSs and a single MSC.
- **Public Switched Telephone Network** (PSTN): Collection of interconnected voice-oriented public telephone networks.

When one turns on a mobile station, a registration process takes place. Registration lets the system know that the mobile station is active, where it is located, and that it can now take incoming calls. After registration, when a mobile station places a call, the steps are as follows.

1. The mobile station sends a call initiation request to the nearest base station. This request is sent on a channel called the reverse control channel, and contains the telephone number of the called party and the information needed to register the call on the network.
2. The base station sends the request to the MSC.

3. The MSC validates the request and uses the number of the called party to connect to that party via the PSTN. First, the MSC connects itself to the MSC of the called party, then the MSC instructs the base station, and the mobile station that placed the call, to switch to voice channels.
4. The mobile station that placed the call is then connected to the called station, using voice channels.

A mobile station continuously scans for paging signals from base stations through a channel called the forward control channel. The paging signal informs the mobile station of incoming calls. If a MSC receives a request for a connection to a mobile station in its area, it sends a broadcast message to all base stations under its control. The message contains the number of the mobile station being called. The base stations then broadcast the message on control channels. The correct mobile station acknowledges the page by identifying itself over the reverse control channel. The MSC receives this acknowledgement via the base station, and instructs it and the mobile station to switch to a voice channel. A data message is then transmitted over the voice channel, which instructs the station to ring.

4.3.1 Resource Allocation

The work presented in this chapter relates to Frequency Division Multiple Access (FDMA) and Time Division Multiple Access (TDMA) on cellular radio networks. In a dynamic channel allocation, a channel is eligible for use at any cell if signal interference levels are satisfied. Dynamic channel allocation offers a more efficient use of channels than fixed channel allocation. On the other hand, power control is a mechanism that can be used to regulate signal interference levels. Thus, the use of dynamic channel allocation and power control increases even further the efficiency, or capacity, of a system, rather than using dynamic allocation alone. A helpful review of channel assignment schemes can be found in [30], and a helpful review of the performance and capacity of dynamic channel assignment and power control can be found in [60].

One of the most cited papers on power control in cellular systems is [27], where a fixed channel assignment with channel reused in every other cell is assumed. This work was later extended and presented in [25], where a dynamic channel assignment is assumed. We have used [25] as the foundation for the work presented in this chapter. As mentioned in [25], the capacity offered by resource allocation using power control depends on the spectrum-packing capability of the channel assignment scheme. The increasing demand for mobile communication services calls for an efficient use of limited transmission power and spectrum.

In this work, we evaluate a mechanism used to allocate resources on a cellular radio network that has power control capability. First, we obtain an estimation of the power to be transmitted by every MS waiting to be allocated. This estimation is used to express a knapsack problem, which is then relaxed. Next, the problem is solved, and a series of pseudo-utility values is obtained for each MS. These values are used to establish a sequence by which attempts to allocate channels are carried out.

The system's performance is studied, as in [20, 25], in one randomly chosen instant of time. Frozen in this instant of time, the system is referred to as a snapshot. The resource allocation scheme is then applied to the snapshot and its performance evaluated[1]. The power estimation of the MSs waiting to be allocated is obtained from a "copy" of a slightly modified Distributed Dynamic Resource Allocation (DDRA) mechanism named the "phantom network."

As in [25], the resource allocation mechanism presented in this chapter is for the uplink (MS to BS) only. In the uplink, the transmitter is in the MS and the receiver is at the BS. For the downlink, the analysis is identical, provided the roles of the transmitter and receiver are reversed.

This chapter is organized as follows. Section 4.4 explains the channel assignment and

[1]It is assumed that the resource allocation scheme reaches its final state before the performance is evaluated.

power control schemes. Section 4.5 describes the channel assignment problems using a knapsack formulation. Section 4.6 explains the resource allocation procedure. Section 4.7 provides numerical results. Finally, Section 4.9 provides conclusions.

4.4 Channel, Power and DDRA Schemes Background

The purpose of a channel assignment scheme is to assign radio channels to MSs so that a certain level of Carrier to Interference Ratio (CIR) is maintained. This CIR level can be achieved using power control. The higher the CIR level the better the quality of the communication link. The idea behind a power control scheme is based on the fact that the CIR of a MS at its BS is directly proportional to the power level of the desired signal and inversely proportional to the sum of the power of co-channel interferers[2]. Therefore, by increasing the power of the desired signal and/or decreasing the power level of interfering signals, the CIR level can be achieved. Power control schemes try to reduce the overall CIR on the system by measuring the received power and increasing (or decreasing) the transmitting power, in order to maximize the minimum CIR on the system [30].

A description of the channel assignment, and the power control schemes used, follows (descriptions taken from [25]).

4.4.1 Channel Assignment

The channel assignment scheme is based on minimum interference. The scheme is distributed because it is based on local interference power measurements. When the MS-i (also referred to as a terminal, or simply as a user) requires service, it signals the nearest BS its need for a channel. The BS measures the interfering signal power on its channels that are not in use, and assigns a channel j^* with a link quality, measured by CIR, that

[2]Co-channel interferer: MSs that cause co-channel interference; the interference is mainly due to the fact that the spectrum allocated is being reused multiple times.

satisfies

$$j^* = \underset{j \in C}{\text{argmax}}\{\gamma_{i(j)}\} \tag{4.1}$$

where C represents the set of channels not assigned to other MSs, and $\gamma_{i(j)}$ represents the CIR of the MS-i, as if it were to use channel j.

An MS-i may be denied service in two situations:

1. When all the channels at the BS are already assigned to other MSs.
2. When the MS-i suffers a CIR less than the lowest CIR, γ_f, acceptable to the receiver.

The CIR is measured at the BS. The CIR of the MS-i at its BS, as if it were to use channel-j, is given by

$$\gamma_{i(j)} = \frac{P_{i(j)} G_i}{\sum\limits_{\substack{m \in M \\ m \neq i}} P_{m(j)} G_m + v} \tag{4.2}$$

where:

M	the set that contains the indexes of all the MSs, on the network, using channel-j.
G_i	the gain factor on the desired communication link between the MS-i and its BS.
G_m	the gain factor on the interference links between the MS-m and the BS of MS-i.
v	the noise at the receiver BS of MS-i.
$P_{i(j)}$	the power transmitted by MS-i and using channel-j.

Let us assume that Figure 4.2 describes the following channel allocation. MSs 1 and 2 are attended by BS-A; MS-3 is attended by BS-B; and, MSs 4 and 5 by BS-C. Note that MS 1, 3 and 4 are using channel 1. If the CIR of MS-3 at its BS needs to be computed, then the gain factors and powers of the interfering MSs 1 and 4 need to be considered in that CIR computation.

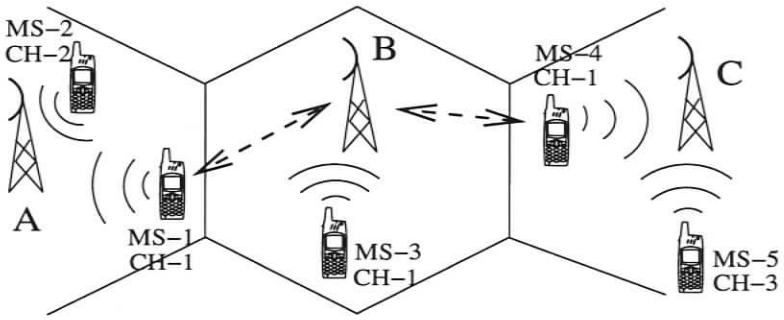

Figure 4.2. *Interference in cellular radio network with power control*

4.4.2 Power Control Scheme

A Distributed Constrained Power Control (DCPC) scheme [25, 26, 27] is used to adjust power. In the DCPC, each M terminal on a given channel adjusts its transmitting power in order to achieve a target CIR denoted by γ_t. A transmitter power is updated using the formula

$$P_i^{(n)} = \min\left\{P_{\max}, \gamma_t \frac{P_i^{(n-1)}}{\gamma_i^{(n-1)}}\right\} \qquad 1 \leq i \leq M, \qquad n \geq 1 \tag{4.3}$$

where:

$P_{\max}$	the maximum transmitting power
$P_i^{(n-1)}$	the transmitting power of the MS-i at the $(n-1)$-th instant of time
$\gamma_i^{(n-1)}$	the CIR of the MS-i at its BS at the $(n-1)$-th instant of time

The probability of assignment failure P_a is the performance measure used. This is the probability that a randomly selected terminal suffers an assignment failure. An estimate of P_a is obtained as an average over K snapshots of the cellular system; the estimate $\hat{P}_a$ is given by

$$\hat{P}_a = \frac{\sum_{i=1}^{K} \text{terminals denied service in snapshot } i}{\sum_{i=1}^{K} \text{terminals that need service in snapshot } i} \tag{4.4}$$

4.4.3 Distributed Dynamic Resource Allocation (DDRA) Scheme

The DDRA is a merge of the channel assignment and power control schemes explained above, and it works as follows:

1. When a user requires service, it signals its need to the nearest base station.
2. The base station assigns a channel that satisfies Equation 4.1.
3. With the knowledge of the interference on the channel and the received signal strength on the control channel, the base station estimates the CIR assuming a unity power in the transmitter. Using this information, the transmitter power necessary to achieve a target CIR is calculated and communicated to the mobile on the control channel, that is,

$$P_i^{(0)} = \min\left\{P_{\max}, \gamma_t \frac{1}{\gamma_i}\right\} \tag{4.5}$$

4. The terminal then joins the rest of the terminals on the same channel in updating powers according to the DCPC scheme.

4.5 Knapsack on a Cellular System

The allocation mechanism described in this section is centralized. It is assumed that there exists a Resource Allocation Centre (RAC) that has information on the MSs' physical location, as well as on the network CIR and transmitting power status.

Let us designate by

- U_k the set denoting the MSs that need to be allocated channels at cell k
- C_k the set denoting the available channels at cell k
- V_k the set denoting the cells that form the vicinity[3] of cell k

An estimation of the transmitting power for each MS waiting for a channel is performed as follows. Assume that the MS-i at cell k requires a channel. If there are available channels at cell k then:

1. A phantom network (simulated system) is reproduced at the RAC. The actual status of cell k and of every cell $z \in V_k$ is reproduced, or copied, (that is, physical location and transmitting power information regarding the MSs in the system) onto the phantom network.
2. For each available channel $j \in C_k$, the following procedure is performed on the phantom network:
   - The MS-i is assigned channel j.
   - If at the vicinity cell $z \in V_k$ there are MSs waiting to be allocated channels, and if channel j is also available at that cell, then an MS is assigned to channel j at that cell. The MS to be assigned at the vicinity cell is the one that will cause the greatest interference to the MS-i on channel j at cell k. This step is performed at every vicinity cell where channel j is available, and where there is the need to allocate a channel to a MS in that vicinity cell.
   - Then, the DCPC mechanism is executed on the phantom network, and a power estimation for the MS-i at cell k is obtained; that is, all terminals using channel j update transmitting power according to the DCPC scheme.

This process is performed for every MS, and for every available channel in its corre-

[3]The vicinity cells of cell z are those located within a predefined distance from cell z.

sponding cell, as well as at every cell that needs to allocate channels.

Note: The interference a MS causes to another MS is only known after the DCPC mechanism has been performed, and final transmitting power values are computed. Therefore, the selection of the most interfering MSs is not possible unless assumptions are made. We assume that MSs waiting for allocation at vicinity cells use a transmitting power of $\bar{P}$, that is, a constant transmitting power (before DCPC is applied). Therefore, the MSs that cause the greatest interference are the ones with the highest gain factors on the interference links to BS-k (Equation 4.2). Since the gain of a link relates to distance, then the MSs closest to BS-k are selected at the vicinity cells. For instance, if from MS 1 and 2 we need to select the one that causes the greatest interference to MS 3 at cell B, then we select MS 1 at cell A because MS 1 is closer to BS B than MS 2 (Figure 4.3). For the same reason, we select MS 4 instead of MS 5 at cell C.

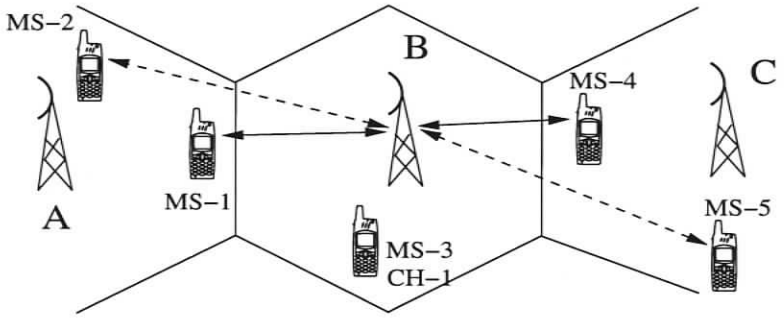

Figure 4.3. *Mobile station selection in phantom network*

It is clear that there will be several power estimations for every terminal that requires a channel. If there are r terminals at cell k waiting for channels, and if there are w available channels at cell k, then there will be, at most, rw power estimations. Each power estimation relates a specific terminal to a specific channel.

The selection of a specific terminal-channel assignment can be expressed through a selection variable x_{ij}. If $x_{ij} = 1$, then terminal-i is assigned channel-j, and if $x_{ij} = 0$, then terminal-i is not assigned channel-j. The following constraints are also needed in the formulation of the allocation problem.

1. A terminal cannot be allocated more than one channel; at cell k we have the following constraints

$$\begin{array}{lll} \text{Terminal 1 ``on'' channel } j \ (j = 1, ..., w): & x_{11} + ... + x_{1w} \leq 1 \\ \vdots & \vdots \\ \text{Terminal } r \text{ ``on'' channel } j \ (j = 1, ..., w): & x_{r1} + ... + x_{rw} \leq 1 \end{array} \tag{4.6}$$

2. It is also assumed that at no time can a channel be allocated to more than one terminal from a particular cell. Then at cell k we have

$$\begin{array}{lll} \text{Channel 1 ``to'' Terminal } i \ (i = 1, ..., r): & x_{11} + ... + x_{r1} \leq 1 \\ \vdots & \vdots \\ \text{Channel } w \text{ ``to'' Terminal } i \ (i = 1, ..., r): & x_{1w} + ... + x_{rw} \leq 1 \end{array} \tag{4.7}$$

Let us assume that we want to have a network with an average transmitter power per channel equal to $\bar{P}$. If q cells want to assign terminals to a particular channel, then the sum of the transmitting power from the terminals on that channel must be less than, or equal to, $q\bar{P}$. The channel allocation and power constraints can be combined to construct a knapsack problem. An example follows.

4.5.1 Example

For simplicity, formulation is obtained for only two cells, Cell ψ and Cell k. For Cell ψ, every step needed to obtain the desired information is explained. For Cell k, it is assumed that the procedure used in Cell ψ is followed, and the desired information obtained. c_{ij} is used to represents the estimated transmitting power of terminal-i on channel-j.

Let us assume that the number of channels at each cell is two, and under average load conditions the average transmitting power per channel is known to be $\bar{P}$.

An estimated power of the terminals waiting to be allocated channels at Cell ψ can be computed as follows.

Conditions at Cells ψ and vicinity Cells χ and ω:

- Cell ψ is empty; that is, there are two available channels. There are also two terminals waiting for channels, say, terminals 1 and 2.
- Cell χ is full; that is, all channels are in use. There are no terminals waiting for channels. Cell ω is empty; that is, there are two available channels. There are also three terminals waiting for channels, say, terminals $3, 4$ and 5.

Based on the given conditions:

1. The actual status of the network is reproduced on a phantom network. This status is: Cell χ is full, and Cells ψ and ω are empty. Therefore, the only terminals in the phantom network at this point are the terminals being attended by Cell χ.
2. Terminal-1 is assigned channel-1 at Cell ψ and then the terminal (among terminals $3, 4$ and 5) that will cause the greatest interference to terminal-1 is assigned to channel-1, at Cell ω. No allocation is attempted to channel-1 at Cell χ because there are no available channels (if there were, no allocation would be attempted because there are no terminals waiting for channels).

3. The DCPC mechanism is executed on channel-1 on our phantom network and an estimated transmitting power c_{11}, as if terminal-1 were to use channel-1, is obtained.
4. Steps $1-3$ are repeated, but now assuming that terminal 1 is assigned channel 2 at Cell ψ. An estimated transmitting power c_{12} is obtained, as if terminal-1 were to use channel-2.
5. Repeat, once again, steps $1-3$, but now for terminal-2 at Cell ψ. Obtain the corresponding power estimates c_{21} and c_{22}, as if terminal-2 were to use channel-1, and as if it were to use channel-2, respectively.

A selection variable is assigned to every terminal-channel pair generated from the previous steps. x_{11} to terminal-1/channel-1; x_{12} to terminal-1/channel-2; x_{21} to terminal-2/channel-1; and x_{22} to terminal-2/channel-2.

Cell ψ is "competing" for channel 1 because there exists the possibility of allocating the channel to a terminal. The cell is also competing for channel 2 for the same reason. The terms

$$\mathbf{X}^{\psi} = \begin{bmatrix} x_{11} & x_{12} & x_{21} & x_{22} \end{bmatrix}^T,$$

$$\mathbf{A}_1^{\psi} = \begin{bmatrix} c_{11} & 0 & c_{21} & 0 \\ 0 & c_{12} & 0 & c_{22} \end{bmatrix}, \qquad \mathbf{B}_1^{\psi} = \begin{bmatrix} \bar{P} \\ \bar{P} \end{bmatrix},$$

$$\mathbf{A}_2^{\psi} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}, \text{ and } \mathbf{B}_2^{\psi} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix},$$

where $x_{ij} \in \{0, 1\}$, can express the status at Cell ψ. That is, the term $\mathbf{A}_2^{\psi}\mathbf{X}^{\psi} \leq \mathbf{B}_2^{\psi}$ expresses the constraints of not allocating to a terminal more than one channel, and not

allocating a channel to more than one terminal. On the other hand, the term $\mathbf{A}_1^{\psi}\mathbf{X}^{\psi} \leq \mathbf{B}_1^{\psi}$ expresses that the transmitting power of the terminals has to be less than, or equal to, the average transmitting power $\bar{P}$.

Following the procedure just given, the status of Cell k can be obtained. Let us assume that Cell k is not completely empty, that there is one available channel, say, channel 1. Let us also assume that there are two terminals waiting for channels, say, terminals 8 and 9. After DCPC is applied to the phantom network, the following terms can be generated

$$\begin{aligned} \mathbf{X}^k &= \begin{bmatrix} x_{81} & x_{91} \end{bmatrix}^T, \\ \mathbf{A}_1^k &= \begin{bmatrix} c_{81} & c_{91} \end{bmatrix}, \quad \mathbf{B}_1^k = \begin{bmatrix} \bar{P} \end{bmatrix}, \\ \mathbf{A}_2^k &= \begin{bmatrix} 1 & 1 \end{bmatrix}, \text{ and } \mathbf{B}_2^k = \begin{bmatrix} 1 \end{bmatrix}, \end{aligned}$$

where x_{81} and x_{91} represent the terminal-channel pairs, and c_{81} and c_{91} represent the transmitting power estimates. The term $\mathbf{A}_2^k\mathbf{X}^k \leq \mathbf{B}_2^k$ expresses the constraints of not allocating more than one terminal to the only available channel, and the term $\mathbf{A}_1^k\mathbf{X}^k \leq \mathbf{B}_1^k$ expresses the estimation-power constraints.

A knapsack problem from Cells ψ and k can be described as shown in Equation 4.8. The formulation can include information from several cells. The upper part of the constraint matrix is related to the transmitting power of the terminals waiting to be allocated (cells competing for channels). The rest of the constraint matrix is related to terminal-channel assignments (channel assignment constraints). The constraint matrix is also padded with zeros to obtain the proper matrix dimension.

$$\max \sum_{i,j} x_{ij}$$

$$\underbrace{\begin{bmatrix} c_{11} & 0 & c_{21} & 0 & c_{81} & c_{91} \\ 0 & c_{12} & 0 & c_{22} & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}}_{\begin{matrix}\text{Cell } \psi & \text{Cell } k\end{matrix}} \begin{bmatrix} x_{11} \\ x_{12} \\ x_{21} \\ x_{22} \\ x_{81} \\ x_{91} \end{bmatrix} \leq \begin{bmatrix} 2\bar{P} \\ \bar{P} \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \begin{matrix} \} \text{ Cells competing for channels} \\ \\ \\ \\ \} \text{ Channel assignment constraints} \\ \\ \\ \end{matrix} \tag{4.8}$$

Using the ideas discussed, the channel allocation problem can be formulated as

$$\begin{aligned} &\max \sum_{ij} x_{ij} \\ \text{subject to: } &\sum_{ij} a_{ijk} x_{ij} \leq b_k \quad k = 1, ..., m \\ &x_{ij} \in \{0, 1\} \end{aligned} \tag{4.9}$$

The formulation changes if we consider an instance when a terminal must be allocated a channel. Some inequalities from the channel assignment constraints group will become equalities. If by some means we also assign values to the channels (that is, if we introduce coefficients for the selection variables in the objective function), then the goal of the allocation scheme should be to maximize such a value instead of maximizing the number of terminals attended.

The solution of the knapsack problem will not generate an actual allocation on the system. The knapsack was obtained based on local information and assuming a certain allocation on the phantom network. Moreover, assuming that terminal i is actually allocated channel j, then the assumptions based on terminal i allocated on other channel no longer

hold. Nevertheless, it was observed that the use of pseudo-utility values (from the knapsack problem) as a means to identify terminal-channel allocation is helpful. The pseudo-utility values can be used to obtain an order by which channel allocations are attempted. These values are computed using heuristics for solving the knapsack problem, including [51], [38], and [56]. In this work the heuristic presented in [51] was used.

4.6 Channel Allocation Attempts

As mentioned, pseudo-utility values can be used to choose the selection variables, x_{ij}, that would maximize the number of terminals attended. These values are ratios of the objective function's coefficients to factors of the resource constraints' coefficients.

In the approach presented in [51], MKHEUR, the pseudo-utility values are given by

$$u_{ij} = \frac{v_{ij}}{\sum_{k=1}^{m} w_k a_{ijk}} \tag{4.10}$$

The coefficients w_k are Lagrange multipliers from the solution of the Linear Programming Relaxation of the Knapsack Problem (LPR-KP). In the LPR-KP we assumed that $0 \leq x_{ij} \leq 1$.

4.6.1 Resource Allocation Strategy

Following is the description of the allocation scheme that is based on the knapsack formulation.

Ordered-DDRA

1. Create the knapsack problem given in Equation 4.9. Solve the LPR-KP, and obtain the Lagrange multipliers.

2. Calculate the pseudo-utility values using Equation 4.10. Sort and renumber the selection variables in the decreasing order of these values.
3. Allocate channels to terminals as expressed by the selection variables, and according to the order determined in Step 2. After each allocation, perform the DCPC mechanism. If, while performing DCPC, it is observed that the new admitted terminal will cause the rejection of any terminal already in the network, or the rejection of the lately-admitted terminal itself, then reject the lately-admitted terminal and continue. While at a particular cell, if the terminal-i-channel-j allocation is successful, then the remaining variables that considered terminal-i or channel j at that particular cell are discarded.
4. Using the terminals not yet attended, repeat Steps $1-3$. Repeat Step 4 as many times as needed. Stop when either a knapsack problem cannot be constructed (that is, when a single terminal needs allocation) or when the KP has solution zero (that is, when all estimated transmitting powers are greater than the "available" transmitting power (from Cells competing for channels, see Equation 4.8)). If there are terminals waiting to be allocated and available channels at their respective cells, then apply the standard DDRA mechanism.

The allocation mechanism just described can be represented by the block diagram shown in Figure 4.4. That is, with information from the actual system, we create a phantom network and use it to formulate a knapsack problem. Next, with the use of pseudo-utility values, attempts to allocate channels are carried out (Trial Generator). How trials are attempted also depend on how the actual system evolves.

4.7 Simulation Results

For comparison purposes, the values used in our simulations are similar to the ones used in [25],

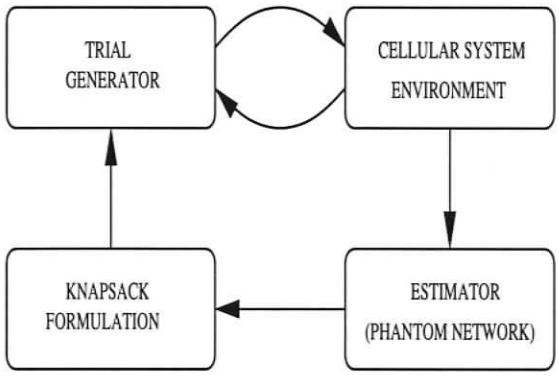

Figure 4.4. *Cellular channel allocation*

- $G_{ij} = d_{ij}^{-4}$, where d_{ij} is the distance between the i-th BS and the j-th terminal.
- The DDRA is applied in an asynchronous fashion, and 80 cycles of power control are performed before the next terminal is served. The decision as to whether a terminal is rejected is taken after a minimum of 30 cycles of power control have been performed.
- γ_f, for the system, equals 15.89 dB. γ_t, for power control, equals 18.89 dB. P_{max} is set to 1 W, and the receiver noise power equals 10^{-10} W.
- The statistics are taken from the inner 20 cells. The cell size is set to 100 m. The number of terminals in a cell is a Poisson random variable with mean λ_c. N is the number of channels and $\lambda = \lambda_c/N$ is the mean traffic per cell per channel.
- The mean transmitting power per terminal, $\bar{P}$, which was used in the phantom network, was set equal to 0.1 W.
- The cellular system is a one-dimensional system of 50 cells. The length of the vicinity was set equal to one. That is, for a given cell, only the adjacent cells are considered to form the vicinity of that given cell.

The snapshot analysis used does not consider the time correlation properties of the traffic. However, it can be used as a first tool in the analysis of resource allocation schemes. A snapshot occurs at a randomly chosen instant of time. In the standard-DDRA scheme, a snapshot analysis is performed every time a new terminal arrives (that is, on a first-come

first-served basis). In the ordered-DDRA scheme, a snapshot analysis is performed when all terminals have arrived. That is, in the ordered-DDRA approach terminals must wait (however long this might take) to be attended until all terminals have arrived. In both schemes we assume that a number of terminals are already on the system when either DDRA scheme is used. For a given λ, the total number of terminals generated form a group. A total of 2000 groups were evaluated for a given λ. The probability of assignment failure is computed as given by Equation 4.4. Numerical results on Tables 4.1 and 4.2 show the assignment failure probability for the schemes evaluated.

Table 4.1. *Standard-DDRA and ordered-DDRA schemes*

Mean traffic λ	Assignment failure $\bar{P}_a$ for DDRA	
	Standard	Ordered
0.20	0.254	0.131
0.25	0.649	0.386
0.30	1.288	0.878
0.35	2.115	1.458
0.40	3.094	2.437

$N = 5$ Channels

No rejections are performed when the DCPC mechanism is applied to the phantom network. However, if the CIR of the terminal-channel pair for which we are obtaining a power estimation is less than zero, then that pair is not considered in the knapsack formulation.

Table 4.2. *Standard-DDRA and ordered-DDRA schemes*

Mean traffic λ	Assignment failure $\bar{P}_a$ for DDRA	
	Standard	Ordered
0.20	0.011	0.019
0.25	0.059	0.030
0.30	0.204	0.063
0.35	0.456	0.166
0.40	1.004	0.356

$N = 10$ Channels

4.7.1 Channel Reallocation

Terminals can be reallocated in order to reduce, if possible, the probability of rejection for new terminals. As mentioned in Section 4.5, each inequality constraint in Equation 4.6 expresses that a terminal cannot be allocated more than one channel. If a terminal is to be reallocated, then it must be allocated a channel; in this case, the corresponding inequality constraint in Equation 4.6 needs to be expressed as an equality constraint.

We have evaluated the reallocation case, and the results are provided in Tables 4.3 and 4.4. In this situation, new terminals are allocated using the standard-DDRA scheme, and when the cellular system load reaches a predefined threshold value a reallocation procedure is executed. All allocated terminals are considered for reallocation; no terminal is dropped during the reallocation process. The performances of the standard-DDRA and standard-DDRA *plus* reallocation schemes are compared. Simulation results show that the combined strategy has a superior performance.

Table 4.3. *Standard-DDRA and standard-reallocation-DDRA schemes*

Mean traffic λ	Assignment failure probability and transmitter power							
	5 Channels				10 Channels			
	Standard		Reallocation		Standard		Reallocation	
	$\bar{P}_a$	Power	$\bar{P}_a$	Power	$\bar{P}_a$	Power	$\bar{P}_a$	Power
0.200	0.254	0.108	0.233	0.109	0.011	0.107	0.014	0.108
0.250	0.649	0.109	0.579	0.111	0.059	0.107	0.044	0.109
0.300	1.288	0.113	1.190	0.116	0.204	0.108	0.165	0.110
0.350	2.115	0.116	1.906	0.120	0.456	0.110	0.406	0.112
0.400	3.094	0.120	2.866	0.128	1.004	0.113	0.823	0.119

In general, during the reallocation process a terminal is briefly attended by two channels (the one originally assigned and the one we intent to reallocate to). If the reallocation attempt is successful, then this new allocation prevails and the original is destroyed (that is, the terminal has been successfully reallocated).

In Table 4.3, we provide the numerical results for five and ten channels per cell, and at five different mean traffic load conditions. Reallocation is performed when 20, 40 and 60% of the system's channels have been used. The column "Power" represents the average transmitter power on the network. It can be observed that if reallocation is included, then the assignment failure probability decreases. This decrement is obtained with a marginal increment in the transmitter power.

Although this work is done with FDMA and TDMA networks in mind, we think that the strategy presented can be applied, with the proper modifications, to CDMA and other mobile communication systems. We think that the demanded power increment, after reallocation is performed, is well invested when the reduction in terminal rejection is evaluated.

Table 4.4. *Allocation schemes' comparison*

Mean traffic λ	Percentages in increment on $\bar{P}_a$ and Power			
	5 Channels		10 Channels	
	$\bar{P}_a$	Power	$\bar{P}_a$	Power
0.200	-8.738	0.910	16.667	0.828
0.250	-12.226	2.000	-33.333	2.112
0.300	-8.259	2.702	-23.394	2.200
0.350	-10.990	3.512	-12.141	2.143
0.400	-7.970	6.014	-21.892	4.513

For example, in the reallocation case, at a mean traffic load of 0.35 and using 5 channels, there is a reduction in the assignment failure probability of almost 11%, but an increment of only approximately 3.5% in transmitter power (see Table 4.4).

4.8 Real-Time Considerations

For the simulated cellular system, the time needed to compute a possible channel allocation (or reallocation) is approximately 15 msec when considering 5 channels and a mean traffic load $\lambda = 0.4$. This time increases to 90 msec when considering 10 channels and a mean traffic load $\lambda = 0.4$. These times are related to an AMD 1.4 GHz Personal Computer, with 264 MB of RAM, and running Mandrake Linux OS, kernel 2.4.21-0.13mdk.

An important assumption in this work is that the system does not change significantly

while the above computations are taking place. A possible case is where users move at a pedestrian speed of 1.5 m/sec; they will move only 14cm during this time (90 msec). Even at urban speeds of 30 km/h, a user will move 75cm. That is, the assumptions in users' locations and power consumptions can be considered not to change significantly. Furthermore, the probability of receiving new calls during this time is low since call interarrival times are in the order of seconds. However, for fast-changing or very large cellular systems, the approach proposed may not be suitable.

4.9 Conclusion

The numerical results suggest that the use of pseudo-utility values to obtain an order by which channel allocation could be attempted produces a superior performance to that of a random order. The single allocation and the single reallocation cases evaluated support this observation.

For the reallocation case, we explored whether the performance observed is consistent. A statistical analysis of the probability of rejection on the groups of terminal evaluated was carried out. This analysis is provided in Appendix A, Section A.2. The results indicate that the performance obtained from the strategy proposed can be considered consistent.

Chapter 5

Resource Allocation on the Grid

In this chapter, we present a Grid resource allocation strategy that uses utility functions and is based on a knapsack formulation. First, we briefly describe the Grid concept, applications and standards (descriptions are taken from [18] and [19]). Then, we provide an example of how to express the resource allocation problem as a knapsack problem, and how to use utility functions to implement desired allocation strategies. Using a variety of allocation policies, simulation results show that a knapsack formulation allocate resources according to the chosen policy. The work presented in this chapter is also described in [50].

5.1 Introduction

The term "Grid" is used to denote a distributed computing infrastructure for applications that demand enormous amounts of computational resources and specialized devices. The Grid concept was inspired by the pervasiveness, ease of use, and reliability of the electrical power grid. In the mid-1990s, computer scientists began exploring the design and development of an infrastructure for wide-area parallel and distributed computing. However, the idea of a Grid was first envisioned by Internet pioneer J.C.R. Licklider, in the early 1960s [18].

A Grid enables the sharing, selection and aggregation of a wide variety of geographically distributed resources including, but not limited to, supercomputers, storage systems,

data sources, and specialized devices owned by different organizations. These geographically distributed resources can be selected for use through access points that are also geographically distributed (Figure 5.1). The Grid is a tool that aims to solve large scale resource-intensive problems. In the Grid environment we find heterogeneous resources that can be used for a wide range of applications; however, resources are subject to local systems and management policies.

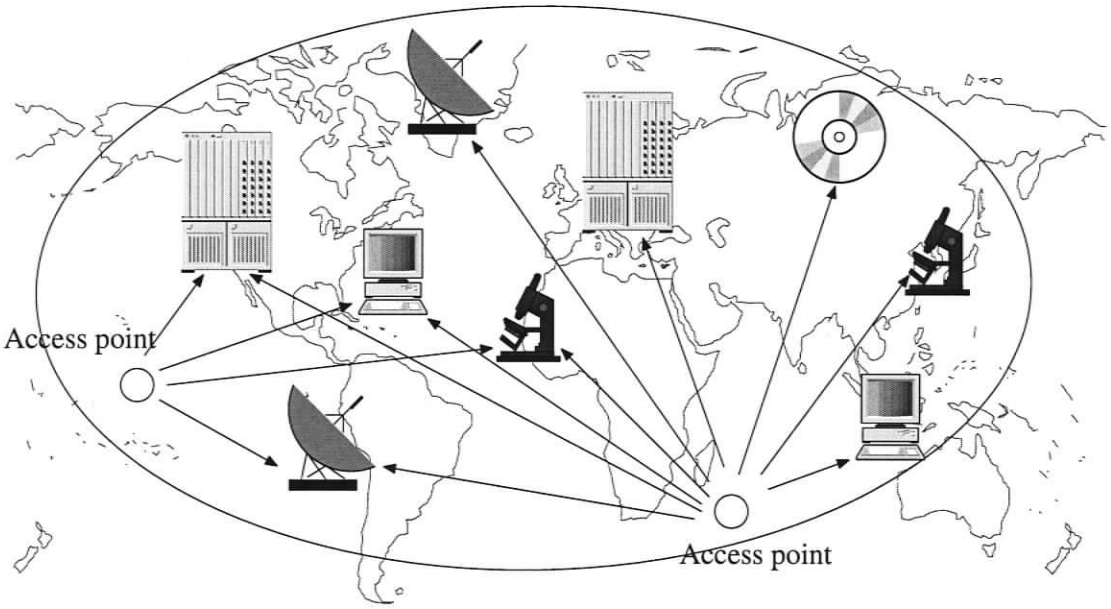

Figure 5.1. *The Grid*

5.2 Grid Applications

It is clear that Grid applications such as those listed below show the Grid's benefits:

- the Distributed Aircraft Maintenance Environment project, which applies Grid-based technologies to the problems of aircraft engine diagnosis, prognosis and maintenance [6];
- NEESgrid, a Grid-based system constructed to link earthquake researchers across the

U.S.A. through computational resources and other equipment, thus allowing collaboration and a more efficient use of resources [8];

- the Virtual Observatory, which aims to integrate astronomy archives into a database containing astronomy literature, images, raw data, derived datasets, and simulation data. This integration will create a virtual telescope [55, 1, 2, 4, 5]; and
- DataGrid, which aims to enable the storage and analysis of real and simulated data from high-energy physics experiments at CERN. The DataGrid is now a part of the EGEE project [3, 7]

However, standards, interfaces and protocols are needed if what is sought is the integration of technologies, applications, files and so on. Interfaces and protocols virtualize resources; that is, users can focus on what they wish to do, rather than on how the resources and technology work. If standards are followed in the creation of interfaces and protocols, then the delivery of a variety of Grid computing services will be possible.

A Grid community is in the process of establishing the necessary conditions that will enable Grid capabilities. The Open Grid Services Architecture is a community standard that defines a core set of interfaces and behaviours that address many of the technical challenges involved in these sharing relationships, such as the implementation of policies, authentication, authorization, and quality of service [19]. The Open Grid Services Architecture provides the framework upon which Grid systems can be constructed.

5.3 Open Grid Services Architecture (OGSA)

The three principal elements of OGSA are the Open Grid Services Infrastructure (OGSI), OGSA services, and OGSA schemas (see Fig 5.2).

Although OGSA builds on Web services[1], Web service standards do not address is-

[1]Web Services is a technology that addresses the integration of heterogeneous distributed information systems. The most important innovation is the separation in the definition of services from the mechanisms

sues such as how services are created, how long they live, and how faults are managed. These, and other, issues are addressed through a core set of interfaces called the Open Grid Services Infrastructure (OGSI) [57].

OGSA services are built upon OGSI; that is to say, services follow OGSI standards. OGSA services are vast and include service discovery, service management, monitoring, security, and data access. On the other hand, OGSA schemas provide the common vocabulary to achieve interoperability among Grid components, in order to build large-scale Grid systems and to achieve code reuse; that is to say, the OGSA schemas define the standard schemas for describing the properties of common Grid entities.

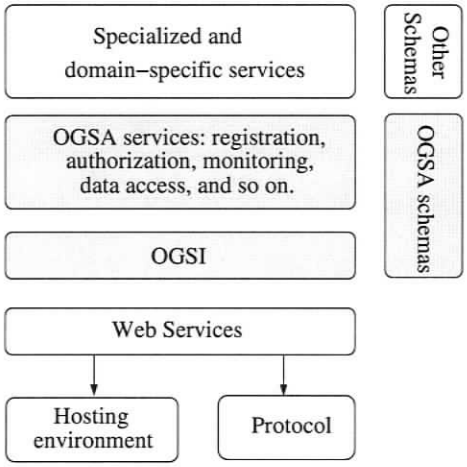

Figure 5.2. *The Open Grid Services Architecture*

Grid services can be arranged in three broad groups:

- Core Services that address issues such as name resolution, service discovery, security, policy, and messaging.
- Data and Information services, which address issues such as data naming and access, replication, and metadata and provenance.

by which those services are invoked.

- Resource and Service Management.

The work presented in this chapter is related to Resource and Service Management, which refers to the operations that control how capabilities provided by resources and services are made available to other entities (whether users, applications, or services). Resource management is not concerned with the core function of a resource or service (what it does for the clients) but with the manner in which this function is performed.

5.4 Resource Allocation

In general, a Grid is comprised of Clients, Servers and a Resource Management Centre (see Figure 5.3). Clients generate tasks that require resources for their execution; Servers provide these resources. The Clients-Servers relationship is established at the Resource Management Centre (RMC).

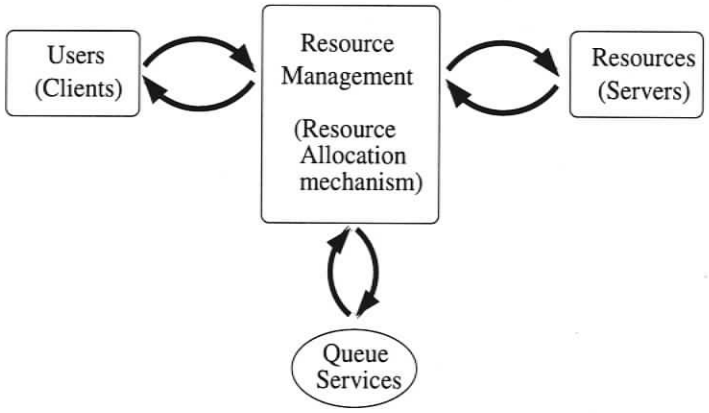

Figure 5.3. *Grid components*

The core goal of resource management is to establish a mutual agreement between a Server and a Client by which the Server agrees to supply a capability that can be used to perform a task on behalf of the Client. Because Grid systems and applications include a rich variety of both tasks and resources, there are several possible resource management

situations that can be present in the Grid [18]. One possible situation involves Resource Brokering, whereby a broker service acts as an intermediary between a set of resources capabilities and users; that is, the broker directs tasks to appropriate resources, based on broker-specific policy. The resource allocation mechanism is an important element of a resource-broker strategy.

Resource allocation techniques can be classified according to two criteria [24]. The first involves the description of the Grid, which can be according to a current snapshot of the Grid (state-based) or to a model of the Grid (model-based). The second criterion involves task behaviour. If tasks assigned to hosts are allowed to migrate to other hosts, we have a preemptive environment. If tasks are assigned to hosts only once, we have a non-preemptive environment.

In this work, we assume that a form of state-based description and preemptive strategy are used in the formulation of the resource allocation problem on the Grid. That means that we expect that tasks are checkpointable and that tasks remain at the resource allocated until they complete or until a scheduling time slice has expired and the task is checkpointed. A mechanism decides on the allocation of resources either upon task arrival or at the beginning of the next scheduling time slice.

We assume that there exists a mechanism (or mechanisms) capable of identifying the available Grid resources at all times. We also assume that metadata associated with the task describes its resource requirements, or that there exists a mechanism (or mechanisms) able to determine the same.

It is understood that the way a task is allocated resources will have an impact on its completion time. Therefore, the selection of a particular allocation policy and the association of policy-specific values to different possible resource allocations, or options, for each task will allow us to control the task completion time.

In this chapter, we use the term Quality of Service (QoS) to refer to the quality of an

allocation option. The choice of resources affects a task's completion time, which is used as the quality criterion. QoS metrics associated with each allocation option quantify the effect of the option (if selected) on the task's completion time. The concept of utility is used to represent the quality of the allocation options and to provide a means of optimizing the allocation of resources.

A task is described by a set of resources needed to conclude its computation (such as the number of computational units (processors), storage requirements, and length of computation). Each task lists a number of options, each of which requires a different amount of the resources listed above. Each option has associated with it a set of QoS metrics. These metrics are used to compute the utility value of that particular option.

There are several strategies or procedures that can be used to perform the resource allocation, including:

- First-Come First-Served. Tasks are allocated on a first-come first-served manner, provided there are available resources on the Grid. A task that cannot be attended, blocks all subsequent tasks, until adequate resources are found and allocated to it.
- Back Filling. Tasks are allowed to skip ahead under certain conditions.
- Gang Scheduling. Gangs (groups) of interdependent tasks are allocated according to a time-slicing schedule.

In this work, we use a knapsack resource allocation strategy. When this strategy is compared with the above-mentioned allocation procedures, numerical results indicate that it presents a superior performance.

5.5 Knapsack Formulation

In this section, we cast the resource allocation problem on the Grid as a knapsack problem. The different resources on the Grid define the dimensions in the knapsack; the total number

of units of each resource define the capacity of the knapsack in each dimension. Utility functions define the value of the coefficients in the knapsack's objective function.

The use of integer programming, including the knapsack formulation, to cast combinatorial problems is not new. A vast number of examples can be found in the literature. Also, the use of utility functions is not new; the have been used by economists for several decades as mentioned in [35] and [36]. However, as far as we know, using the notion of utility, together with knapsack formulations, as an allocation strategy to implement policy-based scheduling on Grids, is new and requires further exploration.

Knapsack formulations have been used to find feasible schedules for computational tasks [44]. Here, we are interested in its use as a means to provide QoS to a schedule. Thus, we express the allocation problem as a knapsack problem and introduce the notion of utility of a task. Then, we allocate the finite resources of the Grid so that the total utility of all the tasks is maximized, while the constraints introduced by the finite resources of the Grid are maintained.

Generally, the steps that take place in the knapsack allocation strategy are the following:

- After users submit tasks to the Grid, those tasks arrive at the RAC and are queued waiting to be allocated the necessary resources to run.
- Each task is associated with a number of options. Each option lists the resources it requires. These options can be constructed by different means. For example, a user can describe different ways his/her task runs, such as the number of processors for each way (say six or eight). Next, a Grid mechanism discovers a number of explicit allocations able to support the task's different ways of running, for example: Cluster A might support six processors, and Cluster B eight. In this work, we assume there is a mechanism (or mechanisms) able to construct explicit allocation options for the task.
- A utility value is then assigned to each option. The utility value is a measure of the

option's "desirability" or "usefulness" to the owner of the task as well as to the Grid itself. The value depends on the metrics of the computation task (these may include the length of the computation, the accumulated computation time, or the existence of deadlines) but also on metrics that reflect the value that the owner of the task attaches to that task. A mapping transforms the metrics to a utility value for the option.
The relationship among the metrics in the mappings depends on the allocation policies. Such policies can be global (the Grid might give priority to long tasks) or local (some domains in the Grid might give priority to long tasks, others to short ones).

- Utility values are then used by the RAC to allocate resources to tasks in an optimum manner.

5.5.1 The Utility Values

As mentioned, utility values are used to describe the different allocation options for a set of tasks. The utility represents the relative merit of each option and depends on the QoS metrics associated with each. A utility function $U(\cdot)$ is defined and used to compute the utility values from these metrics. In general, QoS metrics can be classified as follow.

- Intrinsic. The QoS metrics are calculated based on the inherent properties of the task under the said option. These may include the task's computation length, accumulated time, and deadline, if one exists, and so forth.
- External. An external metric usually reflects the value and desirability of the computation option to the owner of the task, who then assigns a value to that metric. These values cannot be assigned arbitrarily; rather, each user is allocated a maximum credit corresponding to a portion of the existing Grid resources, which the user manages by allocating portions of it to his/her tasks. An option which is highly desirable to the user (for example, an option that will result in a fast response time for an urgent task) may be given a large portion of the available credit.

A utility value for option j of task i is denoted by u_{ij}, and is obtained by the evaluation of the option's utility function $U_{ij}(\cdot)$; this function depends on the QoS-metric-parameter P_{ij}; that is,

$$u_{ij} = U_{ij}(P_{ij}) \tag{5.1}$$

An example of a saturating utility function and its graph are given in Equation 5.2 and in Figure 5.4.

$$U(x) = \begin{cases} 0 & \text{when } 0 \leq x \leq \theta \\ \dfrac{2}{e^{-\alpha(x-\theta)}+1} - 1 & \text{when } \quad x > \theta \end{cases} \tag{5.2}$$

The QoS-metric-parameter P_{ij} is computed from the intrinsic and external QoS metrics p_{ijl} previously discussed. If we denote by p_{ijl} ($l = 1 \ldots m_{ij}$) the l^{th}-QoS metric for option j of task i, then the QoS-metric-parameter P_{ij} could be calculated by

$$P_{ij} = \frac{\sum_l w_l S(p_{ijl})}{\sum_l w_l} \tag{5.3}$$

where:

w_l represents non-negative weighting factors and

$S(\cdot)$ represents a non-decreasing positive function.

The same function as that given in Equation 5.2 can be used as the function $S(\cdot)$, or a different one can be constructed. By varying the weights w_l in the the QoS-metric-parameter P_{ij}, different policies can be implemented. For instance, let us assume that T_i denotes the run time of task i; then, tasks with short run times are preferred if $p_{ij1} = 1/T_i$ and $w_l = 0$ but $w_1 = 1$.

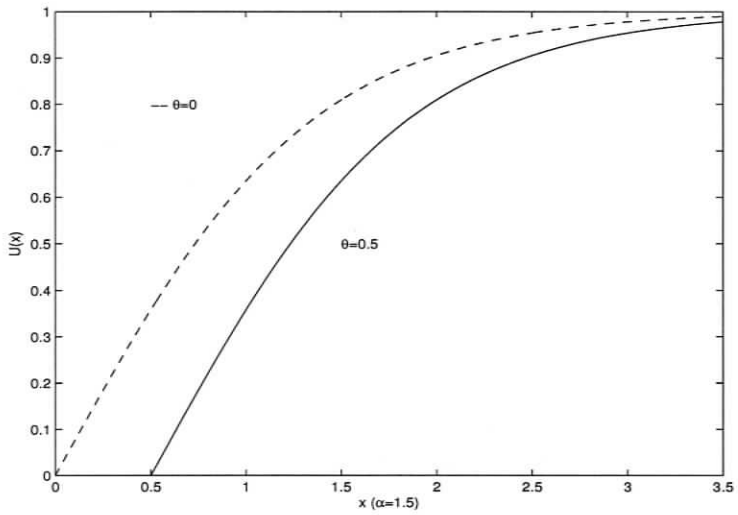

Figure 5.4. *Example of a utility function* $U(x)$

The type of function involved in the computation of the utility values is not limited to the examples given above. In section 5.6, a variety of functions are considered. As mentioned, the selection of the proper mapping of the QoS metrics will depend upon the desired relationship among tasks, options, and the the Grid policies.

5.5.2 Example Formulation

In this Section, we provide an example of a resource allocation formulation where only two tasks are considered. We assume that a Grid is comprised of a number of servers (usually clusters of processors, high performance computational resources or other specialized devices). Each server includes a number of resource types, each of a certain quantity. In this example, we use the following notation:

- R_{kl} represents the quantity of resource l available in server k;
- a_{ij}^{kl} represents the requirement of resource type l from server k by task i as per option j; and
- variable $x_{ij} \in \{0, 1\}$ represents whether task i was allocated option j ($x_{ij} = 1$) or

not ($x_{ij} = 0$).

Assume that a certain Grid comprises two clusters *Cluster1* and *Cluster2*. *Cluster1* includes R_{11} fast machines, while *Cluster2* includes R_{21} slower machines. At time t, two tasks, *Task1* and *Task2*, request resources from the Grid. Each task has a number of options enumerated. In this example, the only resource that is to be allocated is that of processors. The utility value associated with each option is calculated from an external metric assigned by the owner of the task in terms of value units that correspond to the desirability of the particular option. We also assume that *Task1* must be allocated, while *Task2* may or may not be allocated.

Thus *Task1* lists the following three options:

- option 1 requires a_{11}^{11} processors from *Cluster1*. The user assigns an external value $p_{111} = M_{11}$ to the QoS metric p_{111} associated with this option.
- option 2 requires a_{12}^{11} processors from *Cluster1*. The user assigns an external value $p_{121} = M_{12}$ to the QoS metric p_{121} associated with this option.
- option 3 requires a_{13}^{11} processors from *Cluster1* and a_{13}^{21} processors from *Cluster2*. The user assigns an external value $p_{131} = M_{13}$ to the QoS metric p_{131} associated with this option.

On the other hand, *Task2* lists the following two options:

- option 1 requires a_{21}^{11} processors from *Cluster1*. The user assigns an external value $p_{211} = M_{21}$ to the QoS metric p_{211} associated with this option.
- option 2 requires a_{22}^{21} processors from *Cluster2*. The user assigns an external value $p_{221} = M_{22}$ to the QoS metric p_{221} associated with this option.

Given the above description of the proposed options, one can compute the utility values associated with each as $u_{ij} = U_{ij}(S(p_{ij1}))$. Accordingly, we can now formulate the constrained optimization problem as in Equation 5.4.

$$
\begin{aligned}
&\max f(x) = u_{11}x_{11} + u_{12}x_{12} + u_{13}x_{13} + u_{21}x_{21} + u_{22}x_{22} \\
&\text{subject to:} \\
&\begin{aligned}
a_{11}^{11}x_{11} + a_{12}^{11}x_{12} + a_{13}^{11}x_{13} + a_{21}^{11}x_{21} &\leq R_{11} \\
a_{13}^{21}x_{13} + a_{22}^{21}x_{22} &\leq R_{21} \\
x_{11} + x_{12} + x_{13} &= 1 \\
x_{21} + x_{22} &\leq 1 \\
x_{ij} \in \{0,1\}
\end{aligned}
\end{aligned}
\tag{5.4}
$$

It is clear that finding the optimal allocation is not a challenge in this example. However, when a large number of tasks, options, servers and resources, such as such as memory sizes, interconnect bandwidth, file size, and so on have to be considered, then the selection of the algorithm to solve the knapsack problem matters [51], [38], [46], [47].

5.6 Allocation Strategies

A set of non-knapsack and knapsack allocation strategies were evaluated in this work. A description of these strategies and the policies implemented follows.

5.6.1 Non-Knapsack Strategies

Three non-knapsack allocation strategies were evaluated. These strategies are:

First-Come First-Served (FCFS). Tasks in the queue are allocated on a first-come first-served manner, provided there are available resources on the Grid. That is, if the task at the head of the queue cannot be attended, then the task remains in the queue until adequate resources are found and allocated. While in the queue, this task blocks subsequent tasks; this means no other task is attended before the task at the head of the queue.

Back Filling (BF). In this case, some tasks might be allowed to skip ahead of the one at the head of the queue; that is, some later tasks might be attended before resources are found and allocated to the first one in the queue. There are different strategies for back filling, the one implemented in this work is the EASY backfilling. Using this strategy, short tasks are allowed to skip ahead, provided they do not delay the task at the head of the queue [37].

Gang Scheduling. In this strategy [45], processes of a same task run for a certain period of time called time-slicing schedule. At the end of this schedule, processors context-switch to give service to processes of another task. That is, tasks in the system receive service in a coordinated fashion. At each context-switch, running tasks are moved to the end of the queue, and then the queue is allocated using a FCFS strategy. The time quanta explored are equivalent to those used with knapsack strategies.

5.6.2 Knapsack Strategies

A *knapsack strategy* and a combined *knapsack and a non-knapsack strategy* were also evaluated, and are described below.

Knapsack-based allocation. This strategy allocates resources as discussed in Section 5.5 above. Tasks are allocated at set *scheduling time slices* τ (for example, every 12 hours), and are checkpointable. Tasks that have accumulated during the previous allocation interval, along with tasks that are check-pointed at the end of the previous allocation interval, participate in the resource allocation process. As mentioned, each task offers a number of options. The allocation policy formulates a utility function, and the allocation mechanism chooses the options that will maximize the total utility of the Grid. A number of utility functions reflecting policy choices can be constructed. Some of these policies are listed below:

(a) Credit-value-driven policy.

Under this policy, a user is given a fixed number of credits to be managed by allocat-

ing different amounts to each task submitted. Credits loosely represent the value of the computation. For example, a more valuable computation task is awarded a higher credit-value, and a faster computational option is attributed a higher credit-value. The utility value u_{ij} can be computed as

$$u_{ij} = P_{ij} = v_{ij} \tag{5.5}$$

where v_{ij} represents the credit-value assigned to task i under option j. For the simulations reported in this work, we have made the following assumption. For task i under options m and n, and if T_{im} and T_{in} are the anticipated run times under the respective options, then

$$\frac{v_{im}}{v_{in}} = (\frac{T_{in}}{T_{im}})^{\alpha} \tag{5.6}$$

where $\alpha > 1$. That is, the credit-value metric assigned to the different computational options for a given task varies non-linearly with the computational capability of the option considered. In the knapsack strategy, the selection of options is based on ratios of the utility values to the resources consumed. The non-linear variation in the credit-value among the options for a task allows differentiation of these options. Different nonlinear expressions might be chosen, depending on the Grid environment and on the desirability of allocating premium resources.

(b) Credit-value-driven mediated by the estimated response-time policy.

Under this policy, a computational option is assigned a number of credits in accordance with the policy discussed previously. Along with the credit, a factor is introduced that reflects the estimated termination time of the task under the option considered; that is,

$$u_{ij} = P_{ij} = v_{ij}c_{ij} \tag{5.7}$$

where c_{ij} represents the normalized estimated task completion time of task i under option j. If we denote by t the present time, by s_i the submission time for task i, by r_{ij} the remaining computation time for task i under option j, and by N_i the

computation time of task i (as if it were to run on a standard notional computational resource), then one can calculate the normalized estimated task completion time as

$$c_{ij} = \frac{(t - s_i + r_{ij})}{N_i} \tag{5.8}$$

From Equation 5.8, we observed that, under the present policy, tasks which wait for long periods of time slowly increase their value, and eventually have the opportunity to run.

(c) Credit-value-driven mediated by the estimated response time and the closeness to task completion policy.

Under this policy, a term that increases the priority of tasks which are very close to completion is included as part of policy (b). That is, the utility value of task i under option j is

$$u_{ij} = P_{ij} = v_{ij}(c_{ij} + \frac{N_i}{r_{ij}}) \tag{5.9}$$

Thus, the closer the task is to completion, the larger its value becomes, and it eventually have an opportunity to run.

(d) Non-credit-value policies.

These are identical to the policies described in (b) and (c) above, but without the credit-value metric. That is, the utility value is given as

$$u_{ij} = P_{ij} = c_{ij} \tag{5.10}$$

or as

$$u_{ij} = P_{ij} = c_{ij} + \frac{N_i}{r_{ij}} \tag{5.11}$$

(e) As mentioned in section 5.5.1, the utility values are obtained through evaluating a utility function. The utility function depends on the QoS-metric-parameter P_{ij}. In the previously described policies, P_{ij} is based on the credit-value, normalized estimated task completion time, and closeness to task completion. Note that the identity

function has been used as the utility function in these policies. However, a different set of policies can be obtained, if, for example, the utility values are computed using a sigmoidal utility function (as in Equation 5.12) instead of the identity function,

$$u_{ij} = U_{ij}(P_{ij}) = \frac{2.0}{\exp(-5P_{ij}) + 1} - 1 \tag{5.12}$$

Note that, depending on the values of P_{ij}, a normalization might be needed in order to avoid saturation of the utility function.

Knapsack-based allocation and Backfilling (KBA + BF). Under this combined strategy, the policies implemented are identical to the knapsack-based policies discussed above. Additionally, a BF strategy is used to allocate tasks between the scheduling time slices used by the knapsack-based policies. This BF strategy works as previously described. However, at the end of a scheduling time slice all dispatched tasks are check-pointed and, together with the waiting tasks, participate in a knapsack allocation, in accordance with our earlier discussion.

5.7 Simulation Results

In order to evaluate the ideas discussed, we have simulated the following task allocation policies using the Grid simulator Simgrid [11].

- **Policy 1.** This is the credit-value-driven policy.
- **Policy 2.** This is the credit-value-driven policy mediated by the estimated response-time (Equation 5.7)
- **Policy 3.** This is the same as Policy 2, but using a sigmoidal utility function (Equation 5.12).
- **Policy 4.** This is a response-time plus closeness-to-completion driven policy (Equation 5.11)

- **Policy 5.** This is the same as Policy 4, but using a sigmoidal utility function (Equation 5.12).

We have assumed a Grid consisting of four clusters, with 4, 8, 16 and 32 processors, respectively. The clusters were assumed to be homogeneous. Sequences of randomly generated tasks were allocated according to the allocation policies discussed in Section 5.6.

Task arrivals were Poisson with mean inter-arrival time λ, which varied for different experiments. The computation demand for each task (expressed as the required computation time on a standard notional computational resource, that is, a single processor) followed a bimodal distribution function. Each lobe of the distribution was a Gaussian with mean and standard deviation as follows. Short tasks have a length mean value of one day (on a standard notional computational resource), with a standard deviation of 12 hours. Long tasks have a length mean value of eight days, with a standard deviation of 48 hours. Fifty percent of the tasks are short tasks.

Each task was allocated a number of credit-value units that followed a Gaussian distribution. The mean credit-value was ten units with a standard deviation of four units. For each task, we generated a number of options by varying the requested number of processors and assigning computation times accordingly. For each option, we also varied the number of credit-value units, as per Equation 5.6. Resources per options were described explicitly; that is, an option describes the amount and location of the resources for the said option. The task sequences used for this work are available for further research upon request.

Each experiment consisted of a sequence of 2000 tasks; utility values for the options and their allocation were performed according to the policies discussed in Section 5.6. For each sequence, we recorded its completion time as well as the submission s_i and actual completion C_i times for each task i in the sequence, and calculated their response times as $R_i = C_i - s_i$. We have further considered the performance of the FCFS policy as the baseline, to which the performance of the proposed policies is compared. That is, we

have calculated the speedup achieved on a per task basis by dividing the time response F_i of task i under the FCFS strategy, with the time response R_i achieved by each of the proposed policies. We have used the per-task speedup metrics and correlated these to the per-task assigned credit-value metric. The per-task credit-value is the maximum credit-value, among all the options, for a given task. Correlation between these variables allows us to evaluate whether the responsiveness of a policy relates to the QoS metrics that describe the tasks. The correlation was computed as $\frac{\sum (x-\bar{x})(y-\bar{y})}{(n-1)s_X s_Y}$; where x represents a vector with per-task speedup metric elements; y represents a vector with per-task credit-value elements; $\bar{x}$ represents the mean value of x; s_X represents the standard deviation of x, and n represents the length of x.

We have run 10 experiments using different sequences of 2000 tasks, and averaged the results. We have also calculated the average response times for each policy. Figures 5.5, 5.6, and Tables 5.1, 5.2, and 5.3 present partial results. Complete results can be found through Tables A.9−A.11 in Appendix A, Section A.3.

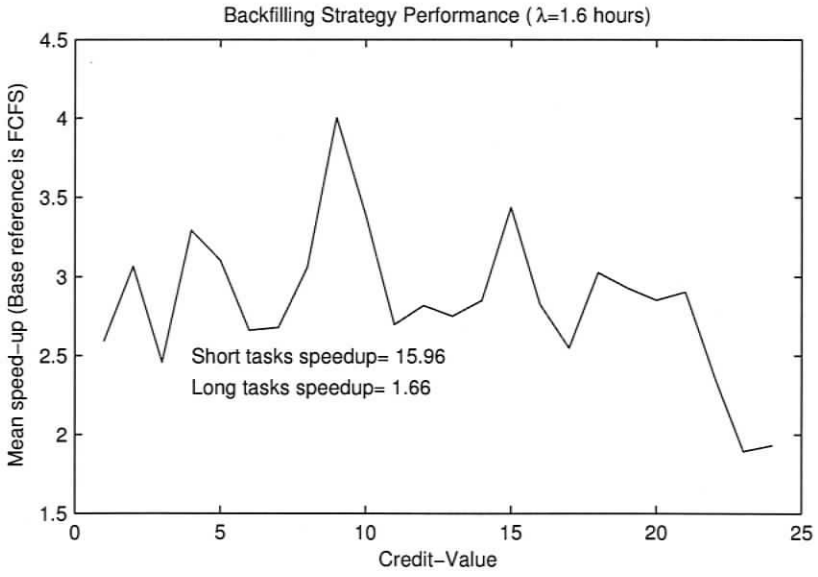

Figure 5.5. *Speedup performance of the BF strategy.*

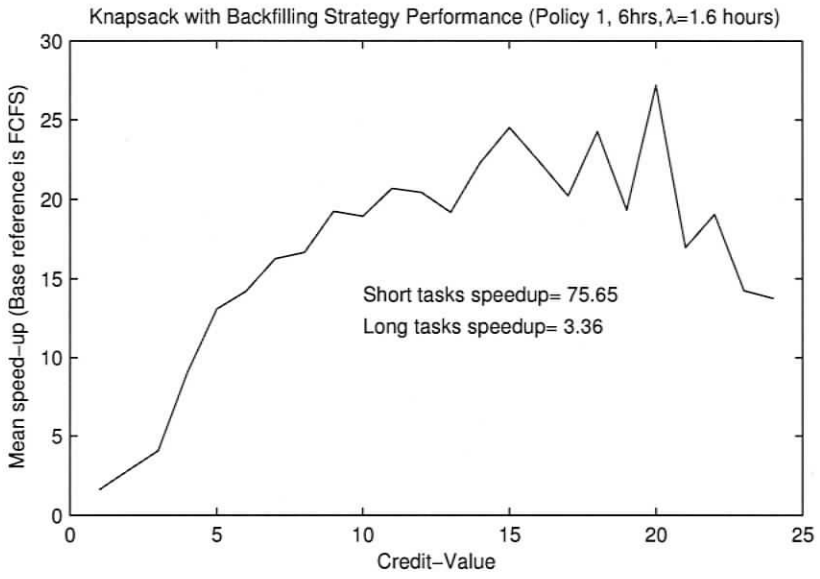

Figure 5.6. *Speedup performance of the KBA+BF strategy.*

Figures 5.5 and 5.6 plot the average speedup value as a function of the assigned credit-value. Both Figures correspond to a task mean inter-arrival time $\lambda = 1.6$h. Figure 5.5 presents the speedup of the Backfilling strategy, while Figure 5.6 presents the corresponding data for Policy 1, with a scheduling time slice of 6 hours.

Table 5.1 tabulates the correlation of the speedup (relative to the baseline of the FCFS policy) for the set of policies outlined earlier, including Gang scheduling and BF. The table rows correspond to different task inter-arrival times, as noted, while we have used different time slices for the policies, also as noted.

Regarding speedup distribution behaviour on a per-experiment basis, we observed that this is a long-not-heavy-tailed distribution. A Kolmogorov-Smirnov goodness-of-fit test for a number of well-known distributions, including weibull, lognormal, t location-scale, and extreme value, concludes that the observed data does not belong to any of the tested distributions. However, a visual inspection shows that perhaps a lognormal distribution could be used to partially describe the observed data (Figure 5.7). Nevertheless, this behaviour

Table 5.1. *Correlation of credit-value metric with speedup relative to FCFS*

λ	FCFS	BF	GANG (τ)		KBA (τ)		KBA+BF(τ)	
			6h	9h	6h	9h	6h	9h
	Ref.					**Policy 1**		
0h	-	0.00	-0.07	-0.07	0.28	0.30	0.27	0.29
1.6h	-	0.00	-0.05	-0.06	0.16	0.17	0.14	0.14
6.4h	-	0.00	-0.04	-0.03	-0.06	-0.05	-0.09	-0.09

λ	KBA (τ)		KBA+BF(τ)		KBA (τ)		KBA+BF(τ)	
	6h	9h	6h	9h	6h	9h	6h	9h
	Policy 2				**Policy 3**			
0h	0.18	0.20	0.18	0.19	0.18	0.20	0.18	0.20
1.6h	0.15	0.16	0.14	0.12	0.14	0.14	0.11	0.09
6.4h	-0.07	-0.06	-0.10	-0.10	-0.07	-0.06	-0.10	-0.10
	Policy 4				**Policy 5**			
0h	0.01	0.01	0.01	0.01	0.00	0.01	0.00	0.01
1.6h	0.02	0.02	0.02	0.02	0.01	0.01	0.01	0.01
6.4h	-0.08	-0.07	-0.10	-0.10	-0.08	-0.07	-0.10	-0.10

(however it might be classified) is consistent.

In order to test this consistency, normality tests were performed on the mean speedup values of some randomly selected strategies. Lilliefors tests for normality indicate that one cannot reject the hypothesis that the observed data has a normal distribution. That is, the consistency in the mean speedup behaviour supports the observation that knapsack strategies do provide a performance superior to other well-known allocation mechanisms. A brief discussion of these statistical evaluations is provided in Appendix A, Section A.3.1.

Table 5.2. *Strategies' ending times in month (28 days) units*

λ	FCFS	BF	GANG (τ)		KBA (τ)		KBA+BF(τ)	
			6h	9h	6h	9h	6h	9h
	Ref.					**Policy 1**		
0h	6.61	5.65	7.98	8.18	5.97	6.21	5.54	5.55
1.6h	6.65	5.76	6.28	6.40	5.97	6.18	5.63	5.64
6.4h	19.42	19.42	19.46	19.46	19.41	19.42	19.41	19.41

λ	KBA (τ)		KBA+BF(τ)		KBA (τ)		KBA+BF(τ)	
	6h	9h	6h	9h	6h	9h	6h	9h
	Policy 2				**Policy 3**			
0h	6.01	6.28	5.55	5.56	5.74	5.87	5.54	5.54
1.6h	6.02	6.26	5.63	5.64	5.83	5.94	5.62	5.61
6.4h	19.41	19.42	19.41	19.41	19.41	19.42	19.41	19.41
	Policy 4				**Policy 5**			
0h	5.99	6.11	5.65	5.65	5.83	5.93	5.61	5.62
1.6h	5.97	6.12	5.63	5.63	5.87	5.97	5.63	5.65
6.4h	19.41	19.42	19.41	19.41	19.41	19.42	19.41	19.41

5.8 Real-Time Considerations

For the simulated Grid scenario, the computational time needed to solve an instance of a knapsack problem is about 3 seconds when considering a mean interarrival time $\lambda = 0$. That is, we assume that all the tasks are already in the queue and thus are considered in the formulation of the knapsack problem. This time is related to an AMD 1.4 GHz Personal Computer, with 264 MB of RAM, and running Mandrake Linux OS, kernel 2.4.21-0.13mdk.

In this work, we assumed a task mean computational time of one day for short tasks and eight days for long tasks. In general, tasks in the Grid are assumed to be highly resource demanding and/or highly computational-time demanding, these tasks can run for hours or days. That is, the time spent in solving the knapsack ($\sim$3 sec) can be considered negligible

Table 5.3. *Mean speedup relative to FCFS*

λ	FCFS	BF	GANG (τ)		KBA (τ)		KBA+BF(τ)	
			6h	9h	6h	9h	6h	9h
	Ref.					**Policy 1**		
0h	1	1.41	2.01	1.80	3.58	3.34	3.72	3.49
1.6h	1	3.00	3.93	2.97	15.99	14.11	16.50	14.71
6.4h	1	1.02	0.84	0.79	1.01	0.95	1.19	1.19

λ	KBA (τ)		KBA+BF(τ)		KBA (τ)		KBA+BF(τ)	
	6h	9h	6h	9h	6h	9h	6h	9h
	Policy 2				**Policy 3**			
0h	10.74	9.36	11.02	9.72	10.76	9.59	10.93	9.81
1.6h	10.37	9.09	11.31	10.62	10.77	9.78	12.06	16.52
6.4h	1.00	0.94	1.18	1.17	1.00	0.94	1.19	1.18
	Policy 4				**Policy 5**			
0h	16.20	13.69	15.90	13.37	15.99	13.60	15.88	13.40
1.6h	6.89	5.97	8.25	7.98	6.85	6.10	8.16	8.19
6.4h	1.01	0.95	1.20	1.19	1.00	0.95	1.19	1.18

when compared to the time tasks run on the Grid.

5.9 Conclusion

As can be seen from Table 5.2, the task completion times of the knapsack policies are better than those of the first-come first-served and of the Gang scheduling. When the knapsack policy is combined with backfilling, it terminates earlier than all other policies (including backfilling). We also observe that, as the mean task inter-arrival time increases, all policies begin converging and, at very low offered loads, they perform identically. This is explained by the fact that at very long task inter-arrival times the computational resources are under utilized. Therefore, every task is allocated as soon as it arrives, and the completion time depends only on the arrival of the last task and its required computation time.

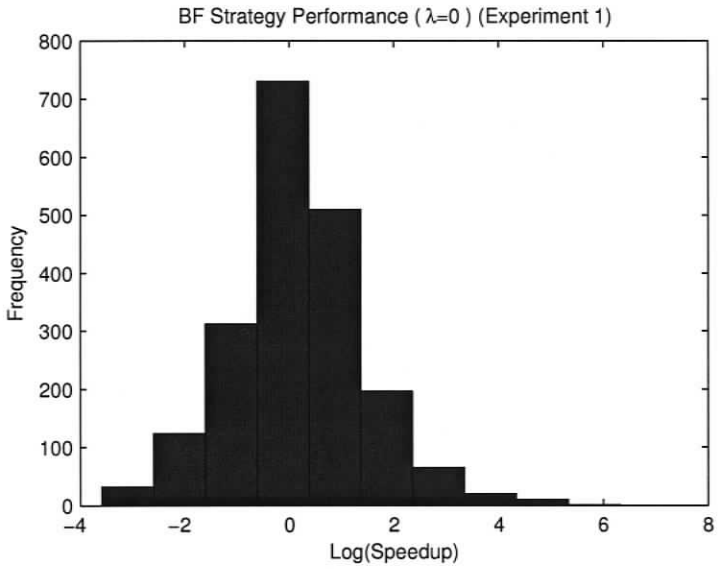

Figure 5.7. *Typical log(speedup) distribution behaviour.*

Table 5.3 shows the mean speedup of the various policies studied. All policies exhibit an improved average completion time as compared to FCFS. The higher speedups are exhibited by Policies 2 to 5, and this is explained by the fact that these policies include in their utilities components which enable preferential allocation for tasks that have waited for longer periods of time, or are close to termination, thus improving average completion times.

The advantages of the knapsack policies are evident in Table 5.1, which tabulates the correlation of the assigned task credit-value with the task's speedup as compared to the FCFS policy. A positive correlation is clear for Policies 1, 2 and 3, each of which includes in its utility function a factor corresponding to the task's assigned credit-value. These numerical results indicate that QoS metrics can be used effectively to allocate tasks preferentially without diminishing the overall efficiency of the scheduling policy. Note that the completion time is not affected; as a matter of fact, it is improved, as compared to the FCFS and the Gang policies. When the knapsack policies are combined with backfilling, they provide optimum completion times. Figures 5.5 and 5.6 provide further insight into

the ability of knapsack policies to complete high-valued tasks earlier. Figure 5.5 shows that there is no correlation between credit-value and the speedup of tasks when the backfilling strategy is used. On the other hand, Figure 5.6 depicts the behaviour of a combined knapsack and backfilling policy, where it is evident that high-valued tasks are experiencing a greater speedup, as compared to those of a lower value.

In this work, we have introduced QoS concepts into the resource allocation problem on a computational Grid, and showed that QoS-enabled policies allocate resources efficiently. However, QoS-enable policies by themselves are not sufficient to achieve efficient allocations, as can be observed from the numerical results when a credit-value-driven policy is implemented. Under this policy, the goal of the allocation strategy was to select the most valuable options for the tasks. All non-knapsack strategies had the opportunity to allocate the most valuable options first. Their allocation mechanism is a localized one (per-task). On the other hand, knapsack strategies are centralized mechanisms (groups of tasks), and allow for preferential allocation to highly-valued tasks with no detriment to the overall completion time of the submitted tasks as a group.

Chapter 6

Conclusions

Without resource allocation mechanisms, systems are only groups of elements, or resources, that cannot be used in an efficient manner because of a lack of organized and coordinated procedures. The aim of efficient resource allocation strategies is to optimize the delivery of value and the use of resources. To obtain such efficient mechanisms, however, one needs to (a) identify or design strategies to make resource allocation decisions; (b) measure how well each strategy contributes to the desired objective, and (c) find the best strategy, or combination of strategies.

6.1 Contributions

In this dissertation, we have explored resource allocation strategies, designed new ones, applied them to three systems, and evaluated their ability to achieve predefined goals. In particular, the following are the main contributions of this work.

1. We have proposed a heuristic approach to solve the multichoice multidimensional knapsack problem. The heuristic is based on the concept of pseudo-utility values. These values, along with a measure of the amount of resources consumed (resource coefficient ratios), were used in order to search for a feasible solution for the knapsack problem. Finally, the quality of this solution is improved through the use of the pseudo-utility values and the coefficient values of the objective function.

2. We have proposed an approach to formulate a mobile network resource allocation problem as a knapsack problem. Pseudo-utility values were used to establish an order by which a series of trials were attempted in order to evaluate whether mobile stations could be allocated channels.
3. We have proposed a resource allocation strategy on a Grid system. We introduced QoS concepts into the resource allocation problem. Utility functions were used to map the QoS metrics, and a knapsack formulation was used to express the allocation problem. The approach presented was used to allocate resources consistent with predefined policies.

In regard to the new heuristic for solving the multichoice multidimensional knapsack problem, this new heuristic produces better solutions than those of other heuristics. Numerical results, as well as a statistical analysis of these results, support this observation. Also, this new heuristic has the quality of being able to find feasible solutions where other heuristics fail; that is, it has a low rate of failure.

In regard to the performance of the proposed knapsack-based allocation strategies, numerical results and a statistical analysis of these results indicate that these knapsack-based allocation strategies have a superior performance than that of other allocation strategies.

The higher computational time required by the new heuristic and by the proposed knapsack-based allocation strategies does not diminish their potential applicability since this time falls within the real-time requirements of the systems studied.

6.2 Future Work

As we have shown in this dissertation, there is the possibility of improvement in any heuristic. There is also the possibility of using a strategy to cover new and more complex formulations. In the following sections, we present ideas for future work.

6.2.1 Computation of Pseudo-Utility Values

As mentioned in Chapter 3, there are several heuristics available to compute the pseudo-utility values. The computation of improved pseudo values will result in a superior performance of the resource allocation strategies. Thus heuristics to compute these improved values are sought.

6.2.2 Construction of Feasible Solutions

According to the strategy presented in Chapter 3, when constructing a feasible solution, information about the actual or perceived value of the variables as a means of selecting them is irrelevant; what matters is the amount of resources the variables consume and the amount of resources available. Thus, better mechanisms to construct feasible solutions are needed (for example, in situations where the selection of certain variables must be made).

6.2.3 Third Generation Cellular Network Systems

Although the procedure presented in Chapter 4 was done with FDMA and TDMA networks in mind, we think the strategy can be applied, with proper modifications, to other Third Generation Cellular Network Systems. The most widely used Third Generation radio interfaces are based on code-division multiple access, and power control is one of the mechanism used to regulate interference in the system.

6.2.4 Grid Computing Systems

The work presented in Chapter 5 assumed the use of a single resource (processors). If a more realistic evaluation performance of allocation strategies is desirable, then the inclusion of a variety of resources, a resource monitoring mechanism, and a task recovery mechanism, need to be considered. Along with the inclusion of a variety of resources, how these resources are valued by the domains they belong to should be taken into account; that

is, QoS metrics that describe these domain values should be considered in the allocation strategy.

Bibliography

[1] Astrogrid Project. website:. http://www.astrogrid.org/

[2] Astrophysical Virtual Observatory. website:. http://www.euro-vo.org/

[3] The DataGrid Project. website:. http://www.edg.org/

[4] International Virtual Observatory Alliance. website:. http://www.ivoa.net/

[5] National virtual observatory. website:. http://www.us-vo.org/

[6] (2004) Distribuited Aircraft Maintenance Environment (DAME). website:. http://www.cs.york.ac.uk/dame/

[7] (2004) Enabling Grids for E-science in Europe (EGEE). website:. http://public.eu-egee.org/

[8] (2004) Neesgrid. website:. http://it.nees.org/

[9] A. Antoniou and W.-S. Lu, Eds., *Optimization: Methods, Algorithms, and Applications*. Kluwer Academic, 2005.

[10] E. Balas and C. Martin, "Pivot and complement-a heuristic for 0-1 programming," *Management Science*, vol. 26, pp. 86–96, Jan. 1980.

[11] H. Casanova, "Simgrid: A toolkit for the simulation of application scheduling," in *Proc. IEEE Symposium on Cluster Computing and the Grid (CCGrid'01)*, May 2001, pp. 430–437.

[12] L. Chen, "The utility model applied to layer-coded sources," Master's thesis, Department of Computer Science. University of Victoria, Canada, 1998.

[13] P. Chu and J. Beasley, "A genetic algorithm for the multidimensional knapsack problem," *Journal of Heuristics*, vol. 4, pp. 63–86, June 1998.

[14] J. Czyzyk, S. Mehrotra, M. Wagner, and S.Wright, "PCx user guide (version 1.1)," Optimization Technology Center, USA, Tech. Rep. OTC, Nov. 1997.

[15] G. Dantzig, "Discrete-variable extremum problems," *Operations Research*, vol. 5, pp. 266–277, 1957.

[16] V. H. M. Donald, "Advance mobile phone service: The cellular concept," *The Bell System Technical Journal*, vol. 58, pp. 15–41, Jan. 1979.

[17] H. Everet, "Generalized lagrange multiplier method for solving problems of optimum allocation of resources," *Operations Research*, vol. 11, pp. 399–417, May 1963.

[18] I. Foster and C. Kesselman, Eds., *The Grid: Blueprint for a New Computing Infrastructure*, ser. Grid Computing. Morgan-Kaufmann, 1998.

[19] I. Foster, C. Kesselman, J. Nick, and S. Tuecke, "The physiology of the Grid: An Open Grid Services Architecture for distributed systems integration," Global Grid Forum, Open Grid Service Infrastructure WG, 2002.

[20] M. Frodigh, "Optimum dynamic channel allocation in certain street microcellular radio systems," in *Proc. IEEE Vehicular Technology Conference*, May 1992, pp. 658–661.

[21] M. Garey and D. Johnson, "Complexity results for multiprocessor scheduling under resource constraints," *SIAM Journal on Computing*, vol. 4, pp. 397–411, 1975.

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

[23] B. Gavish and H. Pirkul, "Efficient algorithm for solving the multiconstraint zero-one knapsack problem to optimality," *Mathematical Programming*, vol. 31, pp. 78–105, 1985.

[24] J. Gomoluch and M. Schroeder, "Market-based resource allocation for grid computing," in *Middleware 2003 Workshops*, Rio de Janeiro, Brazil, June 2004, pp. 211–218.

[25] S. A. Grandhi, R. D. Yates, and D. J. Goodman, "Resource allocation for cellular radio systems," *IEEE Trans. Veh. Technol.*, vol. 46, pp. 581–587, Aug. 1997.

[26] S. A. Grandhi, J. Zander, and R. D. Yates, "Constrained power control," *International Journal of Wireless Personal Communications (Kluwer)*, vol. 1, 1995.

[27] S. Grandhi and J. Zander, "Constrained power control in cellular radio systems," in *Proc. IEEE Vehicular Technology Conference*, June 1994, pp. 824–828.

[28] O. Ibarra and C. Kim, "Fast approximation algorithms for the knapsack and sum of subset problems," *Journal of ACM*, vol. 22, pp. 463–468, 1975.

[29] J. Ji and Y. Ye, "A complexity analysis for interior-point algorithms based on karmarkar's potential function," *SIAM J. Optimization*, vol. 4, pp. 512–520, Aug. 1994.

[30] I. Katzela and M. Naghshineh, "Channel assignment schemes for cellular mobile telecommunication systems: A comprehensive survey," *IEEE Personal Commun.*, vol. 3, no. 3, pp. 10–31, 1996.

[31] S. Khan, K. Li, E. Manning, and M. Akbar, "Solving the knapsack problem for adaptive multimedia system," *Studia Informatica*, 2002.

[32] S. Khan, "Quality adaptation in a multisession multimedia system: Model, algorithms and architecture," Ph.D. dissertation, Department of Electrical and Computer Engineering. University of Victoria, Canada, May 1998.

[33] D. E. Knuth, *The Art of Computer Programming*, ser. Seminumerical Algorithms. Addison-Wesley Publishing Company, 1969, vol. 2.

[34] E. Lawler, *Combinatorial Optimization: Networks and Matroids*. New York, USA: Holt, Rinehart and Winston, 1976.

[35] C. Lee, J. Lehoczky, D. Siewiorek, R. Rajkumar, and J. Hansen, "A scalable solution to the multi-resource QoS problem," in *Proc. IEEE Real-Time Systems Symposium*, Dec. 1999, pp. 315 –326.

[36] C. Lee, J. Lehoezky, R. Rajkumar, and D. Siewiorek, "On quality of service optimization with discrete QoS options," in *Proc. IEEE Real-Time Technology and Applications Symposium*, June 1999, pp. 276–286.

[37] D. Lifka, "The anl/ibm sp scheduling system," in *Job Scheduling Strategies for Parallel Processing*, ser. Lectures Notes Computer Sciences. Springer-Verlag, 1995, vol. 949, pp. 295–303.

[38] R. Loulou and E. Michaelides, "New greedy-like heuristics for the multidimensional 0-1 knapsack problem," *Operations Research*, vol. 27, pp. 1101–1114, Nov. 1979.

[39] I. Lustig, R. Marsten, and D. Shanno, "On implementing mehrotra's predictor-corrector interior-point method for linear programming," *SIAM J. Optimization*, vol. 2, pp. 435–449, Aug. 1992.

[40] M. Magazine and O.Oguz, "A heuristic algorithm for the multidimensional zero-one knapsack problem," *European Journal of Operational Research*, vol. 16, pp. 319–326, June 1984.

[41] S. Martello and P. Toth, *Knapsack Problems: Algorithms and Computer Implementations*. John Wiley & Sons, 1990.

[42] M. Moser, "Declarative scheduling for optimally graceful QoS degradation," in *Proc. IEEE International Conference on Multimedia Computing and Systems (ICMCS)*, 1996, pp. 86–93.

[43] M. Moser, D. Jokanović, and N. Shiratori, "An algorithm for the multidimensional multiple-choice knapsack problem," *IEICE Trans. Fundamentals*, vol. E80-A, pp. 582–589, Mar. 1997.

[44] G. Mounie, C. Rapine, and D. Trystram, "Efficient approximation algorithms for scheduling malleable tasks," in *Proc. of the 11th Annual ACM Symposium on Parallel Algorithms and Architecture*, 1999, pp. pp. 23–32.

[45] J. K. Ousterhout, "Scheduling techniques for concurrent systems," in *3rd Intl. Conf. Distributed Comput. Syst.*, Oct. 1982, pp. 22–30.

[46] R. Parra-Hernandez and N. Dimopoulos, "Heuristic approaches for solving the multi-

dimensional knapsack problem (mkp)," *WSEAS Transactions on Systems*, vol. 1, pp. 248–253, Apr. 2002.

[47] R. Parra-Hernandez and N. Dimopoulos, "A new heuristic for solving the multi-choice multidimensional knapsack problem," *IEEE Trans. Syst., Man, Cybern. A*, vol. 35, pp. 708–717, Sept. 2005.

[48] R. Parra-Hernandez and N. J. Dimopoulos, "Channel resource allocation/reallocation in cellular communication and linear programming," in *Proc. IEEE International Conference on Systems, Man & Cybernetics*, Oct. 2003, pp. 2983–2989.

[49] R. Parra-Hernandez and N. J. Dimopoulos, "On the performance of the ant colony system for solving the multidimensional knapsack problem," in *Proc. IEEE Pacific Rim Conference on Communications*, 2003, pp. 338–341.

[50] R. Parra-Hernandez, D. Vanderster, and N. Dimopoulos, "Resource management and knapsack formulations on the grid," in *IEEE/ACM International Workshop on Grid Computing*, Nov. 2004, pp. 94–101.

[51] H. Pirkul, "A heuristic solution procedure for the multiconstraint zero-one knapsack problem," *Naval Research Logistics*, vol. 34, pp. 161–172, 1987.

[52] D. Pisinger, "Algorithms for knapsack problems," Ph.D. dissertation, Department of Computer Science. University of Copenhagen, Denmark, Feb. 1995.

[53] P.Wolfe, "A duality theorem for nonlinear programming," *Quarterly of Applied Mathematics*, vol. 19, pp. 239–244, 1961.

[54] S. Sahni, "Approximate algorithms for the 0-1 knapsack problem," *Journal of ACM*, vol. 22, pp. 115–124, 1975.

[55] A. Szalay and J. Gray, "The world-wide telescope," *Science*, vol. 293, pp. 2037–2040, 2001.

[56] Y. Toyoda, "A simplified algorithm for obtaining approximate solution to zero-one programming problems," *Management Science*, vol. 21, pp. 1417–1427, Aug. 1975.

[57] S. Tuecke, K. Czajkowski, I. Foster, J. Frey, S. Graham, C. Kesselman, T. Maguire, T. Sandholm, D. Snelling, and P. Vanderbilt. (2003) Open Grid Services Infrastructure. Internet draft, version 1.0. Global Grid Forum. http://www.ggf.org/documents/Drafts

[58] D. Warner and J. Prawda, "A mathematical programming model for scheduling nursing personnel in a hospital," *Management Science, Application Series Part 1*, vol. 19, pp. 411–422, Dec. 1972.

[59] R. Watson, "The optimal admission & adaptation of service level agreements in

packet networks : Applying the utility model," Master's thesis, University of Victoria, Canada, 2001.

[60] J. F. Whitehead, "Performance and capacity of distributed dynamic channel assignment and power control in shadow fading," in *Proc. IEEE International Conference on Communications*, vol. 2, Geneva, Switzerland, May 1993, pp. 910–914.

[61] Y. Ye, *Interior Point Algorithms: Theory and Analysis*. Wiley-Interscience Series in Discrete Mathematics and Optimization, 1997.

[62] Y. Zhang, "Solving large-scale linear programming by interior-point methods under the matlab environment," University of Maryland Baltimore County, USA, Tech. Rep. TR96-01, Feb. 1996.

Appendix A

Numerical Results

This Appendix contains the complete numerical results from the evaluation of the heuristics in Chapter 3 and the allocation strategies in Chapter 5. Section A.1 relates to Chapter 3, and contains the results when the number of variables in each group equals five. Section A.1.1 also relates to Chapter 3, and contains additional numerical results for groups of variables of "varying" size. Finally, Section A.3, which relates to Chapter 5, contains the results from the evaluation of the FCFS, BF, Gang, KBA, and KBA with BF allocation strategies.

A.1 MMKP Heuristics' Performance

As mentioned in Chapter 3, the OR-Library test problems evaluated are contained in the files mknapcb7, mknapcb8, and mknapcb9. The first problems in each file are the most difficult to solve, and they were used for testing MOSER, HEU, and Hmmkp. Only the first ten problems from each of the above-mentioned files were tested.

The available resources, coefficients b_k, were decreased by a factor f. First, no decrement is performed ($f = 1.0$); then, a decrement of 10% ($f = 0.9$) is allowed; finally, a decrement is carried out, so that at least one of the heuristics fails to find a feasible solution. Because of the decrement factor f used, a total of 90 problems were evaluated. The results are given in Tables A.1–A.3.

Table A.1. *Performance comparison of MOSER, HEU and Hmmkp*

MMKP parameters: $n = 20$; $l = 5$; $m = 30$							
Series	factor	Solution found			Execution time (msec)		
mknapcb7	f	MOSER	HEU	Hmmkp	MOSER	HEU	Hmmkp
0	1	18 884	18 884	18 884	0.1	1.8	33
1	1	18 446	17 653	18 632	1.4	2.9	36
2	1	17 993	17 879	17 994	0.1	1.4	36
3	1	18 209	18 209	18 209	0.1	1.7	36
4	1	18 732	18 759	18 759	0.1	1.5	35
5	1	19 066	19 066	19 066	0.1	1.8	35
6	1	18 303	18 252	18 303	0.1	1.7	35
7	1	18 377	18 377	18 377	0.1	1.9	31
8	1	18 898	18 851	19 081	0.3	1.6	37
9	1	17 751	17 432	17 751	0.1	1.6	36
0	0.9	15 183	17 560	17 510	1.3	1.4	31
1	0.9	16 439	17 335	17 948	3.9	1.3	36
2	0.9	16 229	16 907	17 049	1.6	1.1	38
3	0.9	15 901	17 341	17 833	1.5	1.3	38
4	0.9	16 724	18 127	17 315	1.4	1.2	38
5	0.9	17 754	17 824	18 318	1.6	1	35
6	0.9	15 774	17 307	18 045	1.3	1.4	37
7	0.9	16 301	17 117	17 301	1.6	1.4	35
8	0.9	16 583	18 213	17 563	1.7	1.2	35
9	0.9	16 146	16 511	16 691	1.6	1.3	38
0	0.84	*	*	15 617	1.2	0.1	44
1	0.84	*	15 885	15 951	1.2	0.8	42
2	0.84	*	*	14 080	1.2	0.1	39
3	0.84	*	*	14 876	1.1	0.1	40
4	0.84	*	*	15 595	1.2	0.1	40
5	0.84	*	*	15 791	1.2	0.1	38
6	0.84	*	*	15 484	1.1	0.1	38
7	0.84	*	*	14 963	1.1	0.1	38
8	0.84	*	*	16 160	1.2	0.2	41
9	0.84	*	15 437	13 098	1.3	0.6	38

* no feasible solution found

Table A.2. *Performance comparison of MOSER, HEU and Hmmkp*

MMKP parameters: $n = 50$; $l = 5$; $m = 30$							
Series	factor	Solution found			Execution time (msec)		
mknapcb8	f	MOSER	HEU	Hmmkp	MOSER	HEU	Hmmkp
0	1	46 081	46 081	46 081	0.1	10.9	79
1	1	47 514	47 514	47 514	0.1	9.4	80
2	1	45 977	45 977	45 977	0.1	11.2	79
3	1	45 961	45 961	45 961	0.1	10.9	80
4	1	45 576	45 292	45 685	3	9.4	93
5	1	46 685	46 685	46 685	0.1	11.7	80
6	1	46 529	46 529	46 529	0.1	9.8	74
7	1	45 810	45 653	45 810	0.1	10.9	79
8	1	47 232	47 232	47 232	0.1	10	80
9	1	46 296	46 296	46 296	0.1	10.4	79
0	0.9	44 267	45 535	45 493	9.3	10	119
1	0.9	45 963	47 095	47 130	10	9.1	101
2	0.9	42 167	43 810	45 390	10.2	11.3	99
3	0.9	44 732	45 707	45 810	11.3	10.7	95
4	0.9	43 997	44 669	45 270	10.6	9.1	102
5	0.9	46 579	46 447	46 611	0.3	10.6	93
6	0.9	45 743	46 224	46 064	11.4	10.3	94
7	0.9	44 038	44 644	45 450	9.7	10.1	108
8	0.9	46 616	46 939	47 156	9.6	9.8	94
9	0.9	45 195	45 411	45 859	9.9	10.1	101
0	0.8	*	41 308	38 523	6.8	6.4	117
1	0.8	*	*	41 185	6.7	0.2	118
2	0.8	*	*	41 259	8.6	0.2	110
3	0.8	*	*	40 066	6.7	0.2	111
4	0.8	*	*	38 262	6.8	0.2	109
5	0.8	*	*	39 670	8.6	0.2	105
6	0.8	*	*	38 547	8.4	0.2	99
7	0.8	*	*	39 445	6.9	0.2	105
8	0.8	*	*	40 954	6.7	0.2	109
9	0.8	*	40 677	39 834	8	4.8	101

* no feasible solution found

Table A.3. *Performance comparison of MOSER, HEU and Hmmkp*

MMKP parameters: $n = 100$; $l = 5$; $m = 30$							
Series	factor	Solution found			Execution time (msec)		
mknapcb9	f	MOSER	HEU	Hmmkp	MOSER	HEU	Hmmkp
0	1	92 148	92 148	92 148	0.2	37.3	176
1	1	92 371	92 371	92 371	0.2	40.6	170
2	1	93 408	93 408	93 408	0.2	40.9	171
3	1	91 878	91 878	91 878	0.2	45.5	168
4	1	93 367	93 367	93 367	0.2	40.1	166
5	1	91 633	91 633	91 633	0.2	37.8	170
6	1	91 494	91 494	91 494	0.2	39.2	188
7	1	91 799	91 799	91 799	0.2	39.3	170
8	1	93 151	93 151	93 151	0.2	39.1	173
9	1	93 586	93 586	93 586	0.2	41.4	174
0	0.9	90 602	91 879	92 021	41.7	38.3	218
1	0.9	92 371	92 371	92 371	0.2	41.2	191
2	0.9	93 396	93 294	93 396	0.5	40.5	206
3	0.9	90 137	91 716	91 815	38.1	42.7	195
4	0.9	92 682	93 150	93 317	38.8	39.8	200
5	0.9	91 002	91 319	91 547	44.7	37.4	200
6	0.9	90 927	91 322	91 480	1.4	39.2	180
7	0.9	90 861	91 284	91 672	41.5	39.2	198
8	0.9	93 149	93 034	93 149	0.5	41.3	185
9	0.9	92 562	93 385	93 528	41.3	42.2	216
0	0.75	*	*	74 927	27.8	0.2	261
1	0.75	*	*	73 570	27.7	0.2	256
2	0.75	*	*	74 739	27.5	0.3	251
3	0.75	*	*	69 813	27.8	0.2	250
4	0.75	*	*	74 323	28.2	0.2	235
5	0.75	*	*	74 303	27.5	0.3	270
6	0.75	*	*	72 018	27.8	0.3	261
7	0.75	*	*	73 777	27.6	0.2	260
8	0.75	*	*	74 376	30	0.2	252
9	0.75	*	*	73 496	27.7	0.2	247

* no feasible solution found

A.1.1 More Numerical Results on the MMKP Heuristics

We have also tested the heuristics on further synthetic MMKP problems; these problems include groups of variables of varying size. Because we do not possess a code implementation of MOSER or HEU that will allow us to evaluate groups of different sizes, we decided to *include* dummy variables to vary the sizes of the groups; that is, the synthetic problems are derived from the standard MKP test problems. The dummy variables were included as explained below.

- A sequence of random numbers (with values varying from $0 - 3$) was generated. The length of the sequence equals 112. For the mknapcb7 file, the first 23 random numbers were used. For the mknapcb8, the first 56 random numbers were used, and for the mknapcb9 the complete sequence was used.
- Groups from the files were created according to the number of random numbers used in each file. That is, 23 groups were created from file mknapcb7, 56 groups were created from mknapcb8, and 112 groups were created from mknapcb9. Each group relates to a random number.
- Groups have a fixed number of variables equal to six. The number of dummy variables in a group equals the value of its random number; the rest of the variables in the group (in order to obtain a group of six variable) were taken from corresponding files. For example, let us say that we are creating the first group from the mknapcb7 file, and that the first random number in the sequence is two. The first four variables are taken from file mknapcb7, and then two (the value of its random number) dummy variables are added in order to create the first group. If we assume that the second random number equals three, the next three variables from file mknapcb7 are selected, and three dummy variables are added to create the second group, and so on.

A dummy variable is defined as a variable that consumes 100 units of each type of resource and has a value of zero (coefficient in the objective function). That is, the heuristics

should try to avoid the selection of variables that do not contribute to an increase in the value of the objective function. However, such dummy variables are attractive when there is difficulty in finding a feasible solution, since the amount of resources consumed by the dummy variables is less than the average amount consumed by the real variables (500 units of resources, see [13]). The solution to this type of problem shows the same behaviour as previously observed: when compared with MOSER and HEU, Hmmkp found better quality solutions.

Random number sequence used:

0,	2,	2,	3,	0,	0,	3,	3,	1,	3,	3,	2,	0,	0,	3,	3,	3,	1,	2,	1,
1,	2,	0,	0,	3,	2,	0,	0,	2,	0,	2,	0,	3,	3,	0,	1,	3,	0,	0,	0,
1,	1,	1,	3,	0,	3,	3,	2,	0,	3,	2,	2,	2,	3,	1,	2,	0,	2,	1,	1,
1,	0,	1,	3,	0,	2,	0,	2,	3,	2,	1,	0,	1,	3,	2,	0,	3,	3,	2,	2,
1,	3,	0,	2,	3,	0,	1,	1,	1,	3,	2,	2,	2,	3,	1,	1,	2,	1,	2,	0,
0,	0,	3,	1,	0,	1,	2,	3,	2,	2,	3,	3								

Table A.4. *Performance comparison of MOSER, HEU and Hmmkp*

MMKP parameters: $n = 23$; $l = 6$; $m = 30$				
Series	factor	Solution found		
mknapcb7	f	MOSER	HEU	Hmmkp
0	1	19372	18790	20029
1	1	18317	18441	19259
2	1	16323	18062	18633
3	1	16952	19191	19556
4	1	18480	17587	19020
5	1	19014	19868	20399
6	1	18139	17146	18625
7	1	18219	19774	19221
8	1	18979	20692	19889
9	1	17656	18224	19179
0	0.9	15513	17913	18355
1	0.9	15107	16061	16547
2	0.9	14102	16059	15887
3	0.9	16051	16890	16973
4	0.9	16222	15878	17815
5	0.9	16229	17888	18181
6	0.9	15786	15190	16715
7	0.9	15835	17314	16836
8	0.9	17319	17795	17981
9	0.9	15430	16320	17087
0	0.84	15031	16327	16211
1	0.84	14057	15614	14642
2	0.84	13712	14612	14960
3	0.84	13933	15827	15979
4	0.84	14828	14074	16120
5	0.84	15522	15944	15875
6	0.84	14366	13502	15945
7	0.84	13829	15703	15494
8	0.84	15150	16754	15868
9	0.84	14504	14664	15290
Average		16132.56	16936.80	17419.03

Table A.5. *Performance comparison of MOSER, HEU and Hmmkp*

MMKP parameters: $n = 56$; $l = 6$; $m = 30$				
Series	factor	Solution found		
mknapcb8	f	MOSER	HEU	Hmmkp
0	1	48740	50335	50480
1	1	51718	52562	52821
2	1	49690	49350	50301
3	1	49697	49153	50565
4	1	46924	45505	49318
5	1	50277	50497	50872
6	1	50452	51289	51570
7	1	49389	48359	50643
8	1	51293	49842	51648
9	1	49477	49219	50579
0	0.9	42181	46875	46712
1	0.9	42661	47743	49903
2	0.9	42070	44343	47330
3	0.9	45373	43633	47734
4	0.9	41159	43899	46202
5	0.9	40341	46185	47300
6	0.9	44608	46442	47376
7	0.9	43585	43525	47140
8	0.9	43972	45173	49125
9	0.9	43772	43882	45818
0	0.8	34546	41011	41268
1	0.8	36589	42148	42818
2	0.8	35739	38715	41705
3	0.8	34676	38441	41458
4	0.8	35994	39632	40868
5	0.8	35142	40219	42289
6	0.8	34688	41712	38718
7	0.8	36035	38034	40362
8	0.8	35156	38372	41997
9	0.8	35845	36856	39954
Average		42726.3	44765.03	46495.8

Table A.6. *Performance comparison of MOSER, HEU and Hmmkp*

MMKP parameters: $n = 112$; $l = 6$; $m = 30$				
Series	factor	Solution found		
mknapcb9	f	MOSER	HEU	Hmmkp
0	1	101401	101329	101404
1	1	101413	101403	101419
2	1	102932	103226	103889
3	1	100295	101715	101858
4	1	103494	103494	103494
5	1	101609	101571	101628
6	1	100826	98607	101549
7	1	101225	101106	101226
8	1	102529	102478	102514
9	1	101464	102788	103680
0	0.9	89547	96274	95894
1	0.9	91047	96740	96276
2	0.9	88818	97599	100052
3	0.9	91747	96596	96154
4	0.9	92170	93556	98365
5	0.9	88579	93565	97307
6	0.9	86604	88496	96436
7	0.9	90509	96674	96179
8	0.9	90902	94432	98761
9	0.9	85937	95175	98081
0	0.75	65771	79510	79466
1	0.75	68836	79801	78106
2	0.75	70126	80108	80437
3	0.75	69203	79345	77943
4	0.75	69222	71461	80626
5	0.75	68924	77983	78259
6	0.75	68544	69312	77710
7	0.75	67232	79068	77247
8	0.75	64166	76178	76582
9	0.75	70977	77044	80336
Average		86534.96	91221.13	92762.60

A.1.2 On the Performance of the Hmmkp Solutions

In this Appendix, we have reported the performance of the Hmmkp, HEU and MOSER heuristics on a number of problems. This performance is based on the values of the solutions found[1]. Further insight can be provided by observing the cases where the values of the Hmmkp solutions are better, equal or worse than those of other heuristics. That is, one can compare these observed values against the expected values when assuming a particular behaviour in the Hmmkp heuristic. If the probability of these observed values is very low, then one can reject the assumed behaviour.

We have compared the performance of the Hmmkp heuristic against that of a "combined" HEU and MOSER heuristic. First, from the HEU's solution and the MOSER's solution of a particular problem, we select the solution with the largest value. Second, we compare the value of this solution against that of the Hmmkp's solution and then we select the one with the largest value. The heuristic associated with this solution is considered to be the best heuristic for that particular problem. For instance, for the corresponding problem in the first row of Table 3.1, we observe that the combined heuristic (HEU and MOSER) is the best one (108-HEU vs. 105-Hmmkp). On the other hand, for the corresponding problem in the second row of Table A.1, we observe that the Hmmkp heuristic is the best one (18632-Hmmkp vs. 18446-MOSER).

Table A.7 shows the total number of instances where the above described behaviour occurs. We have only considered instances where all heuristics find a feasible solution. The first column names the particular table we are referring to. Column "Hmmkp^{+}" numbers the cases where the Hmmkp heuristic is better than the combined heuristic (HEU and MOSER). The "Combined^{+}" column numbers the cases where the combined heuristic is better than the Hmmkp heuristic. Column "Equal" numbers the cases where the value of the

[1]The solution of a particular problem selects the variables that satisfies the constraints of this problem. The value of this solution is the value obtained when one evaluates the corresponding objective function based on the variables selected by this solution.

Hmmkp's solution equals that of the combined heuristic. Column "Eliminated" numbers the instances not considered in the analysis. These occur because the combine heuristic has failed in finding a solution for these particular instances; that is, either HEU or MOSER or both heuristics were unable to solve these problems. Column "Total" lists the total number of problems in a particular table.

Table A.7. *Heuristics' performance*

Table	Performance				
	Hmmkp$^+$	Combined$^+$	Equal	Eliminated	Total
Table 3.1	15	2	5	2	24
Table A.1	10	3	7	10	30
Table A.2	9	2	9	10	30
Table A.3	7	0	13	10	30
Table A.4	21	9	0	0	30
Table A.5	28	2	0	0	30
Table A.6	20	9	1	0	30
$\sum$	110	27	35		

From Table A.7, we observe that there is a total of 110 instances when Hmmkp is better, a total of 27 instances when the combined heuristic is better, and a total of 35 instances when the performance is the same. We performed a chi-square test on the total numbers of observed instances in order to evaluate whether these numbers can be considered random numbers from an assumed behaviour. This behaviour has three categories: the Hmmkp heuristic produces better results, the combined heuristic produces better results, the heuristics produce the same results. The chi-square statistic V is given by

$$V = \sum_{1 \leq s \leq k} \frac{(Y_s - np_s)^2}{np_s}$$

where n represents the total number of observations ($n = 110 + 27 + 35 = 172$); k represents the number of categories considered ($k = 3$); p_s represents the probability that each observation falls into category s ($s = 1, 2, 3$); Y_s represents the number of observations

that actually do fall into category s. That is,

$$V = \frac{(110 - 172p_1)^2}{172p_1} + \frac{(27 - 172p_2)^2}{172p_2} + \frac{(35 - 172p_3)^2}{172p_3}$$

One can compute or propose probability values p_s for each category. These values should result in a chi-square statistic value X that is neither too low nor too high. The probability of the statistic $V > X$ should not be considered to be significantly high or significantly low. A statistic's probability value between 75% and 50% can be considered adequate [33].

A search for suitable p_s values shows that when $p_1 = 0.63$, $p_2 = 0.14$, and $p_3 = 0.23$, we have $V = 0.9$. Under a chi-square distribution with 2 degrees of freedom, one will have $V > 0.9$ approximately 60% of the time. That is, for these probability values p_s, the total numbers of observed instances can be considered to be random numbers of the assumed behaviour. This behaviour indicates that the Hmmkp heuristic produces better solutions 63% of the time ($p_1 = 0.63$) as compared to only 14% of the time when the combined heuristic produces better results ($p_2 = 0.14$). If one compares the performance of the Hmmkp heuristic against the individual performance of the other heuristics (instead of a combined heuristic), then the Hmmkp heuristic produces better results more than 63% of the time.

Acknowledgment

We are indebted to Dr. Shahadatullah Khan for providing us with the code of the HEU and MOSER heuristics, as well as the code of an exact algorithm to solve the MMKP. The exact algorithm was used for solving to optimality the test problems used in Table 3.1.

A.2 On the Performance of the Rellocation Strategy

As mentioned in Chapter 4, Section 4.9, we explored whether the performance observed in the reallocation case is consistent. For a given mean traffic λ and for a given number of channels, we divided the 2000 groups tested into subsets of 100. That is, 20 subsets, or samples, were created. For each subset we computed the probability of rejection as given by Equation 4.4. Then, we performed Lilliefors tests[2] on these rejection values in order to test for consistency.

A total of ten Lilliefors tests were carried out. Column "LSTAT" in Table A.8, shows the statistic values for the different values of λ and for the different number of channels. These statistics are compared against the Critical Value (CV) of the test. Column "H" shows the result of the hypothesis test. The result H equals 1 if one can reject the hypothesis that the observed data has a normal distribution, or 0 if one cannot reject that hypothesis. In eight of these tests, the results indicate that one cannot reject this hypothesis. These include cases where λ equals 0.30, 0.35 and 0.40 (i.e., at the highest load traffic) and when using five and ten channels. That is, at high traffic loads and when reallocation is applied, the performance obtained from the strategy proposed can be considered consistent.

Table A.8. *Standard-reallocation-DDRA scheme*

Lilliefors tests results ($\alpha = 0.05$)				
λ	5 Channels		10 Channels	
	LSTAT	H	LSTAT	H
0.20	0.187	0	0.367	1
0.25	0.120	0	0.247	1
0.30	0.111	0	0.132	0
0.35	0.137	0	0.108	0
0.40	0.130	0	0.099	0
CV=0.19; 20 samples per test				

[2] A Lilliefors test evaluates the hypothesis that observed data has a normal distribution with unspecified mean and variance, against the alternative that the observed data does not have a normal distribution.

A.3 Grid Allocation Strategies' Numerical Results

This section contains the complete numerical results from the evaluation of the allocation strategies tested in Chapter 5. As mentioned in that Chapter, each experiment comprised a sequence of 2000 tasks, which were allocated according to the policies discussed in section 5.6. The utility values for a task's options were calculated as described in section 5.5.1. For each sequence, we recorded the completion time as well as the submission s_i and actual completion C_i times for each task i, and calculated their response times as $R_i = C_i - s_i$. We have considered the performance of the FCFS policy as the baseline to which the performance of the tested policies is compared. That is to say, we have calculated the speedup achieved on a per task basis by dividing the response F_i of task i under the FCFS strategy with the response R_i achieved by each of the proposed policies. We have run 10 experiments using different sequences of 2000 tasks, and averaged the results. We have used the per task speedup metrics and correlated these to the per task assigned credit-value metric.

Values in Tables A.9 and A.10 are relative to the baseline of the FCFS policy. Table A.9 tabulates the correlation of the speedup for the set of policies outlined in Chapter 5. The rows of the table correspond to different task inter-arrival times, while for each policy we have used three different time slices. Table A.10 shows the mean speedup and Table A.11 shows the task completion times, in month units, of the various strategies studied.

Table A.9. *Correlation of credit-value metric with speedup relative to FCFS*

λ	FCFS	BF	GANG (τ)				KBA (τ)			KBA with BF (τ)		
			6h	9h	30h		6h	9h	30h	6h	9h	30h
	Ref.								**Policy 1**			
0h	-	0.00	-0.07	-0.07	-0.04		0.28	0.30	0.37	0.27	0.29	0.33
0.64h	-	0.01	-0.07	-0.06	-0.04		0.24	0.25	0.27	0.21	0.23	0.23
1.6h	-	0.00	-0.05	-0.06	-0.04		0.16	0.17	0.21	0.14	0.14	0.17
1.92h	-	0.01	-0.04	-0.05	-0.04		0.07	0.09	0.20	0.03	0.02	0.02
6.4h	-	0.00	-0.04	-0.03	0.01		-0.06	-0.05	-0.02	-0.09	-0.09	-0.09

λ	KBA (τ)			KBA with BF (τ)			KBA (τ)			KBA with BF (τ)		
	6h	9h	30h	6h	9h	30h	6h	9h	30h	6h	9h	30h
	Policy 2						**Policy 3**					
0h	0.18	0.20	0.23	0.18	0.19	0.22	0.18	0.20	0.24	0.18	0.20	0.23
0.64h	0.19	0.20	0.23	0.16	0.19	0.13	0.19	0.20	0.22	0.15	0.16	0.14
1.6h	0.15	0.16	0.22	0.14	0.12	0.11	0.14	0.14	0.21	0.11	0.09	0.09
1.92h	0.06	0.09	0.24	0.01	0.01	0.02	0.02	0.04	0.19	0.01	0.00	0.00
6.4h	-0.07	-0.06	-0.02	-0.10	-0.10	-0.10	-0.07	-0.06	-0.02	-0.10	-0.10	-0.09
	Policy 4						**Policy 5**					
0h	0.01	0.01	0.01	0.01	0.01	0.01	0.00	0.01	0.01	0.00	0.01	0.01
0.64h	0.01	0.01	0.02	0.01	0.01	0.01	0.01	0.01	0.01	-0.00	0.00	0.01
1.6h	0.02	0.02	0.03	0.02	0.02	0.02	0.01	0.01	0.02	0.01	0.01	0.01
1.92h	-0.01	0.01	0.04	-0.01	-0.00	0.00	-0.01	-0.00	0.02	-0.01	-0.01	-0.00
6.4h	-0.08	-0.07	-0.03	-0.10	-0.10	-0.10	-0.08	-0.07	-0.04	-0.10	-0.10	-0.10

Table A.10. *Mean speedup relative to FCFS*

λ	FCFS	BF	GANG (τ)				KBA (τ)			KBA with BF (τ)		
			6h	9h	30h		6h	9h	30h	6h	9h	30h
	Ref.								**Policy 1**			
0h	1	1.41	2.01	1.80	1.03		3.58	3.34	2.48	3.72	3.49	2.75
0.64h	1	2.07	2.73	2.28	1.01		14.60	13.47	8.93	15.43	14.19	9.61
1.6h	1	3.00	3.93	2.97	0.85		15.99	14.11	8.29	16.50	14.71	9.16
1.92h	1	9.59	5.71	3.80	0.64		9.69	8.56	4.68	13.26	13.24	10.47
6.4h	1	1.02	0.84	0.79	0.62		1.01	0.95	0.74	1.19	1.19	1.16

λ	KBA (τ)			KBA with BF (τ)			KBA (τ)			KBA with BF (τ)		
	6h	9h	30h	6h	9h	30h	6h	9h	30h	6h	9h	30h
	Policy 2						**Policy 3**					
0h	10.74	9.36	5.09	11.02	9.72	5.61	10.76	9.59	5.75	10.93	9.81	6.09
0.64h	12.46	11.22	6.85	13.28	11.59	8.55	12.68	11.68	8.13	13.29	29.74	10.17
1.6h	10.37	9.09	4.60	11.31	10.62	7.50	10.77	9.78	5.80	12.06	16.52	8.12
1.92h	7.56	6.37	2.88	12.55	12.27	10.32	7.71	6.70	3.55	12.30	11.48	10.33
6.4h	1.00	0.94	0.73	1.18	1.17	1.15	1.00	0.94	0.73	1.19	1.18	1.16
	Policy 4						**Policy 5**					
0h	16.20	13.69	6.74	15.90	13.37	6.72	15.99	13.60	6.76	15.88	13.40	6.87
0.64h	10.74	10.03	7.60	11.38	10.92	9.46	10.59	10.08	7.64	11.31	11.13	9.03
1.6h	6.89	5.97	3.59	8.25	7.98	6.40	6.85	6.10	3.74	8.16	8.19	6.45
1.92h	6.51	5.19	2.25	11.38	11.11	10.02	6.77	5.58	2.42	11.89	11.03	10.35
6.4h	1.01	0.95	0.74	1.20	1.19	1.16	1.00	0.95	0.74	1.19	1.18	1.16

Table A.11. *Strategies' ending times in month (28 days) units*

λ	FCFS	BF	GANG (τ)				KBA (τ)			KBA with BF (τ)		
			6h	9h	30h		6h	9h	30h	6h	9h	30h
	Ref.								**Policy 1**			
0h	6.61	5.65	7.98	8.18	9.73		5.97	6.21	8.19	5.54	5.55	5.56
0.64h	6.64	5.68	6.89	7.09	9.09		5.97	6.21	8.16	5.56	5.57	5.58
1.6h	6.65	5.76	6.28	6.40	8.39		5.97	6.18	7.99	5.63	5.64	5.65
1.92h	6.70	6.10	6.27	6.41	8.20		6.13	6.21	7.84	6.06	6.07	6.07
6.4h	19.42	19.42	19.46	19.46	19.46		19.41	19.42	19.43	19.41	19.41	19.41

λ	KBA (τ)			KBA with BF (τ)			KBA (τ)			KBA with BF (τ)		
	6h	9h	30h	6h	9h	30h	6h	9h	30h	6h	9h	30h
	Policy 2						**Policy 3**					
0h	6.01	6.28	8.43	5.55	5.56	5.57	5.74	5.87	6.73	5.54	5.54	5.57
0.64h	6.03	6.29	8.32	5.58	5.58	5.59	5.78	5.90	6.74	5.56	5.56	5.59
1.6h	6.02	6.26	8.13	5.63	5.64	5.64	5.83	5.94	6.78	5.62	5.61	5.65
1.92h	6.13	6.23	7.89	6.06	6.06	6.07	6.09	6.13	6.81	6.06	6.06	6.07
6.4h	19.41	19.42	19.43	19.41	19.41	19.41	19.41	19.42	19.43	19.41	19.41	19.41
	Policy 4						**Policy 5**					
0h	5.99	6.11	6.73	5.65	5.65	5.66	5.83	5.93	6.53	5.61	5.62	5.65
0.64h	6.00	6.16	6.90	5.64	5.65	5.66	5.84	5.94	6.60	5.61	5.61	5.65
1.6h	5.97	6.12	6.99	5.63	5.63	5.67	5.87	5.97	6.67	5.63	5.65	5.65
1.92h	6.12	6.20	7.03	6.06	6.06	6.07	6.10	6.14	6.77	6.06	6.07	6.07
6.4h	19.41	19.42	19.43	19.41	19.41	19.41	19.41	19.42	19.43	19.41	19.41	19.41

A.3.1 Performance of the Allocation Strategies Tested

In this Appendix, we have reported the performance of the strategies and policies described in Chapter 5, Section 5.6. This performance is based on the mean values from a number of experiments and under different conditions. Further insight can be provided by the distribution of these mean values. For instance, if a particular strategy seems to present a consistent speedup behaviour, then one could expect that the speedup values of the experiments of that strategy would be normally distributed.

We decided to evaluate whether the mean speedup performance of the techniques previously described is consistent across experiments. This evaluation was carried out for a number of randomly selected strategies and inter-arrival times. That is, we did not perform a detailed statistical evaluation of all the experiments, since their characterization is not the purpose of this dissertation. The statistical evaluation of the selected experiments shows that the speedup performance is consistent. This indicates that knapsack strategies do indeed provide a better speedup than that of FCFS, BF or Gang. Below we describe the evaluations carried out.

We performed a Lilliefors test on the mean speedup values from the ten experiments of the BF strategy and the Knapsack with BF strategy (time slice of 6 hours), using an inter-arrival time $\lambda = 0$ hours. We also performed a Lilliefors test on the mean speedup values from the ten experiments of the BF strategy and Knapsack with BF strategy (time slice of 6 hours), using an inter-arrival time $\lambda = 1.6$ hours. The mean speedup values from each of the ten experiments of these strategies is given in Table A.12.

The Lilliefors tests results, comprising the value of the test statistic (LSTAT), the critical value of the test (CV), and the result of the hypothesis test (H) for various levels of significance α are given in Table A.13. The result H equals 1 if one can reject the hypothesis that the observed data has a normal distribution, or 0 if one cannot reject that hypothesis. The Lilliefors tests results indicate that the hypothesis that the observed data has a normal

distribution cannot be rejected.

Table A.12. *Mean speedup values per experiment*

Allocation Strategy			
$\lambda = 0$ hours		$\lambda = 1.6$ hours	
BF	KBA + BF Policy 1	BF	KBA + BF Policy 1
1.27	3.49	3.17	14.00
1.34	4.37	3.71	18.70
1.35	3.64	2.42	14.13
1.27	3.28	3.41	16.26
1.55	4.04	2.29	15.10
1.34	3.55	3.18	17.35
1.43	3.41	3.77	20.21
1.47	3.41	2.64	18.53
1.36	3.96	2.73	14.66
1.66	4.08	2.63	16.02

Table A.13. *Lilliefors tests results*

$\lambda = 0$ hours (Policy 1 for KBA+BF)								
		CV for various α						
Strategy	LSTAT	$\alpha = 0.01$	(H)	$\alpha = 0.05$	(H)	$\alpha = 0.10$	(H)
BF	0.229	0.294	(0)	0.258	(0)	0.239	(0)
KBA+BF	0.192	0.294	(0)	0.258	(0)	0.239	(0)
$\lambda = 1.6$ hours (Policy 1 for KBA+BF)								
BF	0.192	0.294	(0)	0.258	(0)	0.239	(0)
KBA+BF	0.144	0.294	(0)	0.258	(0)	0.239	(0)

Regarding the distribution of the speedup behaviour of individual experiments, we mention in section 5.7 that a Kolmogorov-Smirnov (K-S) goodness-of-fit test[3] for a number of well-known distributions (including weibull, lognormal, t location-scale, and extreme value) indicates that the per experiment data does not belong to any of these distributions.

[3] A K-S test is used to determine if a random sample X could have a hypothesized particular cumulative distribution function using a specified significance level α.

We observed, however, that a lognormal distribution could possibly be used to partially describe the observed data. This observation is based on the fact that a K-S test for lognormal distribution on some of the experiments indicates that the hypothesis that the tested data has a lognormal distribution cannot be rejected. For instance, Table A.14 shows the K-S tests results on each of the ten experiments of the Knapsack with BF strategy (time slice of 6 hours), using an inter-arrival time $\lambda = 1.6$ hours. The K-S tests results comprised the value of the K-S statistic (KSSTAT), the critical value of the test (CV), and the result of the hypothesis test (H) (significance level α=0.05). The result H equals 1 if one can reject the hypothesis that the observed data has the hypothesized distribution, or 0 if one cannot reject that hypothesis.

Table A.14. *K-S tests results ($\alpha = 0.05$)*

KBA+BF (Policy 1) (time slice 6 hrs) ($\lambda = 1.6$ hours)		
KSSTAT	CV	H
0.022	0.030	0
0.035	0.030	1
0.033	0.030	1
0.041	0.030	1
0.020	0.030	0
0.030	0.030	0
0.031	0.030	1
0.032	0.030	1
0.028	0.030	0
0.032	0.030	1

In Chapter 5, Section 5.6, using Figures 5.5 and 5.6, we depicted the average speedup value as a function of the assigned credit value for the BF strategy and for the Knapsack with BF strategy. These figures make evident the fact that high-valued tasks experience a greater speedup, compared to those of a lower value. The behaviour depicted in these figures is representative of the speedup-credit-value correlation observed. For instance,

Figures A.1 and A.2 show another example of this typical speedup-credit-value correlation.

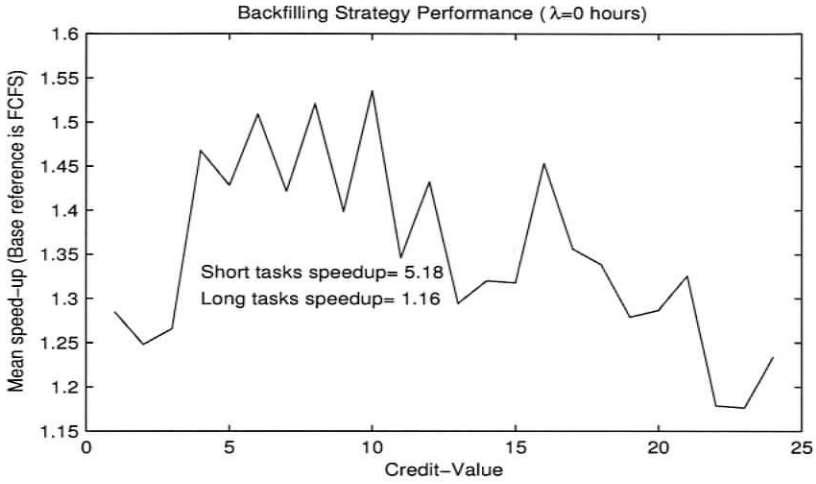

Figure A.1. *Speedup performance of the BF strategy.*

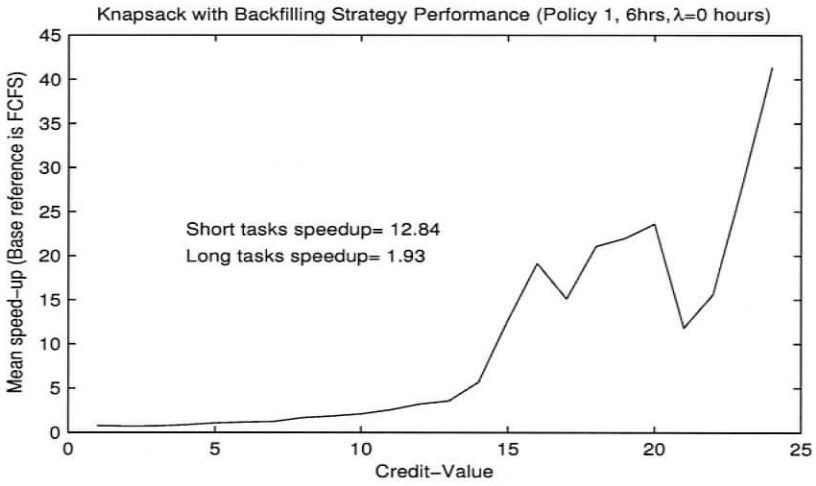

Figure A.2. *Speedup performance of the KBA+BF strategy.*