An Algorithm for Multi-objective Assignment Problem

TSE Hok Man

A Thesis Submitted in Partial Fulfilment
of the Requirements for the Degree of
Master of Philosophy
in
Information Engineering

Supervised by

Prof. LOK Tat Ming

June 2005

Abstract of thesis entitled:

An Algorithm for Multi-objective Assignment Problem

Submitted by TSE Hok Man

for the degree of Master of Philosophy

at The Chinese University of Hong Kong in June 2005

We study the multi-objective assignment problem. This problem is originally from the channel assignment problem in a multicarrier CDMA system and is closely related to the generalised assignment problem.

We observed that the multi-objective assignment problem can be transformed into a dynamic directed graph. We suggest a search method which combines a shortest path algorithm with a negative cycle detection strategy to find a solution to the multi-objective assignment problem. We show that this algorithm performs better on the application of channel assignment problem when comparing to the throughput optimization approach or the greedy approach to the same problem.

Because of the similarity between multi-objective assignment problem and the generalised assignment problem, we also apply the same algorithm to solve the generalised assignment problem and it shows good results in tackling the generalised assignment problem.

這篇論文主要是對多目標分配問題(multi-objective assignment problem)提出解決辦法。此問題是來自下行線路通道分配問題(downlink channel assignment)。多目標分配問題也和一般化分配問題(generalized assignment problem)有密切關係。

我們發現多目標分配問題可被轉變成為可變的有向圖(dynamic directed graph)，所以我們提出用最短距離算法(shortest path algorithm)和負循環探測方法(negative cycle detection)去找尋多目標分配問題的解答。

在下行線路通道分配問題上，傳統上我們會使用吞吐量最佳化(throughput optimization)的方法作為分配的策略，但這方法忽略個別用者的服務質量(quality of service)，而造成分配不合理。而我們提出的方法用於下行線路通道分配問題上，比使用傳統方法更有效率。

由於多目標分配問題和一般化分配問題的密切關係，我們也使用提出的方法去解答一般化分配問題。把以上的方法用於一般化分配問題，可得到理想的解答。

Acknowledgement

I would like to thank my supervisor professor Tat-Ming Lok. He gives me valuable ideas and advices in writing this thesis. I would also like to thank my colleagues Michael Ng, Anderson Cheng and Fan Hu, who discusses the problems with me and help me finish this thesis.

I would also thank people in I.S. Lab including H.Y. Kwan, S.W. Ho, Kelvin Yeung, Michael Ngai, K.K. Leung, C.S. Chen, P.W. Kwok, Robert Leung, Joesph Liu and Kayin Fu.

This work is dedicated to my parent and Amy

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The multi-objective assignment problem comes from the channel assignment problem in a multicarrier CDMA system [13] [11]. It is similar to most assignment problem which assign different agents to different jobs under some constraints. However, the objective function in multi-objective assignment problem may not be a linear function, and even they may not be the same for different jobs. This arises the multi-objective assignment problem which will be discussed in this thesis.

The channel assignment problem is a question in assigning different channels to different users such that the 'happiness' of users is optimal. The 'happiness' may refer to the average system throughput, the average latency or any other utility function and it may be difference for each user.

In many current channel assignment schemes, total system throughput is the main concern. So these schemes aim at maximizing the total system throughput of the wireless network. However, this approach may result an unfair and inefficient channel

assignment to each user. In third generation and future wireless network, quality of service (QoS) requirement of each user may vary a lot. Traditional approach of maximizing the total system throughput may not satisfy the QoS requirement of each user. For example, some users may not get enough throughput while the others have too much beyond their requirement.

By consider the QoS requirement and formulate it as mathematical functions, the channel assignment problem can be generalised and modelled by a goal programming and becomes a multi-objective assignment problem [13].

In multi-objective assignment problem, a set of agents is going to be assigned to a set of jobs. The constraints of the multi-objective assignment problem are that each agent can only be assigned to exactly one job and each job cannot be done by more than certain agents. All jobs have to be finished. The objective function can be any function.

To solve the multi-objective assignment problem, we adopt the Bellman-Ford-Moore algorithm with the Amortized search negative cycle detection strategy. This search tries to move from one feasible solution to another better feasible solution in each iteration. The multi-objective assignment problem is first transformed into a dynamic directed graph. The arc length in the graph is changing according to the current assignment. If there exists an assignment which is better than the current assignment, there exists a negative cycle in the graph. By finding out all the negative cycle, the search is finished.

Since the multi-objective assignment problem is similar to the generalised assignment problem and the quadratic assignment problem, we also apply the same algorithm to solve this two problems. The two problems have been proved to be NP-hard [8].

This thesis is organized as follows. Chapter 2 is the background studies which will discuss the channel assignment problem, the generalised assignment problem and the negative cycle detection. Chapter 3 will discuss the multi-objective assignment problem and the algorithm to tackle this problem. Chapter 4 will apply the algorithm in the channel assignment problem and analysis its performance. Chapter 5 will discuss the application on the generalised assignment problem and the quadratic assignment problem. Chapter 6 is the conclusion.

□ **End of chapter.**

Chapter 2

Background Study

In this chapter, we will discuss the basic concepts and background knowledge to our work. Since the multi-objective assignment problem is originally come from the channel assignment problem in the multicarrier CDMA system, we will discuss the basic knowledge in it. Moreover, the multi-objective assignment problem is similar to the generalised assignment problem, so a brief discussion on that will be included in section 2.2. Since we have adopted the negative cycle detection to tackle the multi-objective assignment problem, we will discuss the labeling method with the negative cycle detection strategy in section 2.3.

2.1 Channel Assignment in Multicarrier CDMA Systems

In a multicarrier CDMA system with orthogonal signature sequences, different users will experience different channel fading for the same signature sequence. If the same

signature sequence is given to two different users, they will have different signal-to-noise ratio, and thus have different throughput. Because of this, it has been show that in [11] that the average throughput or the average latency in the system can be improved by assigning different signature sequences to different users carefully.

2.1.1 Channel Throughput

When a multicarrier CDMA system with orthogonal signature sequences is considered, the multiple access interference (MAI) is zero. We let $\vec{s_i}$ be the i-th signature sequence and each sequence is normalized to unit norm. The background noise is assumed to be the additive white Gaussian noise (AWGN) with mean zero and variance σ^2. Let g_i be the large-scale path loss of client j. Then the signal-to-noise ratio (SNR) of the i-th signature sequence to user j can be calculated by

$$\gamma_{ij} = \frac{g_j^2}{\sigma^2 \vec{s_i}^H A_j^{-1} A_j^{-1H} \vec{s_i}} \tag{2.1}$$

where A_j is a diagonal matrix whose ith element is α_{ij} and α_{ij} accounts for the overall effects of phase shift and fading for the ith carrier of the jth user's received signal. For different signature sequence $\vec{s_i}$, the SNR γ_{ij} may be different.

After defining the SNR, the throughput of that channel can be found by Shannon's capacity formula, which is

$$R_{ij} = B \log_2(1 + \gamma_{ij}) \tag{2.2}$$

where B is the channel bandwidth.

2.1.2 Greedy Approach to Channel Assignment

A greedy approach has been used in [11] to select the channel assignment. It also defines the orders of selection diversity, which means the number of distinct values in the set $\{R_{ij} : 1 \leq i \leq N\}$. When the orders of selection diversity is equal to 1, all channels are the same to individual user. So any arbitrary channel assignment work as good as assigning channels with greedy approach. With the orders of selection diversity increases, the improvement in the average throughput or the average latency increases [11].

In order to achieve the highest orders of selection diversity, orthogonal signature sequences are considered. When the number of channels is equal to the number of clients which is equal to N, the channel assignment problem can be formulated as a linear programming problem which is as follows:

$$\max R = \sum_{i=1}^{N} \sum_{j=1}^{N} \beta_j R_{ij} x_{ij} \tag{2.3}$$

subject to

$$\sum_{i=1}^{N} x_{ij} = 1 \text{ for } j = 1, 2, \ldots, N \tag{2.4}$$

$$\sum_{j=1}^{N} x_{ij} = 1 \text{ for } i = 1, 2, \ldots, N \tag{2.5}$$

and

$$x_{ij} = \begin{cases} 1 & \text{if channel i is assigned to user j,} \\ 0 & \text{otherwise.} \end{cases} \tag{2.6}$$

where β_j is a weighted factor for user j.

The above problem can be solved readily by many mathematical tools to obtain an optimum solution.

2.2 Generalised Assignment Problem

Generalised Assignment Problem (GAP) is the problem to find a minimum cost assignment of a set of jobs to a set of agents subject to the resource constraints. The mathematical model can be formulated as follows:

$$\min c(x) = \sum_{i=1}^{M} \sum_{j=1}^{N} c_{ij} x_{ij} \tag{2.7}$$

subject to

$$\sum_{j=1}^{N} r_{ij} x_{ij} \leq b_i, \forall i \in I, \tag{2.8}$$

$$\sum_{i=1}^{M} x_{ij} = 1, \forall j \in J, \tag{2.9}$$

$$x_{ij} \in \{0, 1\}, \forall i \in I, \forall j \in J. \tag{2.10}$$

where c_{ij} is the cost for assigning agent i to job j and r_{ij} is the resource consumed by agent i when it is assigned to job j. Equation 2.8 is the resources constraints which ensure that the total resource requirement of the jobs assigned to each agent do not exceed its capacity.

The problem in section 2.1 is one of the special case of the GAP where $c_{ij} = \beta_i$, $r_{ij} = 1$ and $b_i = 1$. This type of assignment problem can be solved readily by different LP methods.

The GAP is known to be strongly NP-hard [8]. However, we are still interested in this problem because it has many applications. The GAP can be applied in a wide rage of areas like computer and communication network, location problem, vehicle routing and machine scheduling. And it is related to many classical problems like the multiple knapsack, the bin packing and the set partitioning problem. Different algorithms have been developed to tackle the GAP, including the branch-and-bound approach and the heuristics approach.

2.2.1 Branch and Bound Approach for GAP

Branch and Bound is a general search method. To apply branch and bound, one must have a way to compute a lower bound and an upper bound on an instance of an optimization problem. Moreover, there must be a way to divide the optimization problem into different feasible region to create smaller subproblems.

In branch and bound approach, the whole feasible region is called the root problem. We first computer the lower bound and the upper bound of the root problem. If both bound match, an optimal solution has been found. Otherwise, the whole feasible region is divided into two or more regions and these subregions should cover the whole feasible region. The algorithm apply the upper and lower bound to these subregions recursively and generate a tree of subproblems. The idea of the branch and bound approach is to eliminate those subproblems where the optimal solution cannot occur. If the lower bound of a subproblem exceeds the best know feasible solution, no globally optimal solution can be found in its descendant and this sub-

problem is eliminated. If an optimal solution is found to a subproblem, then it is a feasible solution to the whole problem, but is not necessarily be the global optimal. The search terminate until all the subproblems have been solved or pruned.

When we use the branch and bound approach to solve the GAP, an initial solution is required. In order to obtain a good initial solution, different approaches can be used. One of them is denoted as MTH, which is described in [12] and [7]. Let μ_{ij} be a heuristic measure of the desirability of assigning job j to agent i and can be defined as: (a) $-c_{ij}$, (b) $-c_{ij}/r_{ij}$, (c) $-r_{ij}$ or (d)$-r_{ij}/b'_i$. MTH iteratively considers all the unassigned jobs and determines a job j^* with maximum difference between the largest and the second largest μ_{ij}. The capacity constraints in equation 2.8 may not be violated. Job j^* is then assigned to agent i with the largest μ_{ij^*}. In the second part of the heuristic, the current solution is eventually improved by a local search called shift operations. The shift operations considers each job and tries to reassign it to another feasible agent with the smallest costs. The best solution obtained is the final MTH solution. However, MTH does not guarantee to terminate with a feasible solution.

After an initial solution is found, it serves as the global upper bound for the depth-first branch-and-bound scheme. A lower bound is obtained at each node of the decision tree by solving the relaxed problem which ignore equation 2.9. This subproblem corresponds to m individual 0-1 single knapsack problems which can be solved by dynamic programming efficiently. If the solution obtained in the relaxed

problem satisfies the constraint in equation 2.9, the node generates no descendants [7].

For sufficient time allowed, the branch-and-bound approach can find the exact solution. However, branch-and-bound approach is only effective on certain GAP instances of small and medium size [7]. Larger and more complex instances can be tackled by heuristics approach, and one of them is the genetic algorithm approach.

2.2.2 Genetic Algorithm for GAP

Genetic Algorithm (GA) is an 'intelligent' probabilistic search algorithm which simulates the process of evolution by taking a population of solutions and applying genetic operators in each reproduction [6]. Each solution in the population is evaluated according to some fitness measure. If a solution is highly fit, reproduction opportunity is given to the solution. If a solution is unfit, it is replaced.

In GA approach for the GAP, a candidate solution is represented by a vector $S = (S_1, \ldots, S_n) \in I^J$. For each $j = 1, \ldots, n$, the value S_j represents the agent to which job j is assigned to. This representation ensures that the assignment constraint in equation 2.9 are always satisfied.

Initial candidate solutions can be created uniformly at random. The GA's variation operators are standard one-point crossover and the exchange of two randomly chosen positions representing mutation. Each offspring solution can be heuristically improved by different search methods. One of the search method can be found in [7] and [6]. This search method involves two phases. In the first phase, the procedure

tries to repair infeasible solutions which violate the capacity constraints in equation 2.8 to a feasible solutions. This is done by reassigning jobs from agents with exceeded capacities to other agents. In the second phase, the procedure aims at improving total costs by assigning jobs without further violating the capacity constraints.

Genetic algorithm work efficiently for large and complex GAP problem instances, however, it does not guarantee an exact solution.

2.3 Negative Cycle Detection

Negative cycle detection is the problem to find a negative cycle in a graph or prove there is none. It is closely related to the shortest path problem in a graph and it finds its application on the minimum-cost flow problem, the currency arbitrage and the circuit design problem [5] [4] [15].

The negative cycle detection problem can be solved by different shortest path algorithms combined with different cycle detection strategies. One of the fastest method is the combination of Bellman-Ford-Moore algorithm and Amortized Search, which can achieve an $O(nm)$ bound [5]. Here, n and m denote the number of vertices and the number of arcs in the graph respectively.

2.3.1 Labeling Method

This section will briefly describe the general labeling method for solving the shortest path problem [5] [15]. Most shortest path algorithms, including the Bellman-Ford-

Moore algorithm in section 2.3.2, are based on this labeling method.

Before discussing the labeling method, we would like to define some notations used in it. The input to the shortest path problem is (G, s, l), where $G = (V, E)$ is a directed graph, $l : E \rightarrow R$ is a length function and $s \in V$ is the source vertex. We also denote $|V|$ by n and $|E|$ by m.

The general labeling method is as follows:

1. For every vertex v, it maintains its distance label $d(v)$ and parent $p(v)$.

2. Initially, $d(v) = \infty$ and $p(v)$ =null for all v.

3. Set $d(s) = 0$.

4. Selects an arc (u, v) such that $d(u) < \infty$ and $d(u) + l(u, v) < d(v)$ and sets $d(v) = d(u) + l(u, v), p(v) = u$.

5. Repeat steps 4 until $d(u) + l(u, v) \geq d(v)$ for all u and v.

The labeling method tries to search those admissible arcs which decrease the distance of certain nodes from the starting node. If the graph contains no negative cycle, the labeling method will be terminated and $d(v)$ and $p(v)$ gives the correct shortest distances and parent pointers respectively [5]. Step 4 is called the labeling process.

If the graph contains negative cycles, the labeling method will not be terminated. So a negative cycle detection strategy should be used. Most negative cycle are based on the two facts. First, if the input graph has a negative cycle and the labeling

method is applied with no cycle detection strategy, the distance label of some vertices will get arbitrarily negative. Second, if the distance label of a vertex v is smaller than the length of a shortest simple path from s to t, then the input graph has a negative cycle [5].

2.3.2 Bellman-Ford-Moore algorithm

The Bellman-Ford-Moore algorithm is based on the labeling method in section 2.3.1. It maintains the set of labelled vertices in a FIFO queue. The next vertex to be scanned is removed from the head of the queue and the vertex that just become labelled is added to the end of the queue. The Bellman-Ford-Moore algorithm runs in $O(nm)$ time in the worst case [5]. The Bellman-Ford-Moore algorithm is as follows.

1. initially, $d(v) = \infty$, $p(v) = null$ and $d(s) = 0$.

2. for each vertex i = 1 to V(G) - 1, do the following step

3. for each edge (u, v) in E(G), RELAX(u, v, l).

4. for each edge (u, v) in E(G), if $d(u) + l(u, v) < d(v)$ then return false.

5. return true.

The RELAX(u,v,l) procedure is as follow.

If $d(v) > d(u) + l(u, v)$, then $d(v) = d(u) + l(u, v)$ and $p(v) = u$

The Bellman-Ford-Moore algorithm is based on the labeling method. Step 2 is the modified labeling method and step 4 is for cycle checking. If the graph has cycle,

the algorithm return false and no shortest path can be found.

2.3.3 Amortized Search

Amortized search is one of the simple cycle detection method. After each labeling process, the vertex searches its parent to detect a cycle. The cost of such search is $O(n)$. When it is combined with Bellman-Ford-Moore algorithm, the performance will be affected by a constant factor only. One disadvantage of amortized search is that it does not guarantee to find a cycle at the first search when the cycle appears [5].

By combining the Bellman-Ford-Moore algorithm with the Amortized Search, we can achieve a negative cycle detection strategy which runs in $O(nm)$.

□ **End of chapter.**

Chapter 3

Multi-objective Assignment Problem

The multi-objective assignment problem was first come from the channel assignment problem in a multicarrier CDMA system. In this chapter, we will discuss the details of the multi-objective assignment problem. The multi-objective assignment problem is an NP-Hard problem in general. We observed that this kind of problem can be transformed into a dynamic directed graph problem. Following the definition, an algorithm will be discussed to tackle it in section 3.4. A simple example is provided in section 3.5 to illustrate the algorithm. A special case of the multi-objective assignment problem is considered in section 3.6 and the application on the assignment problem will be discussed in section 3.7.

3.1 Multi-objective Assignment Problem

The multi-objective assignment problem is an assignment problem which assigns a group of N agents to a group of M jobs subject to some constraints. This kind of assignment problem can have multiple objective functions and we are going to minimize them one by one according to their priority.

Before continuing the discussion, we need to define the lexicographic minimum first. The definition of lexicographic minimum is quoted from [10] directly and stated as follow:

> For two vectors $\vec{a}^{(1)}$ and $\vec{a}^{(2)}$, $\vec{a}^{(1)}$ is preferred to $\vec{a}^{(2)}$ if there exists an integer k such that $a_k^{(1)} < a_k^{(2)}$ and all high order terms (i.e. a_1, a_2, ..., a_{k-1}) are equal. If no other vectors is preferred to $\vec{a}$, then $\vec{a}$ is the lexicographic minimum.

Since the assignment problem can have multiple objective functions, we would like to write these objective into a vector form according to their priority and this makes up the multi-objective assignment problem which is described as follow.

There are M jobs which are needed to be done by N agents, where $M \leq N$. We want to find an assignment such that the 'cost' of the assignment is minimized. All the agents cannot be assigned to more than one job and all the jobs cannot be done by more than n_i agents. The mathematical formulation of the problem are as follow.

$$\text{lexmin } \vec{v} = (v_1, v_2, \cdots) = \left(\sum_{i=1}^{M} f_i^{(1)}(\vec{x_i}), \sum_{i=1}^{M} f_i^{(2)}(\vec{x_i}), \cdots \right) \tag{3.1}$$

subject to

$$\sum_{i=1}^{M} x_{ij} \leq 1 \tag{3.2}$$

$$\sum_{j=1}^{N} x_{ij} \leq n_i \tag{3.3}$$

and

$$x_{ij} = \begin{cases} 1 & \text{if agent j is assigned to job i,} \\ 0 & \text{otherwise} \end{cases} \tag{3.4}$$

where $\vec{x_i} = [x_{i1}\ x_{i2}\ \cdots\ x_{iN}]'$. $f_i^{(j)}(\vec{x_i})$ represents the j-th priority objective function of job i. $v_1 = \sum_{i=1}^{M} f_i^{(1)}(\vec{x_i})$ is the first function we want to minimize and it has the highest priority. $v_2 = \sum_{i=1}^{M} f_i^{(2)}(\vec{x_i})$ is the second function we want to minimize and it has the second priority and so on. Since the assignment problem tries to minimize multiple objectives, so it is called the multi-objective assignment problem.

The multi-objective assignment problem is quite similar to the GAP in section 2.2 except that the resources constraints parameters r_{ij} is always equal to 1 and the objective function in equation 3.1 can be non-linear.

The multi-objective assignment problem can model a wide range of problems including the channel assignment problem. The details can be found in chapter 4.

3.2 NP-Hardness

The multi-objective assignment problem is an NP-Hard problem in general. We would like to prove this by transforming the GAP into a special case of the multi-objective assignment problem.

For all GAP, it can be transformed into a multi-objective assignment problem with two different objective functions which is in the following form.

$$\text{lexmin } \vec{v} = (v_1, v_2) = \left(\sum_{i=1}^{M} u(\sum_{j=1}^{N} r_{ij}x_{ij} - b_i), \sum_{i=1}^{M}\sum_{j=1}^{N} c_{ij}x_{ij} \right) \tag{3.5}$$

subject to

$$\sum_{j=1}^{N} x_{ij} \leq n_i, \ \forall i \in I, \tag{3.6}$$

$$\sum_{i=1}^{M} x_{ij} = 1, \ \forall j \in J, \tag{3.7}$$

$$x_{ij} \in \{0, 1\}, \ \forall i \in I, \forall j \in J \tag{3.8}$$

The function $u(x)$ is defined as

$$u(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{3.9}$$

Equation 3.7 is the same as that in GAP and it limits each agent must be assigned to one job. The value n_i can be found by considering the maximum number of agents that can be supported by job i. v_1 corresponds to the resources limitation of each job while v_2 represents the total cost of the assignment. We do not consider the resources limitation constraints in GAP as a constraints in the corresponding multi-objective assignment problem, but consider it as an objective to be minimized. If the

resources limitation constraints in GAP is satisfied, v_1 is equal to zero and the cost of the assignment, v_2, will be considered. Because we use the lexicographic minimum, a feasible solution in the GAP is more preferable in the multi-objective assignment problem since a feasible solution has a value of $(0, v_2)$ while an infeasible solution has a value of (v_1', v_2) and $v_1' > 0$.

If there exists a better solution in the GAP, there exists a better solution in the corresponding multi-objective assignment problem. Consider two feasible solutions with cost c_1 and c_2 , where $c_1 < c_2$. These two solutions will produce the value $(0, c_1)$ and $(0, c_2)$ respectively in the multi-objective assignment problem. Since $c_1 < c_2$, so $(0, c_1)$ is a better solution to the assignment problem.

Since every GAP can be transformed into the above multi-objective assignment problem and GAP is an NP-hard problem. The multi-objective assignment problem is also NP-hard in general.

3.3 Transformation of the Multi-objective Assignment Problem

We would like to transform the multi-objective assignment problem into a dynamic directed graph problem. We need to add some dummy costs and constraints before the transformation. By adding dummy variables and equations, it ensures that each agent must be assigned to exactly one job and each job is done by exactly the

maximum number of agents allowed, that is

$$\sum_{i=1}^{M} x_{ij} = 1 \tag{3.10}$$

$$\sum_{j=1}^{N} x_{ij} = n_i. \tag{3.11}$$

Since there are multiple objectives to be minimized, we would like to define a function $f_i(\vec{x_i})$ which map the vector, $(v_1, v_2, \cdots)$ into a single value. The function $f_i(\vec{x_i})$ is defined as follows.

$$f_i(\vec{x_i}) = \epsilon_1 v_1 + \epsilon_2 v_2 + \cdots \tag{3.12}$$

$$= \epsilon_1 \sum_{i=1}^{M} f_i^{(1)}(\vec{x_i}) + \epsilon_2 \sum_{i=1}^{M} f_i^{(2)}(\vec{x_i}) + \cdots \tag{3.13}$$

By selecting the value of ϵ_i carefully, minimize $f_i(\vec{x_i})$ is the same to minimize the vector $(v_1, v_2, \cdots)$. The value, ϵ_i, should have the property that adding the terms $\epsilon_i v_i$ does not affect the choice of assignments which minimize the value of $\sum_{k=1}^{i-1} \epsilon_k v_k$. The value ϵ_i can be obtained differently for different application. For the simplest case, when there are only one objective, ϵ_1 can be equal to 1 and $f_i(\vec{x_i}) = f_i^{(1)}(\vec{x_i})$.

Each node in the graph represents an agent. An initial feasible assignment need to be obtained before the transformation. We can use the MTH suggested in [12] and [7] to obtain an initial assignment or construct it uniformly and randomly in the feasible region.

We define the function $J(p)$ which is equal to the job that agent p has been assigned. Without loss of generality, we assume $J(p) = i$ in the following expression.

The cost of the edge between two nodes p and q, $d_{p,q}$, is calculated as follows.

$$d_{p,q} = f_i(\vec{x_i}') - f_i(\vec{x_i}) \tag{3.14}$$

where

$$\vec{x_i} = [x_{i1}\ x_{i2}\ \cdots\ x_{ip}\ \cdots\ x_{iq}\ \cdots\ x_{iN}]' \tag{3.15}$$

$$\vec{x_i}' = [x'_{i1}\ x'_{i2}\ \cdots\ x'_{ip}\ \cdots\ x'_{iq}\ \cdots\ x'_{iN}]' \tag{3.16}$$

and

$$x_{ij} = x'_{ij}, \forall j \neq p, q, x_{ip} = 1, x'_{ip} = 0, x_{iq} = 0 \text{ and } x'_{iq} = 1. \tag{3.17}$$

Two nodes are connected if and only if the corresponding two agents are assigned to different jobs. If the two agents are assigned to the same job, there is no edge between the corresponding nodes.

The cost of the edge, $d_{p,q}$, is the change in the objective function if job $J(p)$ is assigned to agent q instead of agent p and it is different from the cost $d_{q,p}$. Since each agent must be assigned to exactly one job and each job is done by the maximum number of agents that it can support, the cost can be calculated based on the current assignment of each job.

For all job exchanges between agents, we can find a cycle in the graph with the total cost equal to the change in the objective function. Consider the following example.

Suppose the initial assignment is as follow.

$$\mathbf{X} = \begin{bmatrix} 1 & 0 & 0 & \cdots \\ 0 & 1 & 0 & \cdots \\ 0 & 0 & 1 & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

Now agent 1 takes agent 2's job, agent 2 takes agent 3'job and agent 3 takes agent 1's job. We denote this job exchange as $1 \rightarrow 2 \rightarrow 3 \rightarrow 1$. And all the other agents take their original assigned job. The assignment after this job exchange is

$$\mathbf{X}' = \begin{bmatrix} 0 & 0 & 1 & \cdots \\ 1 & 0 & 0 & \cdots \\ 0 & 1 & 0 & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

The difference in the objective function is

$$\begin{aligned} D &= F(\mathbf{X}') - F(\mathbf{X}) \\ &= \sum_{i=1}^{M} f_i(\vec{x_i}') - \sum_{i=1}^{M} f_i(\vec{x_i}) \\ &= \sum_{i=1}^{3} f_i(\vec{x_i}') - \sum_{i=1}^{3} f_i(\vec{x_i}) \\ &= \sum_{i=1}^{3} (f_i(\vec{x_i}') - f_i(\vec{x_i})) \\ &= d_{1,2} + d_{2,3} + d_{3,1} \end{aligned}$$

Each cycle in the graph means the movement of a feasible solution into another feasible solution. This is because in each cycle, a job must be taken up by another

agent. Since the constraints are satisfied in the original assignment, the constraints are also satisfied after the job exchange. The total cost of each node from the origin means the change in the objective function. Each negative cycle in the graph means a movement to a better feasible solution. To obtain the optimal solution, we want to find out all the negative cycle in the dynamic directed graph.

If the cost of the edge between all the nodes do not vary, we can always find the negative cycle from the given graph by the negative cycle detection strategy described in section 2.3. However, the cost between nodes vary according to the current state of the graph, that is $d_{p,q}$ varies from time to time according to its parent nodes. So we suggest to use the Adaptive Bellman-Ford algorithm [4] to detect the negative cycles in the graph. The algorithm is modified to adopt the problem and is described in the following section.

3.4 Algorithm

To solve the multi-objective assignment problem, we adopt the Adaptive Bellman-Ford (ABF) algorithm in [4] combined with the Amortized Search in section 2.3.3.

Let $d(i)$ be the distance label of node i from the initial node. The distance label, $d(i)$, is the total cost of edge of node i from the origin. Let $p(i)$ be the list of nodes which are arranged before node i. Suppose node s is the initial node. The algorithm are as follow.

1. Set $d(n) = \infty$ for $1 \leq n \leq N$.

2. Set $d(s) = 0$.

3. Add the initial node to the updated queue.

4. For each node i in the updated queue, do the following

   - Calculate $d_{i,n}$ for $1 \leq n \leq N$ except $p(i) \backslash \{s\}$.
   - For all n, if $d(n) > d(i) + d_{i,n}$, then set $d(n) = d(i) + d_{i,n}$, $p(n) = p(i) + i$ and add the node to the updated list.
   - Repeat until there are no node in the updated queue.

5. If $d(s) < 0$, change the assignment accordingly.

A feasible solution has to be found before the above algorithm runs and can be constructed randomly. Since the constraints are simple, we can construct a feasible solution easily. Like many other algorithms, a good initial solution could help in finding a good final solution. However, getting a good initial solution is sometimes difficult.

After a feasible solution is obtained, the algorithm calculates the difference in the objective function if an agent takes up another agent's job and denotes this as the cost of the edge between two nodes. The above algorithm updates the edge cost between nodes after each labeling process. A negative cycle in the graph means if the agents in this cycle exchange their jobs accordingly, the value of the objective function will be decreased. If a negative cycle is found, it will change the current assignment according to the negative cycle. The algorithm tries to find a better

solution with different starting nodes. If no negative cycle can be found for all starting node after running the algorithm, the search stop.

For each node, ABF algorithm runs once with the Amortized search, which runs in $O(N^2)$. So the above algorithm runs in $O(N^3)$.

In order to illustrate the algorithm better, a simple example can be found in the next section.

3.5 Example

This section will illustrate the algorithm described in section 3.4 by a simple example. This example is a typical assignment problem which contains 3 jobs and 6 agents. Each job must be finished by 2 agents. The cost matrix is

$$C = \begin{bmatrix} 5 & 4 & 2 & 7 & 1 & 10 \\ 9 & 9 & 3 & 3 & 8 & 5 \\ 1 & 1 & 2 & 2 & 5 & 5 \end{bmatrix} \tag{3.18}$$

The values in the cost matrix are randomly generated from 1 to 10. This example is a simple case of the multi-objective assignment problem with one linear objective function. Let $f_i^{(1)}(\vec{x_i}) = \sum_{j=1}^{6} c_{ij}x_{ij}$. The objective is to minimize the total cost, that is

$$\min \sum_{i=1}^{3} \sum_{j=1}^{6} c_{ij}x_{ij}. \tag{3.19}$$

subject to

$$\sum_{i=1}^{3} x_{ij} = 1 \; \forall j, \tag{3.20}$$

$$\sum_{j=1}^{6} x_{ij} = 2 \; \forall i, \tag{3.21}$$

$$x_{ij} \in \{0, 1\} \tag{3.22}$$

The initial distance vector is

$$Dist_0 = \begin{bmatrix} 0 & \infty & \infty & \infty & \infty & \infty \end{bmatrix}. \tag{3.23}$$

We start by an arbitrary assignment

$$A_0 = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \end{bmatrix} \tag{3.24}$$

The value of the objective function of this arbitrary assignment is 30. After obtaining this arbitrary assignment, we can calculate the edge cost matrix by equation 3.14. For example, the cost between node 1 and node 3, $d_{1,3}$, can be calculated as follows:

$$d_{1,3} = (3 + 3) - (9 + 3) = -6. \tag{3.25}$$

The calculated edge cost matrix is

$$D_0 = \begin{bmatrix} \infty & 0 & -6 & \infty & -1 & -4 \\ 0 & \infty & 1 & 1 & \infty & 4 \\ 3 & 2 & \infty & 5 & -1 & \infty \\ \infty & 6 & 0 & \infty & 5 & 2 \\ -4 & \infty & -3 & -3 & \infty & 0 \\ -5 & -6 & \infty & -3 & -9 & \infty \end{bmatrix} \tag{3.26}$$

Figure 3.1 is the graphical representation of the above edge cost matrix. For clarity, we only include the edge from node 1 to the other nodes. Each edge has a value which denote the cost of that edge. Since the cost from node p to node q is different from that from node q to node p, so there are two edges connecting each pair of nodes. In figure 3.1, node 1 and node 4 are not connected because channel 1 and 4 are assigned to the same client. Each node in the graph has two value denoted by A/B, where A is the node number and B is the distance (total cost) from the starting node.

Since $d(1) + d_{1,2} < d(2)$, we set $d(2) = d(1) + d_{1,2} = 0 + 0 = 0$. Other nodes are updated similarly and a new distance vector is found and equals

$$Dist_1 = \begin{bmatrix} 0 & 0 & -6 & \infty & -1 & -4 \end{bmatrix}. \tag{3.27}$$

Figure 3.2 shows the updated graph. The total cost of each node is updated. Some edges in the graph are bold because these edges are admitted.

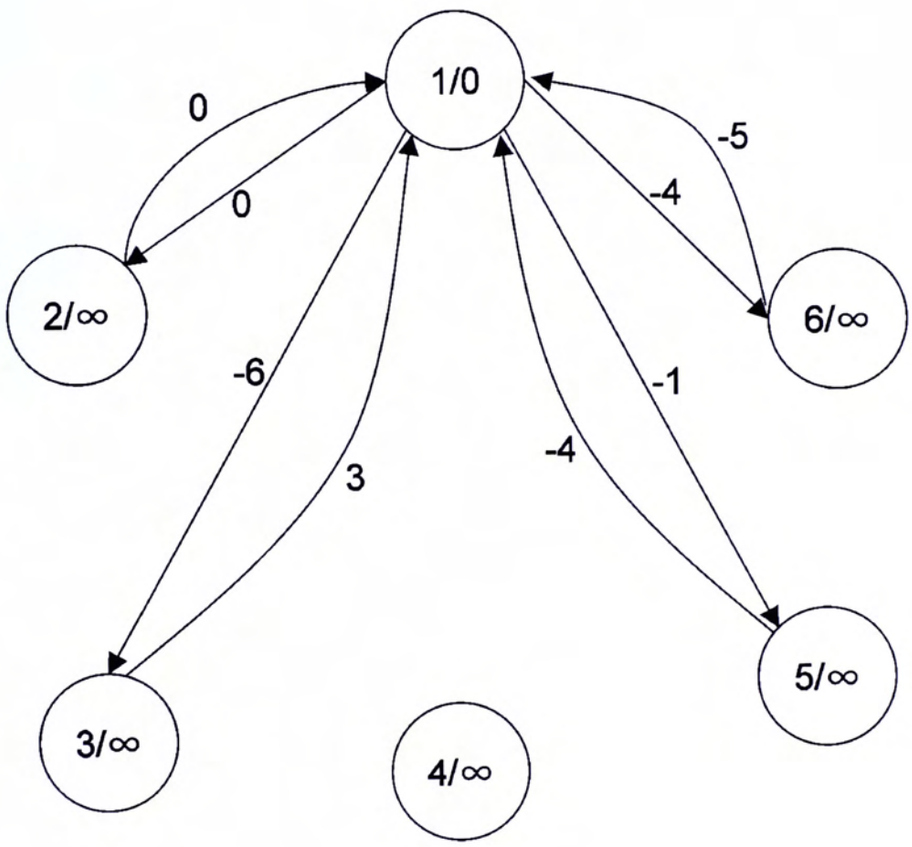

Figure 3.1: Graph before the search (only edge connecting node 1 are shown)

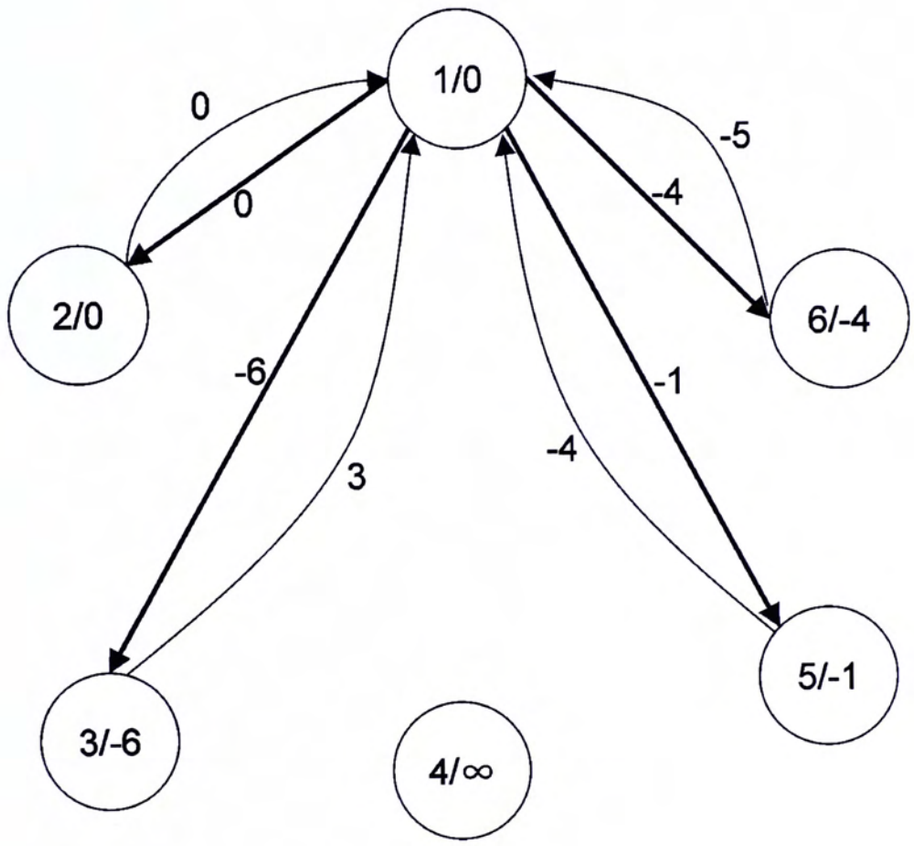

Figure 3.2: Graph after the 1st update (bold means the edge are admitted)

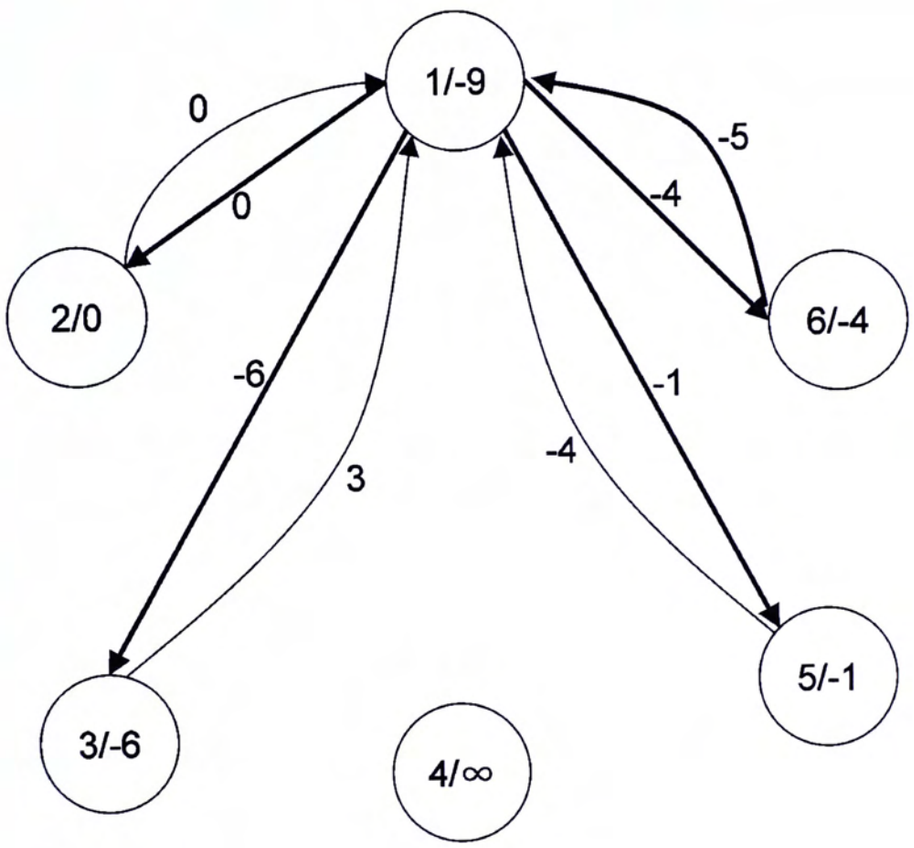

Figure 3.3: Graph after the search

The distance vector is updated repeatedly until the search stop. If the distance of the starting node is negative, a negative cycle is found. In the above example, a negative cycle is found after the algorithm stopped and is equal to $1 \rightarrow 6 \rightarrow 1$. Figure 3.3 shows the result of this search. A cycle is formed from node 1 to node 6 and back to node 1 with distance (total cost) -9. By following this negative cycle, the value of the objective function is reduced to 21 and the new assignment is

$$A_1 = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 \end{bmatrix} \tag{3.28}$$

After the second search, a negative cycle is found and is $1 \rightarrow 5 \rightarrow 1$. By following this negative cycle again, the value of the objective function is reduced to 13 and the assignment matrix is

$$A_2 = \begin{bmatrix} 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.29}$$

By running more search with different nodes as the starting node, no more negative cycle can be found. The algorithm stops and the assignment matrix A_2 is declared as the optimal solution with the optimal value 13.

3.6 A Special Case - Linear Objective Function

In this section, we consider a special case of the multi-objective assignment problem. This special case of the multi-objective assignment problem is that there are only one objective function and this objective function is linear and it reduces to a linear programming problem.

The edge cost between nodes are defined as the change in the objective function when an agent takes up another agents job. If the objective function is a linear function, this change does not depend on other agents who have already assigned to the same job. So no matter how the other agents exchange their job, the edge cost between each node only depends on the original assigned agent and does not change.

If the multi-objective assignment problem has one objective function only and it is a linear function of x_{ij}, then the objective function can be written as

$$\min \sum_{i=1}^{M} f_i(\vec{x_i}) = \sum_{i=1}^{M} \sum_{j=1}^{N} f_i(x_{ij}). \tag{3.30}$$

and the cost between node p and q can be written as follows:

$$d_{p,q} = f_{J(p)}(x_{J(p)q}) - f_{J(p)}(x_{J(p)p}) \tag{3.31}$$

and $J(p)$ is the mapping function which equals to the job assigned to agent p in the original assignment.

Under this situation, the cost of the edge between each node does not vary and the negative cycle detection method can always found the negative cycle if the graph has one. The proof can be found in [5].

3.7 Performance on the assignment problem

Although the algorithm described in section 3.4 can be used to solve the multi-objective assignment problem, we want to evaluate its performance on the typical assignment problem described in section 2.1.2.

The typical assignment problem is a special case of the multi-objective assignment problem, where the objective function is a linear function of x_{ij}. This type of problem can be solved optimally by the algorithm in section 3.4. Since the typical assignment problem is a one-to-one mapping, that is a set of N agents should be assigned to a set of N jobs. When the problem is transformed into a graph, the cost of the edge between each node is fixed because the cost between each nodes does not depend on other agents but the original assigned agent. Thus the ABF algorithm combined with the Amortized search can always find the negative cycle in the graph. That is the algorithm can always find the optimal solution.

Eight typical assignment problems in [2] are used to validate the algorithm. The size of these problem are from 100 to 800. The cost c_{ij} are ranged from 1 to 100. These eight typical assignment problem are described as follow.

$$\min \sum_{i=1}^{N} \sum_{j=1}^{N} c_{ij} x_{ij} \tag{3.32}$$

Problem Size	Optimal Value	Value found by the Algorithm
100	305	305
200	475	475
300	626	626
400	804	804
500	991	991
600	1176	1176
700	1362	1362
800	1552	1552

Table 3.1: Result on the Assignment Problem

subject to

$$\sum_{i=1}^{N} x_{ij} = 1 \; \forall j \tag{3.33}$$

$$\sum_{j=1}^{N} x_{ij} = 1 \; \forall i \tag{3.34}$$

$$x_{ij} \in \{0, 1\} \tag{3.35}$$

Table 3.1 shows the result of the algorithm runs on these eight problems. As expected, the algorithm can find the optimal solution in the eight selected problems in [2]. More performance analysis of the algorithm can be found in chapter 4 and chapter 5.

□ **End of chapter.**

Chapter 4

Goal Programming Model for Channel Assignment Problem

The algorithm described in section 3.4 can solve a wide range of problems including the generalised channel assignment problem stated in section 2.1. The following section will extend the channel assignment problem by a goal programming model and will use the algorithm in section 3.4 to solve it [13].

4.1 Motivation

In the third generation and future wireless system, more and more data applications will be developed. Quality of service (QoS) requirements have a larger variations than previous generation wireless system. In order to meet the QoS more effectively, new resource allocation schemes should be introduced.

Traditionally, throughput is the main measurement of the channel assignment schemes. Many proposed schemes try to optimize the total system throughput and ignore the actual requirement of users application. Some users are not able to obtain their desired throughput while other users obtain a throughput far beyond their needs. This results an unfair and inefficient channel assignment.

Sometimes, there may not be enough channels to satisfy all the users QoS. In this case, admission control can be used to solve this problem. A user is accepted only when its QoS can be satisfied and it's admission does not affect others performance. Another way to solve this problem is to seek for a compromise solution. User requirements are not included in the system constraints but are treated as a measurement of the performance of the channel assignment. This approach is more flexible and can be generalised in a goal programming model. Moreover, algorithm in section 3.4 can be used to solve this problem.

4.2 System Model

We consider a multicarrier CDMA system with N channels and K clients. Each channel is orthogonal to one another, that is, the multiple access interference (MAI) is zero. For each client i, at most n_i channels can be assigned to him/her. On the other hand, each channel can only be assigned to at most one client. These two

constraints can be written as

$$\sum_{j=1}^{N} x_{ij} \leq n_i, \ \forall i, \tag{4.1}$$

$$\sum_{i=1}^{K} x_{ij} \leq 1, \ \forall j. \tag{4.2}$$

For each channel j to each client i, we define a value called quality index, R_{ij}. This value describes the quality of the j-th channel enjoyed by the i-th client. One example of the quality index is the throughput obtained by client i when channel j is assigned to him/her. The choice of the quality index depends on the application of each client.

For each client i, we define the unsatisfactory function, $d_i^+(\sum_{j=1}^{N} R_{ij}x_{ij})$ and the bonus function, $d_i^-(\sum_{j=1}^{N} R_{ij}x_{ij})$. The unsatisfactory function is a monotonic decreasing function of $\sum_{j=1}^{N} R_{ij}x_{ij}$ while the bonus function is a monotonic increasing function of $\sum_{j=1}^{N} R_{ij}x_{ij}$. These functions are based on the user application.

The following is an example of the unsatisfactory function and the bonus function and will be used as the objective function in section 4.4.

$$d_i^+(\sum_{j=1}^{N} R_{ij}x_{ij}) = max\left\{D_i - \sum_{j=1}^{N} R_{ij}x_{ij}, 0\right\} \tag{4.3}$$

$$d_i^-(\sum_{j=1}^{N} R_{ij}x_{ij}) = max\left\{\sum_{j=1}^{N} R_{ij}x_{ij} - D_i, 0\right\} \tag{4.4}$$

where D_i is the demand throughput of user i.

4.3 Goal Programming Model for Channel Assignment Problem

For client i, there are two objective functions:

$$\min d_i^+(\sum_{j=1}^{N} R_{ij}x_{ij}) \tag{4.5}$$

$$\max d_i^-(\sum_{j=1}^{N} R_{ij}x_{ij}). \tag{4.6}$$

Since these two objective functions conflict with each other, we use the goal programming model to resolve this conflict.

$$\text{lexmin } \vec{u} = \left(\sum_{i=1}^{K} \beta_i d_i^+(\sum_{j=1}^{N} R_{ij}x_{ij}), -\sum_{i=1}^{K} \beta_i d_i^-(\sum_{j=1}^{N} R_{ij}x_{ij})\right)^T \tag{4.7}$$

subject to

$$\sum_{j=1}^{N} x_{ij} \leq n_i, \ \forall i, \tag{4.8}$$

$$\sum_{i=1}^{K} x_{ij} \leq 1, \ \forall j, \tag{4.9}$$

and

$$x_{ij} = \begin{cases} 1 & \text{if channel j is assigned to user i,} \\ 0 & \text{otherwise.} \end{cases} \tag{4.10}$$

where β_i are predefined constants [13].

In the above model, we use the lexicographic minimum [10] as the objective function of this channel assignment problem. The unsatisfactory function is the first term in $\vec{u}$ which means that it is more important than the bonus function. This model

tries to satisfy all the clients demand first before maximizing the extra throughput that can be assigned.

The goal programming model of the channel assignment problem is exactly the same as the multi-objective assignment problem in section 3.1, so it is an NP-hard problem.

4.4 Simulation

To solve the above problem, algorithm in section 3.4 is used. In order to compare the performance of the algorithm, we introduce two different methods in finding the channel assignment. These two methods are the traditional throughput optimization approach and the low complexity greedy approach.

The quality index, R_{ij}, is defined to be the throughput, which is described in section 2.1.1. The demand, D_i, of each user is randomly generated. β_i is equal to 1 for all i. And we use two different sets of unsatisfactory function and bonus function. The first set is

$$d_i^+(\sum_{j=1}^{N} R_{ij}x_{ij}) = max\left\{D_i - \sum_{j=1}^{N} R_{ij}x_{ij}, 0\right\} \tag{4.11}$$

$$d_i^-(\sum_{j=1}^{N} R_{ij}x_{ij}) = max\left\{\sum_{j=1}^{N} R_{ij}x_{ij} - D_i, 0\right\} \tag{4.12}$$

and the second set is

$$d_i^+(\sum_{j=1}^{N} R_{ij}x_{ij}) = \begin{cases} D_i & \text{if } D_i - \sum_{j=1}^{N} R_{ij}x_{ij} > 0 \\ 0 & \text{otherwise} \end{cases} \quad (4.13)$$

$$d_i^-(\sum_{j=1}^{N} R_{ij}x_{ij}) = max\left\{\sum_{j=1}^{N} R_{ij}x_{ij} - D_i, 0\right\} \quad (4.14)$$

The first unsatisfactory function measures the total throughput that has not been satisfied for all users. The second unsatisfactory function sums the total demand of users who has not been satisfied by the assignment. The bonus function are the same and is equal to the extra throughput enjoyed by the clients.

4.4.1 Throughput Optimization

By ignoring the objective function defined in equation 4.7, this method aims at maximizing the total throughput of the system. The throughput optimization problem can be written as follows:

$$\max \sum_{i=1}^{K}\sum_{j=1}^{N} R_{ij}x_{ij} \quad (4.15)$$

subject to

$$\sum_{j=1}^{N} x_{ij} \leq n_i, \ \forall i, \quad (4.16)$$

$$\sum_{i=1}^{K} x_{ij} \leq 1, \ \forall j. \quad (4.17)$$

Since this is a linear programming problem, it can be solved readily by different methods. We include the throughput optimization as a comparison in this selected application because total system throughput is usually used in measuring the

performance of the assignment in wireless network. Although new generation wireless network should satisfy individual users QoS, which may be quite different from throughput, we want to compare the performance of traditional approach to the new problem.

4.4.2 Best-First-Assign Algorithm

The Best-First-Assign algorithm is a greedy approach and is described as follows:

1. For each client i, sort the value R_{ij} in descending order.
2. For each client i, evaluate the reduction of d_i^+ if a unassigned channel with highest value of R_{ij} is assigned to him/her.
3. Among these clients, choose the client has the greatest reduction calculated in step 2 and perform the corresponding channel assignment.
4. If there are still some channels can be assigned to the clients, go to step 2.

The complexity of sorting in step 1 is $O(N \log N)$ and that of step 2 and 3 are $O(K)$. So the time complexity of the above algorithm is $O(NK)$. Greedy approach is selected as a comparison to the algorithm because of its simplicity and it can be adopted to different QoS measurement easily in this application.

4.4.3 Channel Swapping Algorithm

We use the algorithm in section 3.4 to solve this channel assignment problem. Since the algorithm tries to swap channel between users to search for a better channel

assignment, we give it the name Channel Swapping Algorithm in this special application. The complexity of this algorithm is highest among the algorithm which will be used for comparison.

In order to transform the problem into a graph correctly, we have defined the function $f_i(\vec{x_i})$ which map a vector into a value in section 3.3. In this example, we let $\epsilon_1 = 1$ and ϵ_2 changes according to the channel throughput and clients demand.

Let δ_{min} be the lower bound of the changes for the first objective function $d_i^+(\sum_{j=1}^{N} R_{ij}x_{ij})$ and δ_{max} be the upper bound of the changes for the second objective function $d_i^-(\sum_{j=1}^{N} R_{ij}x_{ij})$. In this simulation, we try to estimate the value δ_{min} by the difference of the channel throughput.

Let Θ be the set of values of channel throughput difference and the values of channel throughput.

$$\Theta = \{\delta : \delta = |R_{ij} - R_{ik}|, \forall i, j, k\} \cup \{\delta : \delta = |R_{ij}|, \forall i, j\}. \tag{4.18}$$

We can find δ_{min} and δ_{max} by the following method.

$$\delta_{min} = \min\{\Theta - \{0\}\} \tag{4.19}$$

and

$$\delta_{max} = \sum_{i=1}^{K}\sum_{j=1}^{N} R_{ij}. \tag{4.20}$$

After obtaining δ_{min} and δ_{max}, we can find ϵ_2 by

$$\epsilon_2 = \frac{\delta_{min}}{c\,\delta_{max}} \tag{4.21}$$

where c is a constant which is larger than 1. By selecting ϵ_2 according to equation 4.21, the value $\epsilon_2 d_i^-(\sum_{j=1}^{N} R_{ij}x_{ij})$ does not affect the choice of assignment which is minimizing the value $d_i^+(\sum_{j=1}^{N} R_{ij}x_{ij})$.

4.4.4 Lower Bound

The lower bound is obtained in two different cases. When $N > K$ and $n_i > 1$, the lower bound is obtained by ignoring equation 4.9. That is each user will be assigned the best channels. Obviously, this gives the lower bound of the weighted sum of the unsatisfactory functions. When $N \leq K$ and $n_i = 1$, the problem will become a typical assignment problem and the optimal solution can be found by linear programming algorithms. This optimal solution will serve as a lower bound to the channel assignment problem.

4.4.5 Result

We define the proportion of deficient throughput as the weighted sum of unsatisfactory function divided by the weighted sum of unsatisfactory function without assigning any channels. In this example, we can express the proportion of deficient throughput as

$$\eta = \frac{\sum_{i=1}^{K} \beta_i d_i^+(\sum_{j=1}^{N} R_{ij}x_{ij})}{\sum_{i=1}^{K} D_i}. \tag{4.22}$$

We would like to compare how the performance varies with the number of channels and the number of clients.

Figure 4.1 and 4.2 shows the performance when the number of channels is varying.

The number of clients is fixed at 16 while the number of channels is increasing from 16 to 64. As expected, the proportion of deficient decreases as the number of channels increases. This is because each client can have more channels when the number of channels increases.

Although the Best-First-Assign algorithm is simple, it works better than the traditional throughput optimization approach. In throughput optimization approach, clients needs are ignored. It is often that some clients have a total throughput far beyond their needs while the others do not have enough throughput. So resources are wasted in throughput optimization approach and the performance is worse than the greedy approach. The Channel Swapping Algorithm always achieve a better solution than the Best-First-Assign algorithm and the throughput optimization approach. Since the complexity of the Channel Swapping algorithm is higher, it use extra time to search for a better solution. As expected, the algorithm found the optimal solution when $N = K$.

The improvement made by the Channel Swapping algorithm over the Best-First-Assign algorithm and the throughput optimization approach is largest when the number of channels is double of the number of clients. The improvement diminishes when the number of channels further increases. Channel Swapping algorithm works better because it tries to avoid the case where a client gets a throughput which is far beyond its needs. When the channels are more than enough, throughput can be wasted in the above case with little effect on the unsatisfactory function because

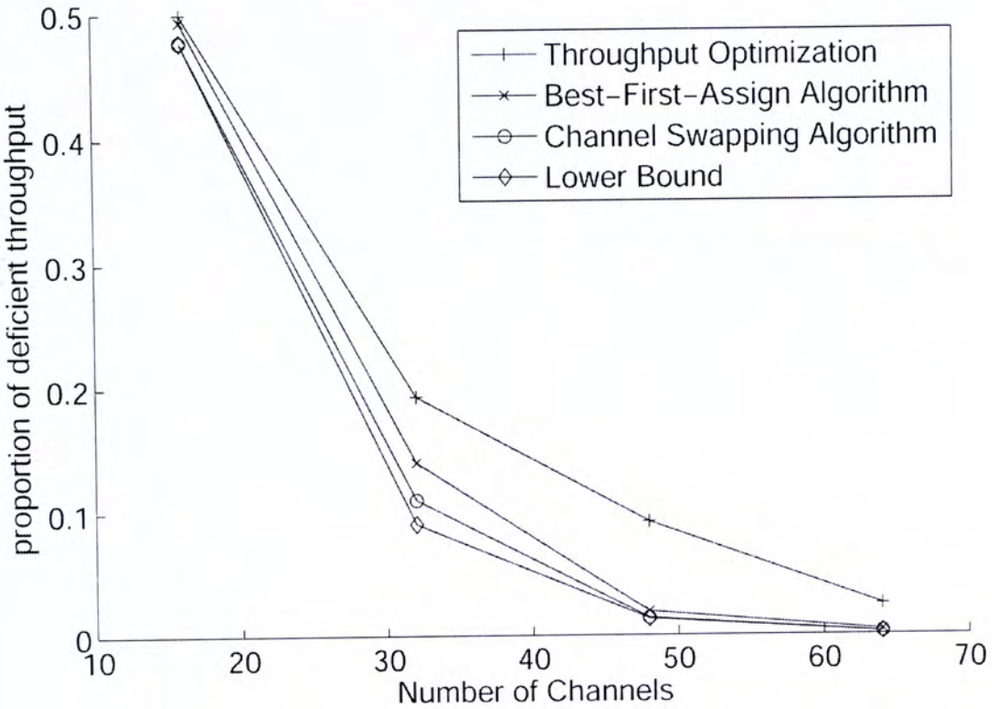

Figure 4.1: Relationship between the proportion of deficient throughput and number of channels (unsatisfactory function 1)

each client can eventually get enough throughput by having more channels.

Figure 4.3 and 4.4 shows the performance when the number of clients is varying. The number of channels is fixed at 16 while the number of clients is increasing from 8 to 32 in figure 4.3 and from 16 to 64 in figure 4.4. The proportion of deficient throughput increases when the number of clients increases. This is because there are not enough channels to support all the clients. Once again, the Channel Swapping algorithm always achieve a better solution than the Best-First-Assign algorithm and the throughput optimization approach. When $N \geq 16$, the algorithm always find

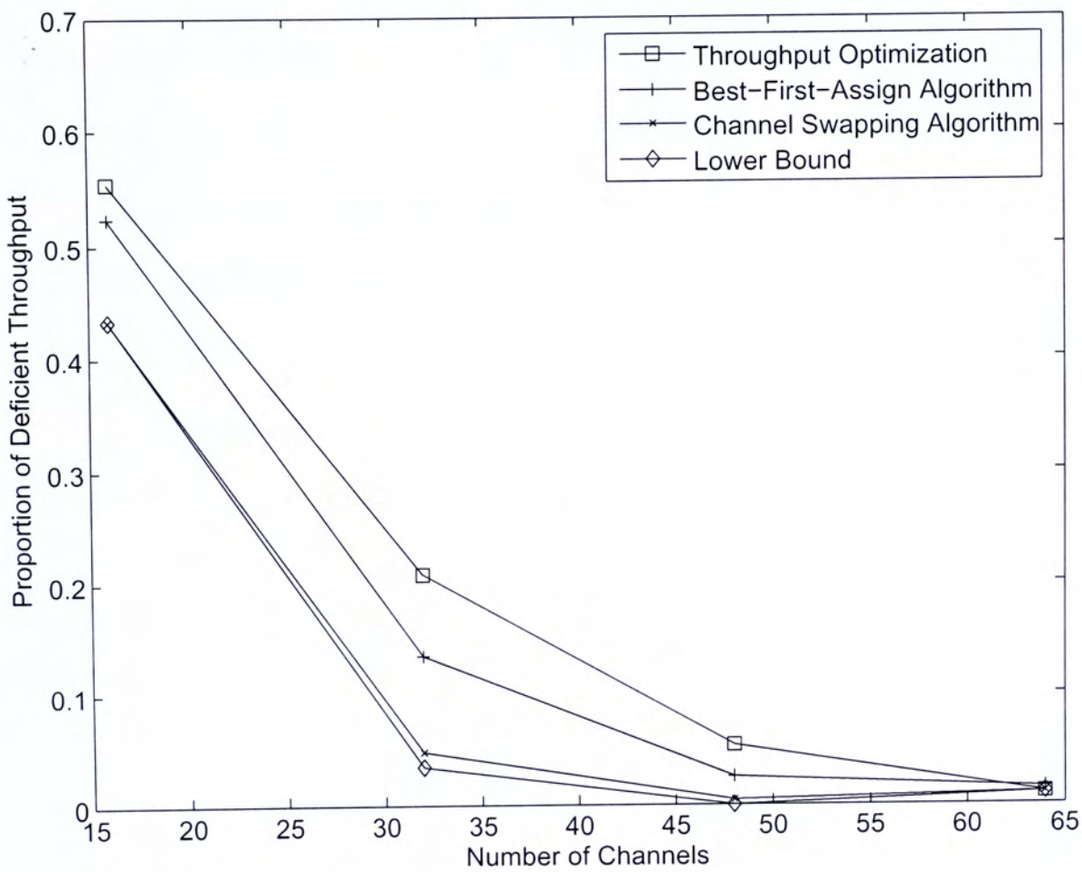

Figure 4.2: Relationship between the proportion of deficient throughput and number of channels (unsatisfactory function 2)

the optimal solution since the problem is a typical assignment problem.

The improvement made by the Channel Swapping algorithm over the Best-First-Assign algorithm and the throughput optimization approach decreases when the number of clients further increases. In this case, total channel throughput is much less than the total demand, thus the chance of assigning a good channel to a user with low demand is small and the improvement can be made is small. In the extreme, when the demand of each user is much more than the throughput that the system can support, the throughput optimization approach is already the best solution. Because the demand of each user is very large, the unsatisfactory function can be simplified as

$$d_i^+(\sum_{j=1}^{N} R_{ij}x_{ij}) = D_i - \sum_{j=1}^{N} R_{ij}x_{ij} \tag{4.23}$$

since $D_i - \sum_{j=1}^{N} R_{ij}x_{ij}$ is always larger than zero and the objective function can be rewritten as

$$\begin{aligned} \min d_i^+(\sum_{j=1}^{N} R_{ij}x_{ij}) &= \min(D_i - \sum_{j=1}^{N} R_{ij}x_{ij}) && (4.24)\\ &= \min(-\sum_{j=1}^{N} R_{ij}x_{ij}) && (4.25)\\ &= \max(\sum_{j=1}^{N} R_{ij}x_{ij}), && (4.26) \end{aligned}$$

which is the same as the throughput optimization problem.

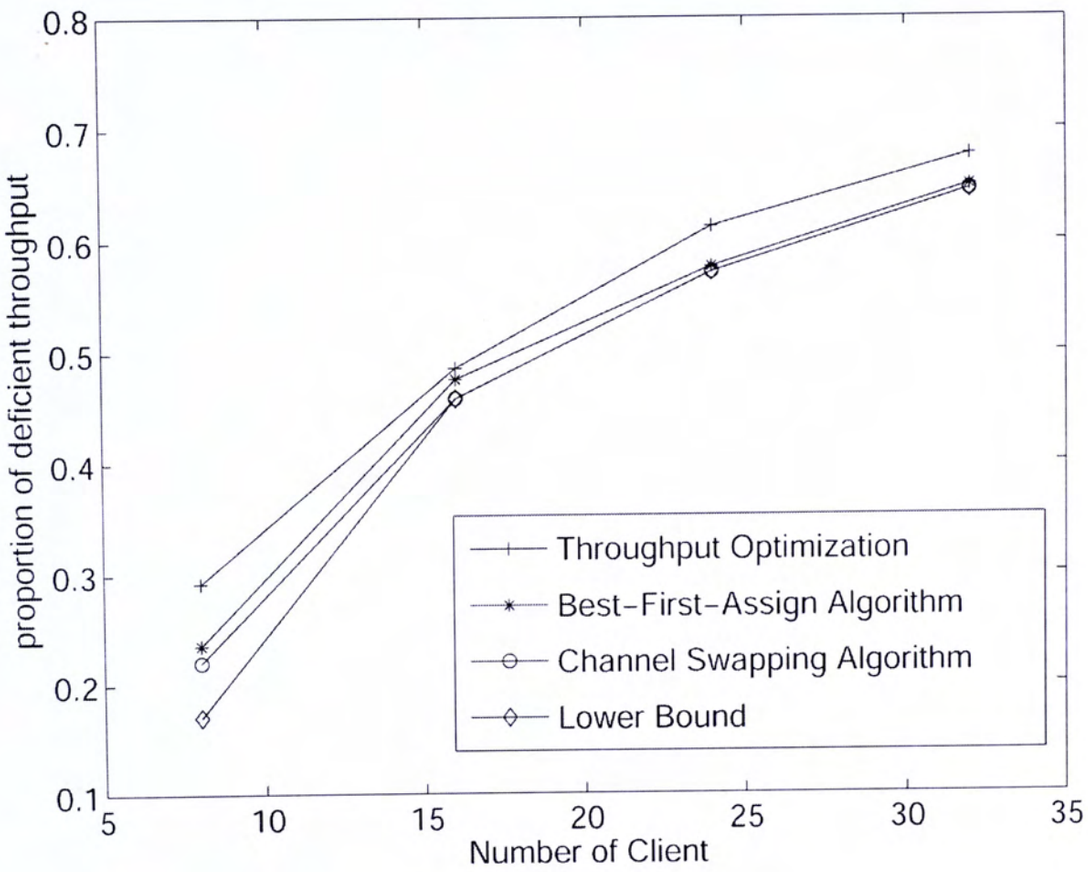

Figure 4.3: Relationship between the proportion of deficient throughput and number of clients (unsatisfactory function 1)

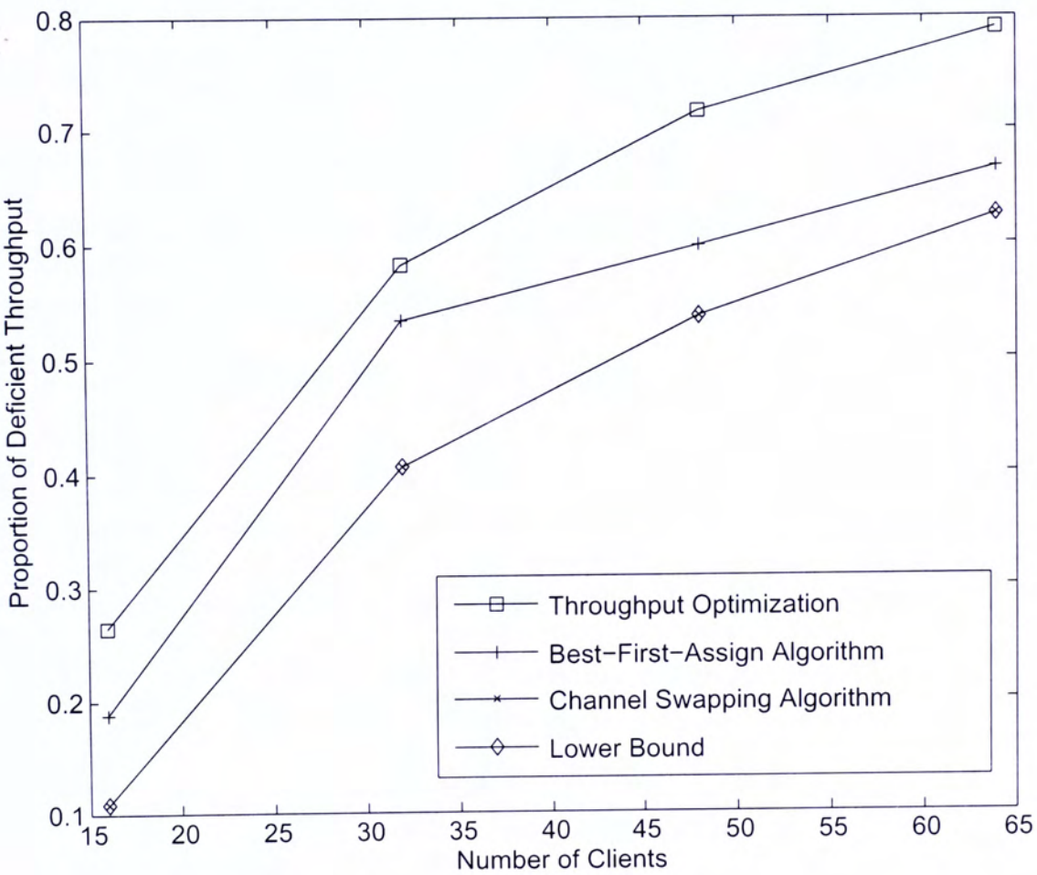

Figure 4.4: Relationship between the proportion of deficient throughput and number of clients (unsatisfactory function 2)

4.5 Future Works

The goal programming model for channel assignment problem is a very general model, so it can be easily extended to model the same problem in multiple time slots or multi-cellular environment.

One special case in multiple time slots environment has been solved in [9]. It considers a transmission time minimization in a multicarrier CDMA data network. Each users in the data network have a packet to send. The objective is to minimize the total transmission time slots to send all the packet. In this special case, the problem can be transformed into a linear programming problem and the solution can be obtained by mathematical software.

Other users dependent objective can also be employed in the multiple time slots environment, but it will greatly increase the complexity of that problem. So more effort have to be put in these kind of problem in order to solve it.

□ **End of chapter.**

Chapter 5

Extended Application on the General Problem

The algorithm in section 3.4 has a wide range of application. In this section, we will extend its application on some general problems. Extending the channel assignment problem in section 2.1 and chapter 4, we would like to measure the performance on different objective function. One of them is the latency, which is defined as the time to transmit a data packet.

Because of the similarity between the multi-objective assignment problem, the generalised assignment problem and the quadratic assignment problem, we have also applied the algorithm on these two problems.

5.1 Latency Minimization

Latency minimization is one of the interested problem in the channel assignment problem. Client i in the wireless network has a data packet of length D_i to send. The objective is to minimize the total time required to send out all the packets. we define the latency of client i as

$$L_i = \frac{D_i}{\sum_{j=1}^{N} R_{ij} x_{ij}} \tag{5.1}$$

where R_{ij} is the throughput enjoyed by client i when channel j is assigned to it. The objective of this problem is to minimize the total latency

$$\min \sum_{i=1}^{M} L_i = \sum_{i=1}^{M} \frac{D_i}{\sum_{j=1}^{N} R_{ij} x_{ij}} \tag{5.2}$$

subject to

$$\sum_{i=1}^{M} x_{ij} \leq 1, \ \forall j \tag{5.3}$$

$$\sum_{j=1}^{N} x_{ij} \leq n_i, \ \forall i \tag{5.4}$$

$$x_{ij} \in \{0, 1\}. \tag{5.5}$$

The latency minimization problem is a special case of the multi-objective assignment problem with single objective function. We can formulate the latency minimization problem into a multi-objective assignment problem where $f_i^{(1)}(\vec{x_i}) = \frac{D_i}{\sum_{j=1}^{N} R_{ij} x_{ij}}$. We found that traditional approach of channel assignment is not working well when we want to minimize the total latency of the system, so we try to apply the algorithm in section 3.4 to tackle this problem.

Figure 5.1 shows the relationship between the total latency and the number of channels. In the simulation, there are 16 clients in the wireless network. The number of channels is increasing from 16 to 64. The total latency decreases when the number of channels increases. This is because when the number of channels is more than the number of clients, each client can enjoy more channels, and thus the throughput of each client increases and the latency of each client decreases.

The throughput optimization approach and the greedy approach are included for comparison. These two approaches are described in section 4.4.1 and section 4.4.2 respectively. A lower bound is also included which ignore the constraints in equation 5.3. When the number of channels is small, the Channel Swapping Algorithm make a huge improvement over the throughput optimization approach and the greedy approach. However, when the number of channels increases, the improvement made by it diminished. This is because the average latency is already very small when the number of channels is greater than the number of users. The improvement can be made by the algorithm becomes small.

5.2 Generalised Assignment Problem

Generalised Assignment Problem (GAP) is a well known strongly NP-hard problem [8]. In GAP, the objective is to find a minimum costs assignment of M jobs to N agents such that each agent is assigned to exactly one job and does not exceed the resources of each job. Let $I \in \{1, \ldots, N\}$ be the set of jobs and $J \in \{1, \ldots, M\}$

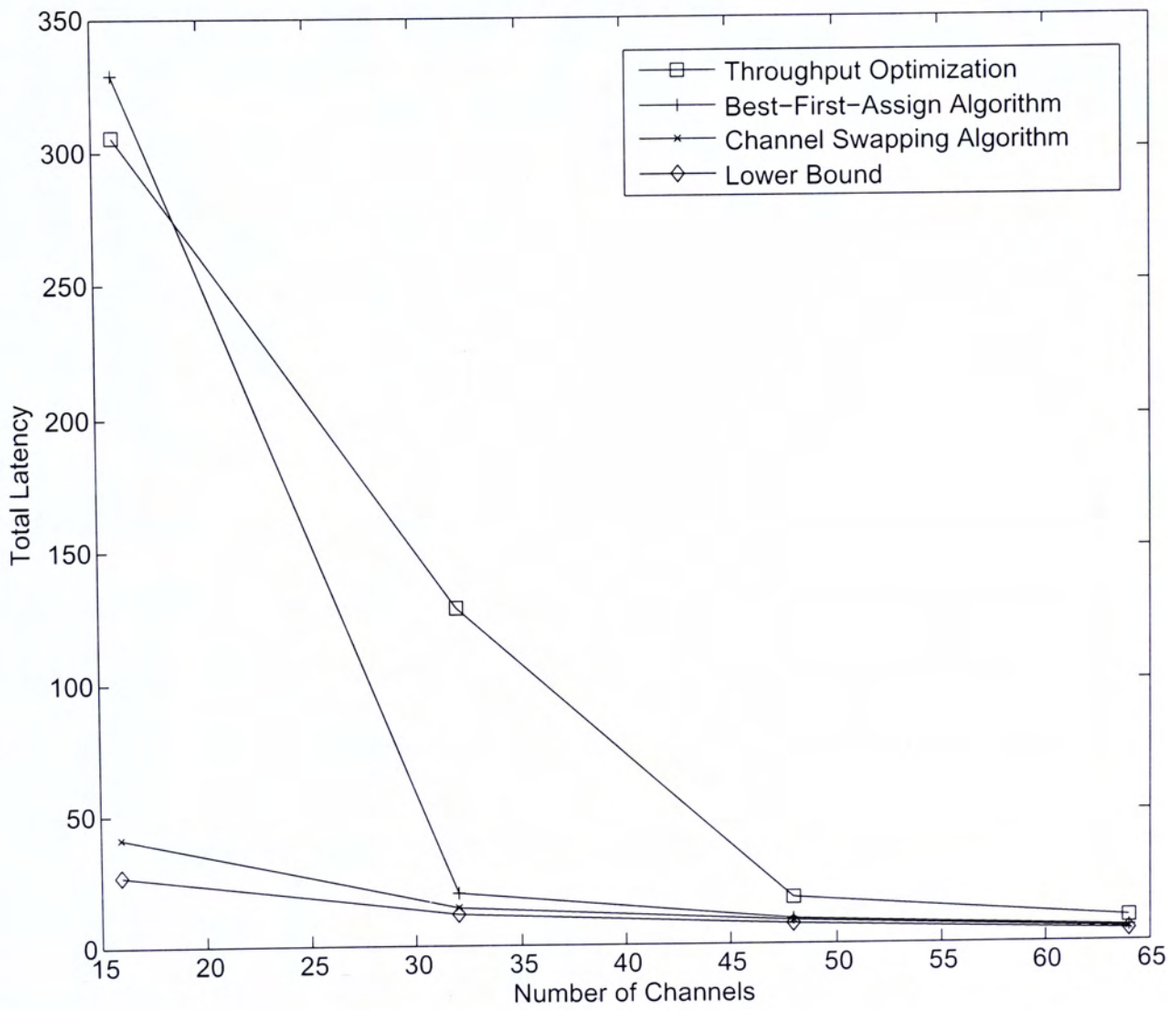

Figure 5.1: Relation between Total Latency and Number of Channels

be the set of agents. For each job i, we are given the resources capacity b_i. For each $i \in I$ and each $j \in J$, the costs c_{ij} and the resource requirement r_{ij} is given for assigning agent j to job i. If agent j is assigned to job i, then $x_{ij} = 1$ and 0 otherwise. The objective is to minimizes the total costs

$$\min c(x) = \sum_{i=1}^{M} \sum_{j=1}^{N} c_{ij} x_{ij} \tag{5.6}$$

subject to

$$\sum_{j=1}^{N} r_{ij} x_{ij} \leq b_i, \ \forall i \in I, \tag{5.7}$$

$$\sum_{i=1}^{M} x_{ij} = 1, \ \forall j \in J. \tag{5.8}$$

By modify the GAP into a multi-objective assignment problem, the algorithm in section 3.4 can be used to solve the GAP. We use the same method in section 3.2 to construct a new multi-objective assignment problem which is equivalent to the GAP. The objective of this new multi-objective assignment problem is as follows:

$$\text{lexmin } \vec{v} = (v_1, v_2) = \left(\sum_{i=1}^{M} u(\sum_{j=1}^{N} r_{ij} x_{ij} - b_i), \sum_{i=1}^{M} \sum_{j=1}^{N} c_{ij} x_{ij} \right). \tag{5.9}$$

subject to

$$\sum_{j=1}^{N} x_{ij} \leq n_i, \ \forall i \in I, \tag{5.10}$$

$$\sum_{i=1}^{M} x_{ij} = 1, \ \forall j \in J. \tag{5.11}$$

If the resources constraints is satisfied, v_1 is equal to zero since $\sum_{j=1}^{N} r_{ij} x_{ij} < b_i, \forall i \in I$. If the resources constraints is not satisfied, v_1 is larger than 0, so a

feasible solution in more preferable than an infeasible solution in the multi-objective assignment problem. A feasible solution with a lower cost will have a more preferable $\vec{v}$. Consider two feasible assignments in the GAP with costs c_1 and c_2, these two assignments produce $(0, c_1)$ and $(0, c_2)$ in the multi-objective assignment problem. Since $c_1 < c_2$, so $(0, c_1)$ is preferred to $(0, c_2)$ and the assignment with smaller cost is more preferable in the multi-objective assignment problem.

Problem set from [1] is used to evaluate the performance of the algorithm. This problem set consists of 60 GAP with known optimal solution. Since these problem are the maximization problem, we have changed them into the minimization problem by adding a minus sign to the objective function, and expressed as

$$\min c(x) = \sum_{i=1}^{M} \sum_{j=1}^{N} -c_{ij} x_{ij}. \tag{5.12}$$

c515-1 denotes problem 1 with 5 jobs and 15 agents. This program is implemented in MATLAB and run on a Pentium4 2.66GHz computer. The % gap is calculated as the difference between the optimal value C_{opt} and the value found by the algorithm C_a. That is

$$\% \text{ gap } = \frac{\|C_a - C_{opt}\|}{C_{opt}} \times 100. \tag{5.13}$$

Table 5.1, table 5.2 and table 5.3 shows the result of the algorithm applied on these problem sets. An initial solution is constructed randomly and uniformly. The algorithm performs well in small size problems and finds the exact solution. For larger and more complex instances, the algorithm also finds a sub-optimal solution which is very close to the optimal solution.

Problem set	C_{opt}	C_a	Time (sec)	% gap
c515-1	336	336	8.782	0
c515-2	327	327	8.047	0
c515-3	339	339	10.516	0
c515-4	341	341	15.969	0
c515-5	326	326	14.75	0
c520-1	434	434	20.891	0
c520-2	436	436	24.187	0
c520-3	420	418	20.062	0.476190476
c520-4	419	416	50.109	0.715990453
c520-5	428	428	22.75	0
c525-1	580	574	54.89	1.034482759
c525-2	564	558	55.672	1.063829787
c525-3	573	572	36.609	0.17452007
c525-4	570	570	57.532	0
c525-5	564	562	55	0.354609929
c530-1	656	656	90.75	0
c530-2	644	644	126.72	0
c530-3	673	673	150.59	0
c530-4	647	647	53.735	0
c530-5	664	661	108.67	0.451807229

Table 5.1: Result on the Generalised Assignment Problem Set (Number of Job = 5)

Problem set	C_{opt}	C_a	Time (sec)	% gap
c824-1	563	563	143.67	0
c824-2	558	558	142.25	0
c824-3	564	564	100.8	0
c824-4	568	567	61.922	0.176056338
c824-5	559	558	82.109	0.178890877
c832-1	761	761	138.97	0
c832-2	759	758	166.49	0.131752306
c832-3	758	758	241.28	0
c832-4	752	752	146.86	0
c832-5	747	747	274.16	0
c840-1	942	942	440.73	0
c840-2	949	949	635.47	0
c840-3	968	968	638.69	0
c840-4	945	945	476.75	0
c840-5	951	950	453.36	0.105152471
c848-1	1133	1131	616.89	0.176522507
c848-2	1134	1131	405.92	0.264550265
c848-3	1141	1138	556.42	0.262927257
c848-4	1117	1113	446.28	0.358102059
c848-5	1127	1124	429.31	0.266193434

Table 5.2: Result on the Generalised Assignment Problem Set (Number of Job = 8)

Problem set	C_{opt}	C_a	Time (sec)	% gap
c1030-1	709	708	172.88	0.141043724
c1030-2	717	715	156.58	0.278940028
c1030-3	712	711	211.89	0.140449438
c1030-4	723	723	354.23	0
c1030-5	706	705	215.86	0.141643059
c1040-1	958	957	360.7	0.104384134
c1040-2	963	961	486.66	0.20768432
c1040-3	960	957	502.58	0.3125
c1040-4	947	944	336.31	0.316789863
c1040-5	947	944	592.44	0.316789863
c1050-1	1139	1139	1075.5	0
c1050-2	1178	1178	1141.2	0
c1050-3	1195	1195	1082.3	0
c1050-4	1171	1167	1201.1	0.341588386
c1050-5	1171	1167	1183.4	0.341588386
c1060-1	1451	1449	1150.2	0.137835975
c1060-2	1449	1449	1233.2	0
c1060-3	1433	1433	1469.9	0
c1060-4	1447	1444	955.01	0.207325501
c1060-5	1446	1444	1261.6	0.138312586

Table 5.3: Result on the Generalised Assignment Problem Set (Number of Job = 10)

5.3 Quadratic Assignment Problem

Quadratic assignment problem (QAP) is one of the NP-hard problem. QAP is a question of locating facilities to different locations. There is a distance between each location and a flow between each facility. The question is to minimize the cost, which is defined by multiplying the flow and the distance. The objective function of QAP can be expressed as follow:

$$\min C(p) = \sum_{i=1}^{N} \sum_{j=1}^{N} a_{ij} b_{p(i)p(j)} \tag{5.14}$$

where p is the corresponding permutation, a_{ij} and b_{ij} are the flow and the distance between facility i and j respectively [1].

We can also express the QAP into the form which is similar to most of the assignment problem [14]. We consider an assignment problem with the following objective function

$$\sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{k=1}^{N} \sum_{l=1}^{N} a_{ij} b_{kl} x_{ik} x_{jl}. \tag{5.15}$$

subject to

$$\sum_{i=1}^{N} x_{ij} = 1, \ \forall j \tag{5.16}$$

$$\sum_{j=1}^{N} x_{ij} = 1, \ \forall i. \tag{5.17}$$

and

$$x_{ij} = \begin{cases} 1 & \text{if facility i is assigned to location j} \\ 0 & \text{otherwise.} \end{cases} \tag{5.18}$$

The minimization problem defined from equation 5.15 to equation 5.18 is the same as that in equation 5.14. When facility i and j is assigned to location k and l respectively, that is $p(i) = k$ and $p(j) = l$, the contribution to the objective function is equal to $a_{ij}b_{p(i)p(j)}$. Since $x_{ik} = 1$ and $x_{jl} = 1$, $a_{ij}b_{p(i)p(j)}$ can be written into $a_{ij}b_{kl}x_{ik}x_{jl}$.

The objective function of QAP cannot decompose into the sum of individual assignment, that is it cannot rewrite into $\sum_{i=1}^{N}\left(f_i^{(1)}(\vec{x_i}), f_i^{(2)}(\vec{x_i}), \cdots\right)$. So QAP does not fall on the class of the multi-objective assignment problem described in section 3.1. However, we still want to evaluate the performance of the algorithm applied on the QAP because we found that QAP can model a special case in the channel assignment problem.

QAP can be used to formulate the channel assignment problem when the channel assignment problem is considered as a partition problem [14]. For example, partitioning the channels into different groups such that the dissimilarities between the channels in the same group is minimized.

To adopt the algorithm in section 3.4 to solve the QAP, we first need to transform the QAP problem into a dynamic directed graph. We consider the facilities as jobs and the locations as agents. Since each facility must be assigned to one and only one location, so each agent in the multi-objective assignment problem must be assigned to one job.

We consider each agent as a node in the constructed dynamic directed graph and

the cost of the edge is equal to the changes in the objective. $p(s)$ is equal to the job that has been assigned to the agent s. The cost of edge, $d_{s,t}$, in the graph is calculated if job $p(s)$ is assigned to agent t instead of agent s.

Let $F(\mathbf{X})$ be the whole objective function, that is

$$F(\mathbf{X}) = \sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{k=1}^{N}\sum_{l=1}^{N} a_{ij}b_{kl}x_{ik}x_{jl}. \tag{5.19}$$

Let $\mathbf{X}$ be an assignment matrix where job i is assigned to agent s, that is $x_{is} = 1$ in the assignment matrix $\mathbf{X}$. $\mathbf{X}$ can be a feasible or infeasible solution to the QAP. Let $\mathbf{X}'$ be another assignment matrix which is equal to $\mathbf{X}$ except that $x'_{is} = 0$ and $x'_{it} = 1$. Although $\mathbf{X}$ and $\mathbf{X}'$ may not be a feasible solution to the QAP, it still produces a value with the function $F(\mathbf{X})$ in equation 5.19. The cost of the edge from s to t is calculated as

$$d_{s,t} = F(\mathbf{X}') - F(\mathbf{X}). \tag{5.20}$$

After defining the nodes and the cost of the edge between nodes, we can apply the algorithm in section 3.4 to tackle this problem.

Problems in [3] has been used to analysis the performance of the algorithm in section 3.4. Problem set Had and Nug are used. Table 5.4 and Table 5.5 shows the result. An initial solution is obtained randomly and uniformly. The algorithm performs well in small size problem and finds a sub-optimal solution which is very close to the optimal solution.

□ **End of chapter.**

Problem	Size	C_{opt}	C_a	% gap
Had12	12	1652	1652	0
Had14	14	2724	2726	0.073
Had16	16	3720	3728	0.215
Had18	18	5358	5366	0.149
Had20	20	6922	6925	0.043

Table 5.4: Results on the Nad QAP Problem Set

Problem	Size	C_{opt}	C_a	% gap
Nug12	12	578	578	0
Nug14	14	1014	1014	0
Nug15	15	1150	1150	0
Nug16a	16	1610	1615	0.311
Nug16b	16	1240	1242	0.161
Nug17	17	1732	1736	0.231
Nug18	18	1930	1932	0.104
Nug20	20	2570	2571	0.039
Nug21	21	2438	2446	0.328
Nug22	22	3596	3599	0.083
Nug24	24	3488	3494	0.172
Nug25	25	3744	3752	0.214
Nug27	27	5234	5235	0.019
Nug28	28	5166	5167	0.019
Nug30	30	6124	6133	0.147

Table 5.5: Results on the Nug QAP Problem Set

Chapter 6

Conclusion

In this thesis, we have applied a search method which combine the Bellman-Ford-Moore algorithm with Amortized search negative cycle detection strategy to tackle the multi-objective assignment problem.

We first found out that the channel assignment problem in the multicarrier CDMA system can be generalised into a multi-objective assignment problem by the goal programming approach. The goal programming model works better in responding to the QoS requirement of individual user. Traditional throughput optimization approach is not working very well in new generation wireless network because it ignores the QoS requirement of each user. The new goal programming model considers the QoS of each user and tries to compromise them. This approach is more preferable in the new generation wireless system because the QoS requirement have larger variations than previous generation wireless system.

We introduce a search method to tackle the multi-objective assignment problem.

We observed that the multi-objective assignment problem can be transformed into a dynamic directed graph. And each negative cycle in the graph represents a movement from one feasible solution to another feasible solution with a lower cost. By finding out all the negative cycles in the graph, the assignment problem can be solved. So we consider a search method which combine the shortest path algorithm with the negative cycle detection strategy to solve the multi-objective assignment problem.

Although the suggested algorithm cannot always find the negative cycle in the graph, and thus cannot search for the optimal solution. We show that when this algorithm is applied on the channel assignment problem, it works good when comparing with throughput optimization approach and the greedy approach.

We shows that the suggested algorithm can always found the optimal solution of a special case of the multi-objective assignment problem, where the objective function is a linear function. In this special case, the multi-objective assignment problem is reduced into a linear programming problem. We also shows that the suggested algorithm can always found the optimal solution when the number of jobs is equal to the number of agents.

We also applied the suggested algorithm to the GAP, which is an NP-hard problem. It works well in small size problem and provides a sub-optimal solution in large size problem with a solution very close to the optimal solution.

□ **End of chapter.**

Bibliography

[1] J. E. Beasley. Or library. In *http://people.brunel.ac.uk/ mastjjb/jeb/info.html.*

[2] J. E. Beasley. Linear programming on cray supercomputers. In *Journal of the Operational Research Society*, volume 41, 1990.

[3] R. E. Burkard, E. Cela, S. E. Karisch, and F. Rendl. Qaplib - a quadratic assignment problem library. In *http://www.opt.math.tu-graz.ac.at/qaplib/inst.html.*

[4] N. Chandrachoodan, S. S. Bhattacharyya, and K. R. Liu. Adaptive negative cycle detection in dynamic graphs. In *International Symposium on Circuits and Systems, Sydney, Australia*, pages 163–166, May 2001.

[5] B. V. Cherkassky and A. V. Goldberg. Negative cycle detection algorithms. In *NEC Research Institute, Inc.*, March 1996.

[6] P. C. H. Chu. *A Genetic Algorithm Approach for Combinatorial Optimisation Problems.* PhD thesis, Imperial College of Science, Technology and Medicine, London, U.K., 1997.

[7] H. Feltl and G. R. Raidl. An improved hybrid genetic algorithm for the generalised assignment problem. In *2004 ACM Symposium on Applied Computing*, March 2004.

[8] M. R. Garey and G. S. Johnson. *Computers and Intractability: A guide to the Theory of NP-completeness.* W.H. Freeman, San Francisco, 1979.

[9] F. Hu, H. M. Tse, and T. M. Lok. Minimization of transmission time for multicarrier cdma data networks. In *Vehicular Technology Conference*, Sept 2004.

[10] J. P. Ignizio. *Introduction to Linear Goal Programming.* SAGE Publications, Inc., 1985.

[11] T. M. Lok. Downlink code assignment for throughput or latency optimization in multicarries cdma systems. In *Personal, Indoor and Mobile Radio Communications, 2003. PIMRC 2003.*, Sept, 7-10 2003.

[12] S. Martello and P. Toth. An algorithm for the generalised assignment problem. In *Operational Research*, volume 81, 1981.

[13] C. Y. Ng, H. M. Tse, and T. M. Lok. A goal programming model and schemes for channel assignment in general downlink transmission system. In *International Conference on Communications*, May 2005.

[14] P. M. Pardalos and H. Wolkowicz. *Quadratic Assignment and Related Problems.* American Mathematical Society, 1994.

[15] A. Schrijver. *Combinatorial optimization : polyhedra and efficiency.* Springer, New York :, 2003.