ALGORITHMS FOR THE RESOURCE LEVELLING PROBLEM

Allen R. Mushi

A thesis submitted in fulfilment of the requirements for the degree of

PhD. in Computer Applications

School of Computer Applications

Dublin City University

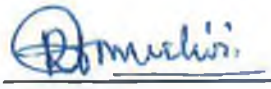
Dublin 9.

Supervisor: Dr. Micheál O'hEigeartaigh

September 1997

Declaration

I hereby certify that this material, which I now submit for assessment on the programme of study leading to the award of PhD. is entirely my own work and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my wok.

Signed:  Date: SEPTEMBER, 1997

Dublin City University

Abstract

ALGORITHMS FOR THE RESOURCE LEVELLING PROBLEM

The Resource Levelling Problem (RLP) is a variation of resource-constrained scheduling problems with major applications in manufacturing. A critical path is determined for a set of jobs/items that needs to be manufactured or assembled to form a final product. In capacity planning, each of these jobs is associated with a predetermined load or 'hassle' factor (resource level). When these jobs are scheduled together in a planning horizon, different profiles with respect to the resource levels are developed. The objective of the RLP is to minimise the maximum resource peaks by moving jobs within their slack times.

RLP is an NP-Hard combinatorial optimisation problem and therefore, no optimal algorithm is known. A few studies have been done on this problem during 1960's and early 1970's and various heuristics suggested (see [W64], [D66], [BK62], [D73]). However, no recent studies have been done despite recent developments in algorithmic techniques.

The objective of this project is to develop new algorithmic approaches to the RLP. In Chapter 1 we provide a definition of the problem using an example from the manufacturing industry. Chapter 2 presents a review of pre-processing techniques, which are useful in improving the performance of Integer Programming methods. In Chapter 3 we develop four Mixed Integer Programming (MIP) formulations. Our analysis shows that *time-indexed* formulations perform better for the RLP.

In Chapters 4 and 5 we present three global heuristic methods; Simulated Annealing, Tabu Search and a Perturbation algorithm. We conclude that, these algorithms are feasible and good approaches to the RLP. The perturbation algorithm was found to perform better than the rest. Chapter 6 provides a discussion on how the theory of Human-Computer Interaction can be used to improve the solution of the RLP by providing a good Interface design. Lastly we summarise our project in Chapter 7 and suggest areas of further research.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I would like to extend my heartfelt gratitude to my supervisor Dr. Micheál O'hEigeartaigh for all his help and guidance throughout the period of my research. Without his assistance I would not have been able to reach this stage.

My sincere gratitude to Prof. E. O'Kelly for his invaluable suggestions, especially on the application of Human Computer Interaction theory.

I would like to thank Prof. Michael Ryan and the School of Computer applications in general for providing me with the opportunity and the necessary funding to undertake this research.

Many thanks to Professor Lawrence Wolsey, the director of the Centre for Operations Research and Econometrics (CORE) for inviting me to his research centre in Brussels. My study tour in CORE has been very useful to my research. His advice and support in my project are highly appreciated.

Special thanks to Prof. Martin Savelsberg for his advice especially on the MIP formulation of my research problem.

I'm indebted to Dr. Robert Daniel, the director of Dash Associates, a mathematical Programming software company. His free copy of the EXPRESS-MP, a powerful mathematical programming package, have been a big boost on my research.

I'm deeply grateful to Mrs. Maev Maguire and Mr. Eoin Dunne of D.I.T Kevin Street. They have been my advisors and great friends throughout my stay in Ireland.

Finally my great appreciation and thanks to my friends Oscar Manso, Carmel Ryan and all the postgraduate students of the school of computer applications for making my research life a great fun. Special thanks to Yvette Kabaadi for her continuos support and understanding during my busy times.

May God bless you all.

Chapter 1

RESOURCE LEVELLING PROBLEM (RLP) - PROBLEM DEFINITION

Introduction

Scheduling a project consists of determining a set of starting times for the activities of the project in such a way that the precedence constraints between them are satisfied and the total completion time is minimised. If there are no resource constraints, problems of realistic size can be solved by the critical path and network flow techniques. The activity start times developed from these techniques imply specified patterns or 'profiles' of resource usage over time. When resource availability levels are checked against the required levels of demand, the problem of resource allocation arises. It may be that demands exceeds availability levels in certain time periods or that the variation in resource profiles is considered excessive, and there is a reason to reduce excessive peaks and 'smooth' the profiles of usage.

The objective of the resource levelling process is to 'smooth' as much as possible the profiles of usage over time, within the given project duration. This is accomplished by judicious rescheduling of activities within their available slack (floating time) to give the most acceptable profile.

Clearly, there is an exponential number of combinations of starting times (feasible solutions) to be considered in the search for this optimal profile. This is a NP-Hard combinatorial optimisation problem and thus no exact algorithm is currently known for its solution.

We illustrate the problem by an application, which appears in the operations of manufacturing industries.

Resource levelling in manufacturing industries

In manufacturing industry, the scheduling process involves the allocation of the available capacity or resources (equipment, labour, and space) to jobs, activities, tasks, or customers through time [R93]. Important documents required in this process include the Bill of Material (BOM) and the Master Production Schedule (MPS).

Bill of Material

A structured list of all the materials or parts needed to produce a particular finished product, assembly, subassembly, manufactured part, or purchased part [J89].

An example of BOM is shown in Figure 1 for the assembly of a Motor bike.

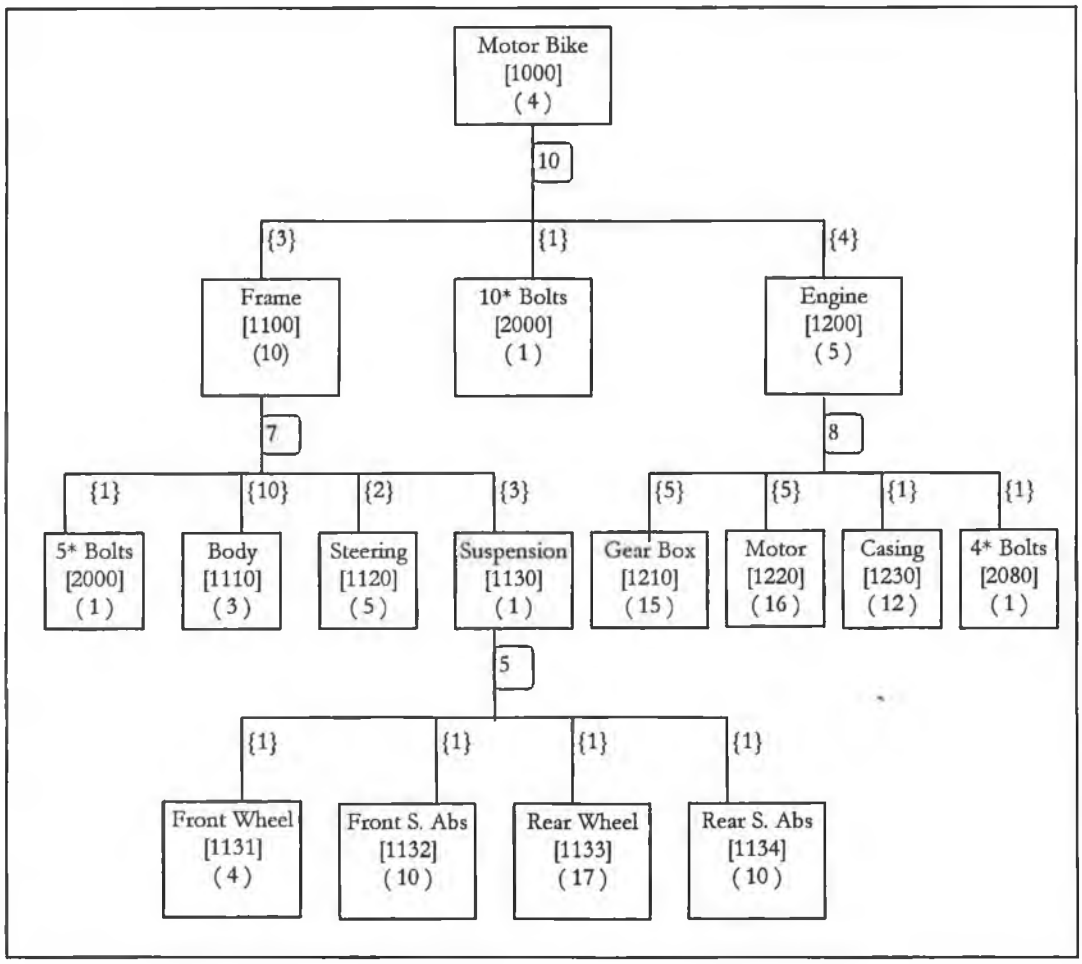


Figure 1: *An Example of a BOM.*

The BOM shows the relationships between various activities, which are used to assemble a specified product. For example, a frame cannot be assembled before a suspension. The notations used in the example are defined as follows;

- [ ], e.g. [1000] : Stock number, i.e. the task identifier,
- ( ), e.g. (10) : Simple lead time if all subassemblies are available, and purchase lead time for bought in components (duration).
- ☐ , e.g. 7 : Load (resource) level on production department.
- { }, e.g. {1} : Assembly time for an activity i.e. time required to fix an item into its parent.

We note that, this BOM can be converted to an activity network and solved by the Critical Path Methods (CPM) to obtain all the time-oriented parameters of each activity such as 'Earliest Starting Time' (EST), 'Latest Starting Time' (LST), 'Total Floating Time' (TFloat) and the critical path duration.

Table 1 shows a summary of these parameters as solved by the critical path methods for the BOM example of Figure 1 using TORA package [H92].

Activity	Durtn	Earliest		Latest		Total	Free
		Start	Complete	Start	Complete	Float	Float
Front Wheel	4	0	4	13	17	13	13
Front Absorber	10	0	10	7	17	7	7
Rear Wheel	17	0	17	0	17	0	0
Rear Absorber	10	0	10	7	17	7	7
5*Bolts	1	0	1	17	18	17	17
Body	3	0	3	15	18	15	15
Steering	5	0	5	13	18	13	13
Suspension	1	17	18	17	18	0	0
Gear Box	15	0	15	8	23	8	1
Motor	16	0	16	7	23	7	0
Casing	12	0	12	11	23	11	4
4*Bolts	1	0	1	22	23	22	15
Frame	10	18	28	18	28	0	0
10*Bolts	1	0	1	27	28	27	27
Engine	5	16	21	23	28	7	7
Motor Bike	4	28	32	28	32	0	0

Table 1: *Results from the Critical Path Method.*

Master Production Schedule (MPS)

A statement of requirements for 'end - items' by date and quantity. For a particular component, an end - item is the highest level item recognised in its bill of materials.

MASTER PRODUCTION SCHEDULE			
Job	**Subassembly name**	**Period required (days)**	**Number required**
1	**Motor bike**	45	5
2	**Engine**	40	8
3	**Motor bike**	50	6
4	**Frame**	42	4

Table 2: *An example of an MPS.*

One of the applications of both these documents (BOM and MPS) is to develop a resource profile by the work centre (or facility). This is used to obtain an idea of whether the given orders can be delivered within the given time scale (Planning horizon) without exceeding the available resources.

Using the above examples of BOM and MPS we apply a scheduling process called backward loading [R93] to develop a resource profile. In this process, the loading begins with the due dates for each activity and loads the processing time requirements (duration) against the facility by proceeding backward in time.

As an illustration, we use the latest starting times and the precedence of each task to construct a resource profile for job 3 (6 Motor Bikes) on the planning horizon. This looks as shown in Figure 2.

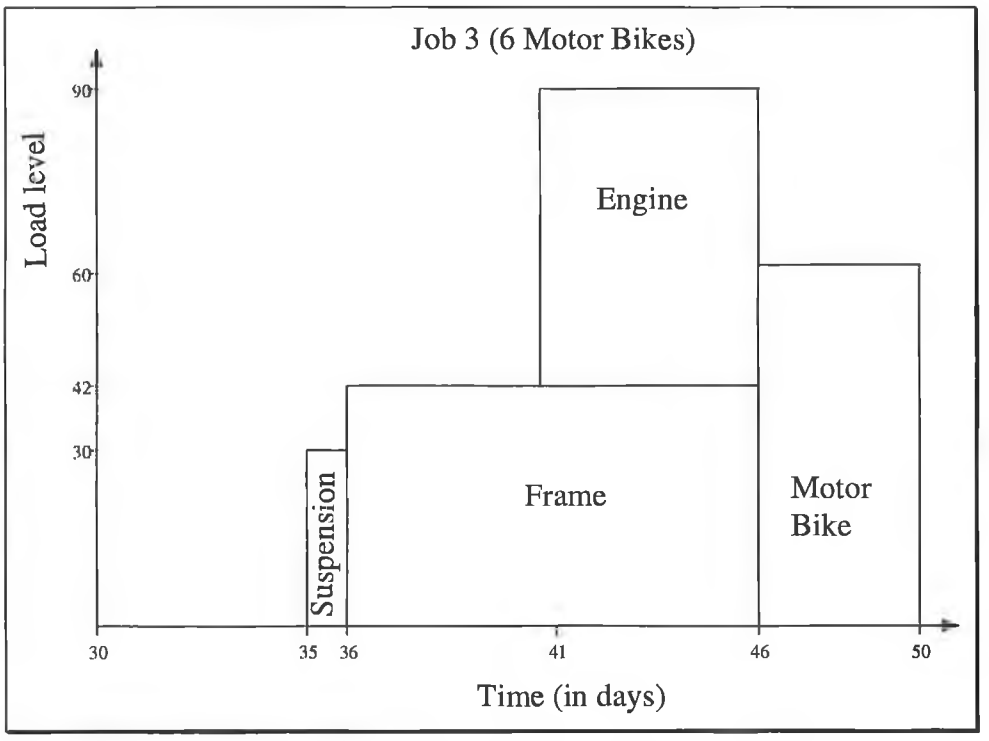


Figure 2: *Load profile for job 3 (6 Motor Bikes).*

The due date for job 3 is day 50 and the lead-time (duration) for a Motor bike is 4 days. From the CPM results in Table 1, the latest start and completion times of Motor Bike without considering the due date are 28 and 32 respectively. Since the due date is day 50, we have an extra time of 18 days (50 - 32) in which the activities can be scheduled. Thus the latest starting time for the assembly of a Motor bike is day 46 (28 + 18). Each assembly of the Motor bike is associated with the resource level of 10 units, and thus the resource level for the assembly of 6 Motor bikes is 60 units.

The lead-time for a frame is 10 days and it has to be completed before the assembly of the Motor bike. From CPM (table 1), the latest starting time for the assembly of frame is day 18. If we take into account the effect of the due date, the latest starting time for frame assembly is day 36 (18 + 18) with the resource level of 42 (6x7).

For the same reasons, the latest starting time for the assembly of the suspension is day 35 with the resource level of 30 (6x5), and that of the engine is day 41 with the resource level of 48 units (6x8).

The same is done to generate the resource profiles of the remaining jobs. Since we are dealing with one facility, the resource profiles of all jobs have to be put together. This leads to the total resource profile shown in Figure 3.

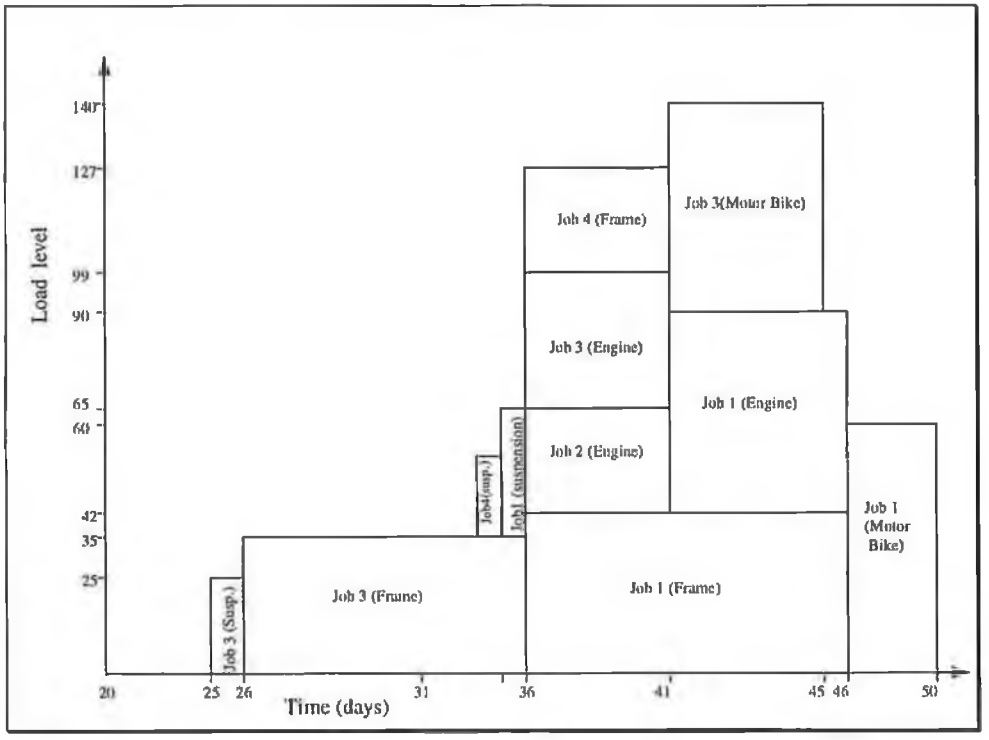


Figure 3: *Total load profile*

This resource profile is rather uneven. We have a very high resource level in day 41 (140 units, which may exceed the available resources), compared to only 25 units in day 25. No activity has been assigned between days 18 and 24 although it is also a part of the feasible region.

Since the times allocated for each activity are the latest starting times, it is possible to distribute activities within the planning horizon without violating the precedence relations in order to decrease the maximum production resource level. Note that, an activity can not be subdivided.

By visual inspection, the optimal schedule for the example above is shown in Figure 4 with the minimum resource level of 106 units.

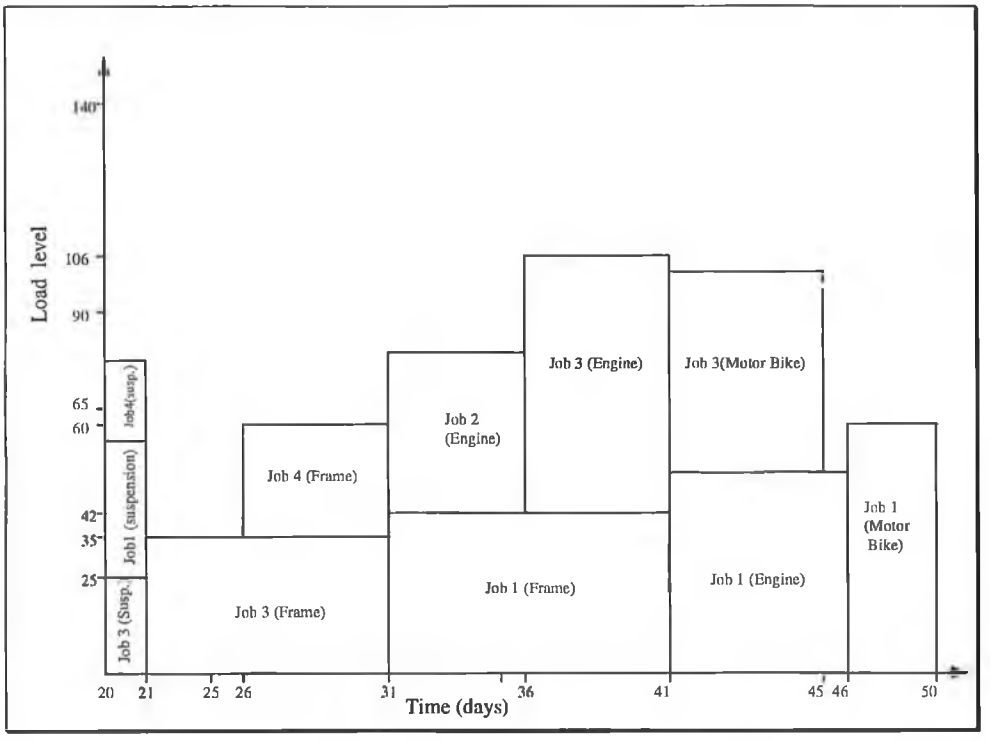


Figure 4: *Optimal load level*

It was possible to get an optimal solution to this example because of its small size. A real BOM usually consists of hundreds or even thousands of components (e.g. in the manufacture of an Aircraft). Since the solution space is exponential in size, this requires an enormous amount of searching.

Few studies have been done on this problem, all during 1960's and early 70's with suggested heuristics (see [BK62], [W64], [D66] and [D73]).

Chapter 2

A SURVEY ON PRE-PROCESSING AND THE GENERAL VALID INEQUALITIES OF THE MIXED INTEGER PROGRAMS

Introduction

The term Operations Research was coined to describe an area of applied mathematics in which algorithms were sought to solve practical industrial and military management problems. Initial successes in this domain included the development of Dynamic Programming (DP) and Linear Programming (LP). Subsequently, the domain of applications was enlarged to include mathematical programming problems involving only integral variables i.e. Integer Programming (IP) or a mixture of real and integral variables i.e. Mixed Integer Programming (MIP).

Integer programming problems are characterised by the property of having a large number of feasible solutions. As many of the algorithms used to try to solve these problems involved counting or searching procedures in a high dimensional space, the area of study became known as combinatorial optimisation. By the same token, the successful techniques developed within the framework of operations research were subsumed into accountancy and other management functional areas, thus de-emphasising its core theme.

Algorithms for integer programming problems can be classified into two categories: Heuristics, which seek (quick) approximate solutions and Exact Methods. Heuristic algorithms, which generate solutions, that can be proven to be within a tight bound of optimal solutions, have an important role to play in providing stopping rules for exact methods.

Recent advances in mathematical programming include the improved performance of Integer Programming (IP) using strong Linear Programming formulations. One focus is on the area of *reformulation* and *preprocessing* of the LP relaxation.
Preprocessing refers to elementary operations that can be performed automatically to tighten a given formulated model. These strategies can lead to a vast improvement in solution times of IP programs.

Thus, exact solution algorithms typically consist of three stages. In the first stage, attempts are made to reformulate and preprocess the problem, by reducing the size of constants and removing redundant variables and constraints. The ideas employed are independent of the algebraic structure of the formulation. In the second stage, the theory of cutting planes is used to generate valid inequalities to add to the set of constraints, thus reducing the size of the solution space. No global procedure is known for generating cutting planes and facets, so that valid inequalities have to be hand crafted to suit particular problem areas within the IP and MIP world. In the final stage of the solution process, the branch and bound process is used, as a final recourse, to try to find the optimal solution.

The Branch and Bound process is not a panacea for combinatorial optimisation problems. As a tree search procedure, it is critically dependent on the size of the search space it has to explore. The quality of work in the initial stages of the solution process largely determine whether the overall approach can be used to solve practical problems within the time frame allowed by industry.
In this chapter we discuss the idea of general valid inequalities and then concentrate on automatic procedures for reformulating and preprocessing MIP formulations. These procedures together with branch and bound are implemented in an augmented Simplex algorithm and computational results are reported.

Mathematical background

For the sake of completeness, the following definitions are included here.

- *Polyhedron* : Solution set of a finite system of linear inequalities and equations.
- *Polytope* : Bounded polyhedron.
- *Face* : Either an empty set or a subset F of a polyhedron P, obtained by replacing some of the inequalities defining P by equations. If $F \neq P$ then F is called a proper face. Each face is itself a polyhedron.
- *Facet* : Non empty proper face of a polyhedron P which is not contained in any other proper face of P (i.e. maximal).
- Let $X = \{x_1, x_2,...,x_r\}$ be a finite subset of $\Re^n$, $r > 0$. Then X is *linearly independent* if $\Sigma_i \lambda_i x_i = 0, \exists\, \lambda \in \Re^X \Rightarrow \lambda_i = 0, \forall\, i = 1,2,...,r$. Similarly X is said to be *affinely independent* if

  $\Sigma_i \lambda_i x_i = 0, \Sigma_i \lambda_i = 0 \Rightarrow \lambda_i = 0, \forall\, i = 1,2,...,r.$

Note that if a set of vectors X is affinely independent, then a new set of vectors X' obtained by the addition of a new component having value one to each row is linearly independent. For polyhedral combinatorics, affine independence is more important than linear independence because it is invariant under translations of the origin.

For example in $\Re^2$ the vectors $\binom{0}{1}$ and $\binom{1}{2}$ are both linearly and affinely independent, but the vectors $\binom{1}{1}$, are affinely independent but not linearly independent.

However the vectors $\binom{1}{1}$, have been obtained by the translation of $\binom{0}{1}$, $\binom{1}{2}$ by $\binom{1}{0}$.

- A vector $x \in \Re^n$ is said to be a *linear combination* of the vectors $x_1, x_2, ..., x_m \in \Re^n$ if

  $x = \Sigma_i \lambda_i x_i$, $\exists\, \lambda \in \Re^m$.

Additionally,

- If $\lambda_i \geq 0$, i = 1,2,...,m then we call x a *conic combination.*
- If λ is not restricted to nonnegative values but $\Sigma \lambda_i = 1$, then x is said to be an *affine combination.*
- If $\Sigma_i \lambda_i = 1$ and $\lambda_i \geq 0$, i = 1,2,...,m then x is said to be a *convex combination* .

For a non empty subset $X \in \Re^n$ we denote by lin(X) the linear hull of elements of X or space generated by X; lin(X) is the set of all vectors which are linear combinations of finitely many vectors of X. Conic (con(X)), affine (aff(X)), convex (conv(X)) hulls are defined in the same way.
If X = lin(X) (respectively X = aff(X)) then X is called a linear (respectively affine) space.

The linear and affine ranks of $X \in \Re^n$ are the cardinalities of the largest linearly and affinely independent subsets of X respectively, and are denoted by rank(X) and arank(X) respectively.

Lemma 1 [P89] If $\{ 0 \} \in$ aff(X), then arank(X) = rank(X) + 1.

If $\{ 0 \} \notin$ aff(X), then arank(X) = rank(X).

The dimension of a set $S \subseteq \Re^n$ is defined as dimS = arank(S) - 1. The empty set has dimension = -1.

Theorem 1 [R85] $P \in \Re^n$ is a polyhedron if and only if there exists finite sets

$V \in \Re^n$ and $E \in \Re^n$ such that $P = conv(V) + cone(E)$.

In particular, P is a polytope if it is the convex hull of a finite set of vectors. The unique minimal such set, is the set of its vertices. This means if we can find a unique minimal set of vectors P such that each vector in P is a convex combination of the rest of vectors in P (i.e. P = conv(P)), then that set defines the vertices of polytope P. For a polytope representing integral solutions, every vertex is called a *lattice point.*

Let $P \subseteq \Re^n$ be a polyhedron.

An inequality $c^T x \leq c_o$ is called a *valid inequality* with respect to P if

$c^T x \leq c_o$, $\forall x \in P$.

A constraint is said to be *redundant* if it is implied by other constraints in the system.

The system of constraints is said to be *nonredundant* if it has no redundant constraints.

An inequality $ax \leq \beta$ from $Ax \leq b$ is called an *implicit equality* (in $Ax \leq b$) if

$ax = \beta$ for all x satisfying $Ax \leq b$.

We use the following notation;

$A^{=} x \leq b^{=}$ is a system of implicit equalities in $Ax \leq b$

$A^{+} x \leq b^{+}$ is the set of all other inequalities in $Ax \leq b$.

A system of equalities $A^{=} x \leq b^{=}$ is minimal (i.e. nonredundant) if its rows are linearly independent.

Theorem 2 [S86] If no inequality $A^{+} x \leq b^{+}$ is redundant in $Ax \leq b$, then there exists a one-to-one correspondence between the facets of P and the inequalities in $A^{+} x \leq b^{+}$, given by $F = \{x \in P \mid a_i x = \beta_i\}$ for any facet F of P and any inequality $a_i x \leq b_i$ from $A^{+} x \leq b^{+}$.

In this case we say that $a_i x = \beta_i$ defines or determines the facets of P.

Therefore, if we can find the minimal equation system and all facets of the inequality system, the polytope described contains all the vertices of the problem involved and the Simplex algorithm can be applied with a guarantee of an optimal solution as one of the vertices of this polytope. If the problem involved has no equation system, then it clearly suffices to find all the facets of the inequality system.

The following theorem gives the characterisation of those valid inequalities, which determine facets.

Theorem 3: [BG82] Let $P \subseteq \Re^n$ be a Polyhedron and $b \subseteq \Re^n \setminus \{0\}$, $b_o \in \Re$ such that $F = \{x \in P \mid b^T x = b_o\}$ is a proper face of P. Then the following statements are equivalent:

(i.) F is a facet of P.

(ii.) dimF = dimP - 1.

(iii.) If aff(P) = $\{x \subseteq \Re^n \mid Ax = a\}$, where $Ax = a$ is minimal equation system, (i.e. the rows of A are linearly independent) and $F \subseteq \{x \in P \mid c^T x = c_o\}$ where $c^T x \leq c_o$ is a valid inequality for P, then $\exists\, \lambda \in \Re^{n\text{-dimP}}$ and $\mu \geq 0$ such that $c^T = \lambda^T A + \mu b^T$.

(iv.) If P is of full dimension and $F \subseteq \{x \in P \mid c^T x = c_o\}$ for some valid inequality $c^T x \leq c_o$, then $\exists\, \mu \geq 0$ such that $c^T = \mu b^T$.

We call an inequality $b^T x \leq b_o$ facet defining for a polyhedron P if $F=\{x \in P \mid b^T x = b_o\}$ is a facet of P.

General valid inequalities

Early efforts in the search for the solution were concentrated on the general procedures. One of these general approaches involved the idea of cutting planes. This entailed finding the set of general valid inequalities, which cut off infeasible points from the Polytope of the problem's LP relaxation. We present a brief survey of these inequalities and the ideas behind them.

Gomory cuts

Gomory (1958) developed the first finite algorithm for the general solution of the pure IP and later MIP problem, called the cutting plane method. The idea was to define the valid inequalities, which trim off the infeasible points from the LP relaxation of the IP or MIP problem. In spite of this theoretical success, the computational efficacy of the algorithm has been rather disappointing. However, ideas may be, and indeed have been borrowed from this method to enhance the effectiveness of other types of solution techniques [T75] (p160-161).

The IP case;

Suppose we want to optimise an objective function $f(x)$ subject to $Ax \leq b$, $x \geq 0$, where x must be a vector of integral values and A is an $m_x n$ matrix whose values are integral.

We consider the optimal solution of the LP relaxation with at least one nonintegral solution. Let x_i be the basic variable corresponding to the i^{th} row which assumes a nonintegral value β_i. The rest of the variables w_j, j = 1,2,...,n are nonbasic in that row.

Then we have;

$x_i + \sum_j \alpha_i^j w_j = \beta_i$; or

$x_i = \beta_i - \sum_j \alpha_i^j w_j$;

Where α_i^j , $j = 1,2,...,n$ are the coefficients corresponding to the nonbasic variables in row i.

We define $\beta_i = \lfloor \beta_i \rfloor + f_i$,

$\alpha_{ij} = \lfloor \alpha_{ij} \rfloor + f_{ij}$,

where $N = \lfloor a \rfloor$ is the largest integer such that $N \leq a$.

Thus we have

$f_i - \sum_j f_{ij} w_j = x_i - \lfloor \beta_i \rfloor + \sum_j \lfloor \alpha_i^j \rfloor w_j$.

Since x_i, w_i are supposed to be integral, then the right hand side must be integral which in turn implies that the left hand side must be integral.

Clearly

$0 \leq f_{ij} \leq 1$,

$0 < f_i < 1$ (since β_i is nonintegral by assumption) and $w_i, x_i \geq 0$.

Thus

$\sum_j f_{ij} w_j \geq 0$, which implies that

$f_i - \sum_j f_{ij} w_j \leq f_i$, or

$f_i - \sum_j f_{ij} w_j < 1$, (since $f_i < 1$).

But the left hand side must be integral and hence

$$f_i - \sum_j f_{ij} w_j \leq 0$$

This is called the fractional cut or *Gomory cut* for the general IP problem [T89]. Gomory proved that an integral solution could be obtained after a finite number of cut generations in a systematic fashion [S86](p351).

The MIP case;

Again we consider x_k to be an integral basic variable of the MIP problem, where the k[th] equation is given by

$x_k - \lfloor \beta_k \rfloor = f_k - \sum_j \alpha_k^j w_j$ (a.1)

In this case the variables w_j may not necessarily be restricted to integral values. It is noted that, for x_k to be integral, either $x_k \leq \lfloor \beta_k \rfloor$ or $x_k \geq \lfloor \beta_k \rfloor + 1$ must be satisfied.

From (a.1), these two conditions are equivalent to either

$\sum_j \alpha_k^j w_j \geq f_k$ or

$\sum_j \alpha_k^j w_j \leq f_k - 1$.

We define J_+ and J_- to be sets of subscripts of j such that $\alpha_k^j \geq 0$ and $\alpha_k^j < 0$ respectively.

Then either $\sum_{e J_+} \alpha_k^j \geq f_k$ (a.2)

Or $\frac{f_k}{f_k - 1} \sum_{j \in J_-} \alpha_k^j w_j \geq f_k$ (a.3)

The inequalities (a.2) and (a.3) can not be satisfied simultaneously and can thus be combined into one constraint given as;

$$\sum_{j \in J_+} \alpha_k^j w_j + \frac{f_k}{f_k - 1} \sum_{j \in J_-} \alpha_k^j w_j \geq f_k$$

This is the *mixed cut* inequality [T89].

Chvatal inequalities [C73]

IP Case;

Chvatal developed a procedure using the *elementary closure operation* and proved that all inequalities corresponding to a general IP problem can be obtained from that procedure. Consider a feasible region of an IP problem given by $F = \{ x \in Z_+^n \mid Ax \leq b \}$, where Z_+^n is the set of nonnegative integral n-vectors and (A, b) is an $m_x(n+1)$ rational matrix.

If we relax the integrality condition and note that the resulting polyhedron is bounded then this defines a polytope $P = \{ x \in \Re^n \mid Ax \leq b \}$. F is then a set of finite lattice points inside P and conv(P) is again a polytope. Now the problem of optimising an objective function cx over F is equivalent to its optimisation over

conv(P): $\max.\{ cx \mid x \in F \} = \max\{ cx \mid x \in \text{conv}(P) \}$.

F is a subset of conv(P) and the extremum points of conv(P) come from F. The problem defined as

$\text{Max.}\{cx \mid x \in \text{conv}(P) \}$ is an ordinary (continuous) linear programming problem which can be solved by the LP methods.

Closure inequalities

Let S be a set of linear inequalities $Ax \leq b$ that determine a polytope P , the closure of S is the smallest set of inequalities that contains S and is closed under the following two operations called the *elementary closure operations*;

(i) Taking the linear combinations of inequalities

(ii) Replacing an inequality $\sum a_j x_j \leq a_o$ where $a_1, a_2, .., a_n$ are integral by

$$\sum a_j x_j \leq a, \quad \text{where } a > \lfloor a_o \rfloor.$$

Chvatal proved the following theorem, which is the main idea of his procedure:

Theorem 4: [C73]

Let the inequalities

$\sum_i a_{ij} x_j \leq b_i$, i = 1,...,n (b.1)

(where a_{ij}, b_i are real numbers) determine a bounded polyhedron in $\Re^n$.

Let $c_o, c_1, .., c_n$ be integers such that

$\sum_i c_i x_i \leq c_o$ (b.2)

holds for any choice of integers $x_1, x_2, .., x_n$ satisfying (b.1).

Then (b.2) belongs to the closure of (b.1).

He then proved that if (b.1) defines a Polytope P, then the closure of (b.1) determines conv(P).

Thus if we can get all the closure inequalities of S we can solve the LP over the corresponding polytope (conv(P)) by LP methods with a guarantee of an integral optimal solution.

Example: 1 (Moser's cube problem)

This can be represented by the following LP problem;

Maximise $\sum_i x_i$

Subject to $x_i + x_j + x_k \leq 2$, i, j, k = 1,2,...27, i < j < k, (c.1)

$0 < x_i < 1$, i = 1,...,27, (c.2)

x_j = integer, i = 1,...,27. (c.3)

The relaxed formulation is got by dropping the constraint set (c.3). In this case, if we set

$x_i = \frac{2}{3}$ (i = 1,2,...,27) we satisfy all the inequalities of the relaxed formulation and obtain $\sum_{i=1}^{27} x_i = 18$, but violate integrality constraint.

We show below that the inequality however, belongs to the closure of (c.1) and (c.2). If this closure inequality is applied, then
$x_i = \frac{2}{3}$ (i = 1,2,...,27) is no longer feasible and the solution is constrained to integral values. The closure inequality $\sum_{i=1}^{n} x_i \leq 16$ is derived by the elementary closure operations as follows;

(operation (*i*) on some of the constraints of c.2).

$$\frac{5}{6}(x_1 + x_2 + x_3) \leq \frac{5}{3}, \quad \frac{5}{6}(x_1 + x_4 + x_7) \leq \frac{5}{3},$$

$$\frac{5}{6}(x_3 + x_6 + x_9) \leq \frac{5}{3}, \quad \frac{5}{6}(x_7 + x_8 + x_9) \leq \frac{5}{3},$$

$$\frac{1}{3}(x_1 + x_5 + x_9) \leq \frac{2}{3}, \quad \frac{1}{3}(x_3 + x_5 + x_7) \leq \frac{2}{3},$$

$$\frac{1}{6}(x_2 + x_5 + x_8) \leq \frac{1}{3}, \quad \frac{1}{6}(x_4 + x_5 + x_6) \leq \frac{1}{3}.$$

Adding these inequalities we get

$$2(x_1 + x_3 + x_7 + x_9) + (x_2 + x_4 + x_6 + x_8) + 2x_5 \leq 8\frac{2}{3}$$

but since we require x to be integral and the coefficients on the left hand side are integral, it follows that the right hand side must also be integral.

Rounding the right hand side (operation (*ii*)) we conclude that

$$2(x_1 + x_3 + x_7 + x_9) + (x_2 + x_4 + x_6 + x_8) + 2x_5 \leq 8 \qquad \text{(c.4)}$$

belongs to the closure of (c.1) and (c.2). Adding (c.4) to the relaxation and repeating the same elementary operations, it has been shown [C73] that $\sum_{i=1}^{n} x_i \leq 16$ belongs to the closure of (c.1) and (c.2).

In general, the closure inequalities can be generated from the following three step procedure recursively [N84]:

1. $\sum_i ua_i x_i \leq ub$ is valid for F for any $u \in \Re_+^m$
2. $\sum_i \lfloor ua_i \rfloor x_i \leq ub$ is valid for F as $x \in \Re_+^n$
3. $\sum_i \lfloor ua_i \rfloor x_i \leq \lfloor ub \rfloor$ is valid for F as $\lfloor ua_i \rfloor \in Z^1$ for all i and $x \in Z^n$.

The *rank* of an IP is the maximum number of elementary closure operations that are necessary to eliminate the integrality constraint. It was also proved that there is no upper bound on the rank [C73].

The closure inequalities are related to the Gomory's cutting plane inequalities. In fact Hu [C73](p 336) showed that given a system of inequalities S the Gomory cuts belong to the closure of S. Thus the closure inequalities are a formalisation of the Gomory's cutting plane method.

Chvatal's closure inequalities assume a bounded polyhedron on the real space. Theorem 5 by Schrivjer [S80] working on a rational space, showed that the Chvatal closure inequalities are valid for a general polyhedron.

Theorem 5 [S80]:

For any polyhedron P there exists a number t such that $P^{(t)} = P_I$. Where P_I denote the convex hull of the lattice points contained in P and P(t) stands for intersection of t half-spaces H such that $P \subset H$.

MIP case;

Chvatal inequalities for a general MIP problem were described by Nemhauser and Wolsey [N84]. Their procedure called Mixed Integer Rounding (MIR) is summarised as follows:

Suppose $T = \{ x \in Z_+^n, y \in \Re_+^p, Ax + Gy \leq b \}$ where (A, G, b) is an $m_x(n + p + 1)$ matrix with rational coefficients. Let $N = \{ 1,...,n \}$, $J = \{ 1,...,p \}$, then

$$\sum_{j \in N} (ua_j) x_j + \sum_{j \in J} ug_j y_j \leq ub \quad \text{for all } u \in \Re_+^m; \tag{d.1}$$

Given two inequalities,

$$\sum_{j \in N} \pi_j^1 x_j + \sum_{j \in J} \mu_j^1 y_j \leq \pi_0^1$$

$$\sum_{j \in N} \pi_j^2 x_j + \sum_{j \in J} \mu_j^2 y_j \leq \pi_0^2$$

They constructed the third valid inequality

$$\sum_{j \in N} \lfloor \pi_j^2 - \pi_j^1 x_j \rfloor + \frac{1}{1 - f_0} \left(\sum_{j \in N} \pi_j^1 x_j + \sum_{j \in J} \min(\mu_j^1, \mu_j^2) y_j - \pi_0^1 \right) \leq \lfloor \pi_0^2 - \pi_0^1 \rfloor \tag{d.2}$$

Where $\pi_0^2 - \pi_0^1 = \lfloor \pi_0^2 - \pi_0^1 \rfloor + f_0$ and $f_0 = b - \lfloor b \rfloor$.

Theorem 6 [N84]:

Given the two valid inequalities for T, then (d.2) is valid for T.

This can be used recursively to generate all valid inequalities for T from those available.

Example: 2

Let $T = \{ x \in Z_+^2 \mid x_i \leq 1, i = 1,2 \ , y \in \Re_+^2 \mid y_1 + y_2 \leq 7, y_i \leq 5x_i, i = 1,2 \}$

Applying case (d.1) of the MIR procedure (which is the same as operation (i) of the elementary closure operations) we have;

$$\frac{1}{3}(y_1 + y_2) \leq \frac{7}{3} \tag{e.1}$$

$$\frac{1}{3}(y_1 + y_2) \leq \frac{5}{3}(x_1 + x_2) \quad \text{or} \quad -\frac{5}{3}(x_1 + x_2) \leq 0. \tag{e.2}$$

We take (e.1) as the $i = 1$ inequality and (e.2) as the $i = 2$ inequality and apply case (d.2) to get ;

$$\lfloor -\frac{5}{3} \rfloor (x_1 + x_2) + \frac{1}{1-\frac{2}{3}} (\frac{1}{3}(y_1 + y_2) - \frac{7}{3}) \leq \lfloor -\frac{7}{3} \rfloor \text{ or}$$

$$-2(x_1 + x_2) + (y_1 + y_2) \leq 4. \qquad \text{(e.3)}$$

Repeating the MIR procedure for any two inequalities (including e.3) recursively, [N84| have shown that all valid inequalities for MIP problems can be generated.

Instead of this lengthy MIP procedure it can be shown that the same inequalities can be described in terms of functions called superadditive functions as, explained in the next section;

Superadditive valid inequalities

We would like to consider the description of the valid inequalities for the MIP problem of the form

$T = \{ x \in Z_+^n, y \in \Re_+^p \mid Ax + Gy \leq b\}$ in terms of functions , where (A,b) is an $m_x(n+1)$ rational matrix and G is an $m_x p$ rational matrix.

Definition 2 [N84]

A function $F : \Re^m \rightarrow \Re$ is called *superadditive* if $F(0) = 0$, and $F(u) + F(v) \leq F(u+v) \quad \forall \; u, v \in \Re^m$.

F is called nondecreasing if $u \leq v$ implies $F(u) \leq F(v)$.

It has been proved by Jeroslow and Johnson [N84](p11) that if F is superadditive and nondecreasing and $\overline{F}$ (defined by $\overline{F}(d) = \lim_{\lambda \to 0.} \frac{F(\lambda d)}{\lambda}$ exists and is finite for all $d \in \Re^m$), then

$$\sum_{i \in N} F(a_i)\, x_i + \sum_{i \in J} \overline{F}(g_i)\, y_i \leq F(b) \qquad \text{(f.1)}$$

is valid for T.

(f.1) is called a *superadditive valid inequality* determined by F.

Example: 3

Suppose T = $\{x \in Z^1_+, y \in \Re^1_+ : x + y \leq 4.5\}$

From [P89](p468), F(d) = $\lfloor d \rfloor$ is one of the families of the superadditive nondecreasing functions. Clearly $\overline{F}(d) = \lim_{\lambda \to 0} \lfloor \lambda d \rfloor / \lambda$ exist and is equal to zero.

From (f.1);

$F(1)x + \overline{F}(1)y \leq F\lfloor 4.5 \rfloor$

is a valid inequality for T. That is ;

$x + 0 \leq \lfloor 4.5 \rfloor$ or,

$x \leq 4$ is a valid inequality for T, which gives the cut shown on Figure 5.

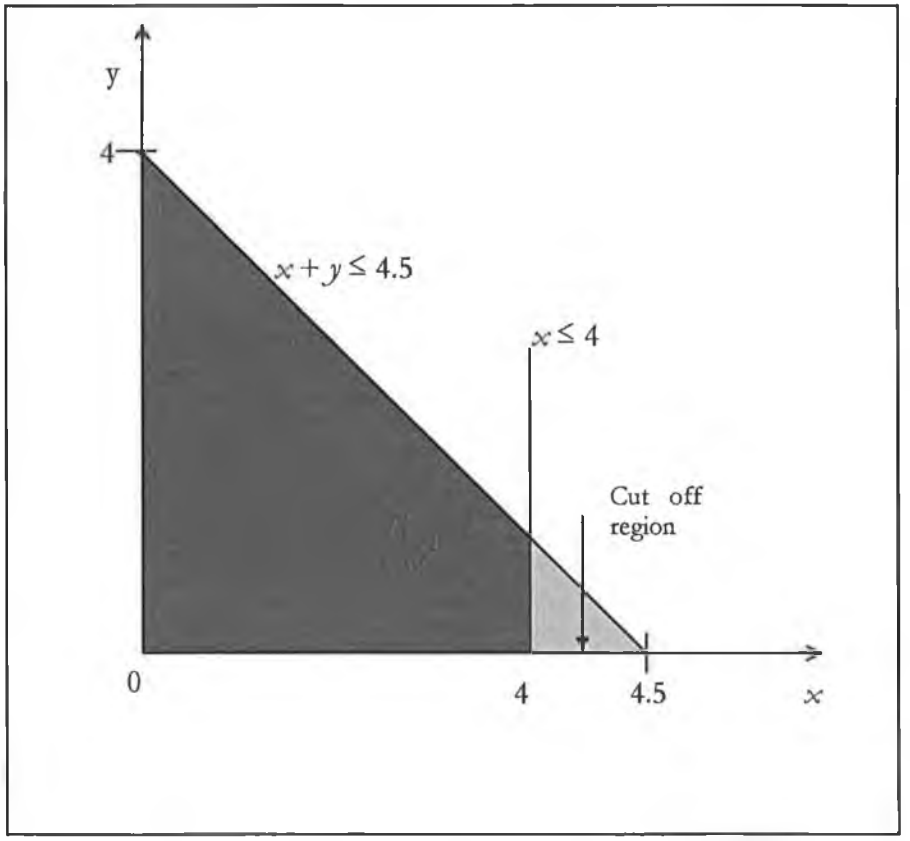


Figure 5: **Superadditive valid inequality**

Any inequality $\pi x + \mu y \leq \pi_o$ is said to be a superadditive valid inequality for T if it is equal to or dominated by a superadditive valid inequality for some F. Nemhauser and Wolsey [N84] proved that every valid inequality for $T \neq \emptyset$ is equal to or dominated by a superadditive valid inequality.

Theorem 7 [N84]**:**

Let $T = \{ x \in Z^n_+, y \in \Re^p_+ : Ax + Gy \leq b \} \neq \emptyset$.

If $[x \in \Re^n_+ : \{ y \in \Re^p_+ : Gy \leq b - Ax \} \neq \emptyset]$ is bounded, then every valid inequality $\pi x + \mu y \leq \pi_o$ for T is equal to or dominated by some superadditive valid inequality

$\sum_{j \in N} F(a_j) x_j + \sum_{j \in J} \overline{F}(g_j) y_j \leq F(b)$.

Proposition 1 [N84]

Let $F_\alpha : \Re^1 \rightarrow \Re^1$ for $0 \leq \alpha \leq 1$ be defined by

$F_\alpha(d) = \lfloor d \rfloor + \frac{(f_0 - \alpha)^+}{1 - \alpha}$

where $f_o = d - \lfloor d \rfloor$ and $(x)^+ = \min \{x, 0)$.

Then

a) F_α is superadditive and nondecreasing,

b) $\overline{F_\alpha}$ exists, and $\overline{F_\alpha}(d) = \min\{\frac{d}{1-d}, 0\}$.

Proposition 2;[N84]

Let $\tilde{F}_\alpha : \Re^1 \rightarrow \Re^1$ *for* $0 < \alpha < 1$ be defined by

$$\tilde{F}_\alpha(d_1, d_2) = \frac{1}{1-\alpha} d_1 + F_\alpha(d_2 - d_1).$$

Then $\tilde{F}_\alpha$ is nondecreasing and superadditive and,

$$\overline{\tilde{F}_\alpha}(d_1, d_2) = \frac{1}{1-\alpha} \min\{d_1, d_2\}.$$

This function can be used to generate valid inequalities recursively like the MIR procedure. That is, given two valid inequalities for T we can apply $\tilde{F}_\alpha$ in (f.1) above to generate another valid inequality.

Proposition 3; [N84]

If $H = \{ x \in Z^{n}_{+}, y \in \Re^{p}_{+} : \sum_{j \in N} \pi^{i}_{j} x_j + \sum_{j \in J} \mu^{i}_{j} y_j \leq \pi^{i}_{0},$

for $i = 1,2\}$,

then the superadditive valid inequality generated by $\tilde{F}_{\alpha}$ with $\alpha = f_{\alpha} = \pi^{2}_{0} - \pi^{1}_{0} - \lfloor \pi^{2}_{0} - \pi^{1}_{0} \rfloor$ is equal to or dominates the MIR inequality (d.2).

Example 4.

We consider the two constraints from example 2.

$\frac{1}{3}(y_1 + y_2) \leq \frac{7}{3}$

$-\frac{5}{3}(x_1 + x_2) + \frac{1}{3}(y_1 + y_2) \leq 0.$

$\alpha = \pi^{2}_{0} - \pi^{1}_{0} - \lfloor \pi^{2}_{0} - \pi^{1}_{0} \rfloor = -\frac{7}{3} + 3 = \frac{2}{3},$

Then the following is a valid inequality for T;

$\sum_{j \in \{1,2\}} \tilde{F}_{2/3}(\pi^{1}_{j}, \pi^{2}_{j}) x_j + \sum_{j \in \{1,2\}} \overline{F}_{2/3}(\mu^{1}_{j}, \mu^{2}_{j}) y_j \leq \tilde{F}_{2/3}(\pi^{1}_{0}, \pi^{2}_{0})$

i.e. $\tilde{F}_{2/3}(0, -\frac{5}{3}) x_1 + \tilde{F}_{2/3}(0, -\frac{5}{3}) x_2 + \overline{F}_{2/3}(\frac{1}{3}, -\frac{1}{3}) y_1 + \overline{F}_{2/3}(\frac{1}{3}, -\frac{1}{3}) y_2 \leq \tilde{F}_{2/3}(\frac{7}{3}, 0)$,

Or $\tilde{F}_{2/3}(0, -\frac{5}{3})(x_1 + x_2) + \overline{F}_{2/3}(\frac{1}{3}, -\frac{1}{3})(y_1 + y_2) \leq \tilde{F}_{2/3}(\frac{7}{3}, 0)$ (g.1)

Now $\tilde{F}_{2/3}(0, -\frac{5}{3}) = 0 + F_{2/3}(-\frac{5}{3}) = -2 + \min\{-1, 0\} = -2.$ (g.2)

And $\overline{F}_{2/3}(\frac{1}{3}, -\frac{1}{3}) = \frac{1}{1 - 2/3} \min(\frac{1}{3}, \frac{1}{3}) = 1,$ (g.3)

And $\tilde{F}_{2/3}(\frac{7}{3}, 0) = 7 + \tilde{F}_{2/3}(-\frac{7}{3}) = 7 - 3 = 4$ (g.4)

Applying (g.2), (g.3) and (g.4) in (g.1) we have;

$2(x_1 + x_2) + (y_1 + y_2) \leq 4$

Which is the inequality generated in example 4.

These functions can be turned into finite algorithms that produce successively tighter linear programming relaxation for solving MIP problems. Unfortunately, finite usually means extremely large for these general procedures, so the algorithms are not practical. However, this basic idea works well for some classes of IP or MIP problems where valid inequalities that define facets of conv(T) are known e.g. the transportation problem.

Preprocessing

There are many ways of representing an IP or MIP problem by linear inequalities and for which the underlying set of feasible integral solutions is essentially unchanged. The computational time of an IP or MIP largely depends on how close (tighter) the LP relaxation formulated is to the integral Polytope.

Preprocessing refers to the elementary operations that can be performed automatically to tighten a given formulation. Thus we present procedures that transform a 'user supplied' formulation automatically into a tighter equivalent representation. The techniques described, permanently fix variables, check and remove redundant constraints and reduce the size of certain coefficients within the coefficient matrix.

Variable fixing.

By fixing some variables at their bounds, the corresponding LP may be tightened, resulting in the optimal LP being closer to the optimal IP solution. We illustrate this by an example.

Example 5:

Minimise $5x_1 + 7x_2 + 10x_3 + 3x_4 + x_5$

Subject to $x_1 - 3x_2 + 5x_3 + x_4 - x_5 \geq 2$, (h.1)

$2x_1 + 6x_2 - 3x_3 - 2x_4 - 2x_5 \geq 0$, (h.2)

$-x_2 + 2x_3 - x_4 - x_5 \geq 1$, (h.3)

Where x_i are 0-1 variables.

By constraint (h.3), $2x_3 \geq 1 + x_2 + x_4 + x_5 \geq 1$, hence $x_3 \geq {}^1/_2$.

Since x_3 is an integer, this implies that the lower bound may be tightened by fixing x_3 at 1 and removing it from the problem.

By constraint (h.2), $6x_2 \geq 3 + 2x_1 + 2x_4 - 2x_5 > 1$, hence $x_2 \geq {}^1/_6$. By a similar argument, x_2 is fixed at 1 and removed from the problem.

By constraint (h.3), $x_4 \leq -x_5 \leq 0$, hence $x_4 \leq 0$. Therefore x_4 can be fixed at 0.
The reduced problem is now;
Minimise $5x_1 + x_5$
Subject to $x_1 - x_5 \geq 0$
$-2x_1 + 2x_5 \leq 3$
$x_5 \leq 2$,and x_i are 0-1 variables.
Again x_5 can be fixed at 0. The only remaining variable is x_1, which is obviously 0 in this case.
Tightening of the LP by fixing variables as above, reduces the problem feasible space to a greater extent and sometimes even solves the problem completely.
We can generalise Example 5 as follows;

Variable fixing algorithm; [C82]

Consider an arbitrary constraint, written for notational simplicity as

$$\sum_{j \in N_+} a_j x_j + \sum_{j \in N_-} a_j x_j \leq a_0 \qquad \text{(h.4)}$$

Where N_+ denotes the index set of coefficients a_j with positive value and N_- the index set of coefficients a_j with negative value. Now we let $j \in N_+$ and suppose that $a_j > a_0 - \sum_{k \in N_-} a_k$; then $x_j = 0$ in any feasible 0-1 solution, we can fix x_j at 0 and drop it from the problem. Likewise, if for some $j \in N_-$ we have $-a_j > a_0 - \sum_{k \in N_-} a_k$, then $x_j = 1$ in every feasible 0-1 solution, we can fix x_j at 1 and remove it from the problem.

Detecting infeasible and inactive constraints

Consider a general constraint, which has been formulated as (h.4).
Clearly if $\Sigma_{j\in N_-} a_j > a_o$, then (h.4) has no feasible solution and the overall problem is therefore infeasible.

On the other hand, if $\Sigma_{j\in N_+} a_j \leq a_o$, then every possible 0-1 vector satisfies constraint (h.4). In this case the constraint is inactive and can be dropped from the problem without excluding any feasible solution.
We also check for redundant constraints, where each row is compared with the rest, as follows;

Starting from the first column of each two rows, the coefficients of the same column are compared. If they are equal, we move to the next column, otherwise the algorithm moves to the next row. If all the left-hand coefficients of two compared rows are equal then the following are possible;

- The two inequalities are exactly the same and the first is removed
- One inequality dominates the other and the dominated row is removed
- The two inequalities form an equation and we leave them
- Otherwise, problem is infeasible.

We note that a row cannot be a multiple of another row because Euclidean reduction(see page 33) is performed before this process. Also we can not have equations, since all rows have previously been transformed into the form shown in (h.4)

Coefficient reduction.

This refers to the process of tightening the LP relaxation by reducing the size of the coefficients of the individual constraints of the problem where possible. Geometrically, this corresponds to a rotation of the constraints so as to increase the number of 0-1 solutions that satisfy them at equality.

To motivate the ideas, consider an example of a 2-dimensional integral constraint,

Example 6;

$4x_1 + 10x_2 \geq 7,$ (k.1)

Where x_1, x_2 are 0-1 variables.

It can be observed in Figure 6 that the constraint

$4x_1 + 7x_2 \geq 7,$ (k.2)

obtained by replacing the x_2 coefficient (10) by the right hand side constant (7) gives a tighter constraint. It is clearly better to use constraint (k.2) than (k.1) since the feasible region of the corresponding LP will be reduced by (k.2), without excluding any integral solution (as shown by the cut off region in Figure 6).

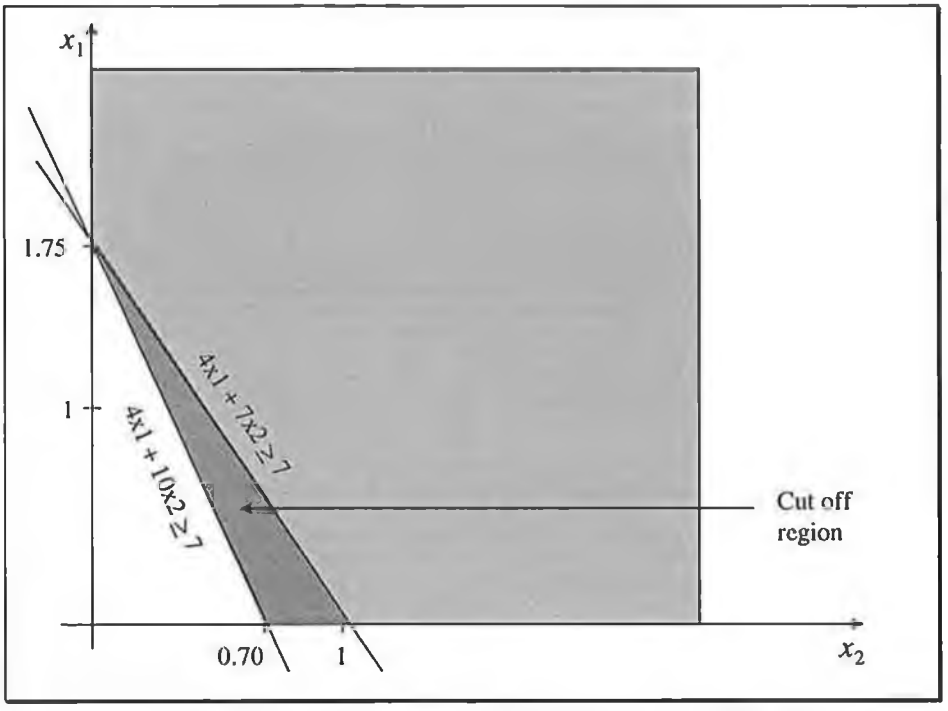


Figure 6: The effect of coefficient reduction on the feasible region.

We note that if we replace x_1 coefficient (4) by 7, the resulting region will be infeasible as shown in Figure 7.

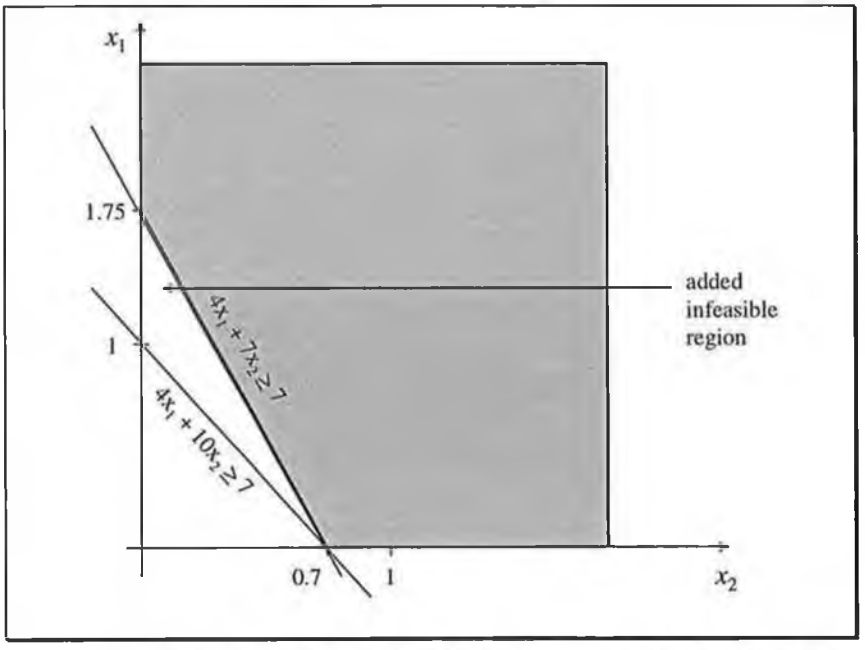


Figure 7: The effect of coefficient increase on the feasible region.

One of the ways of performing this process of coefficient reduction is described by the following algorithm;

Coefficient reduction algorithm [C82]

We consider an inequality in the form

$\sum_j a_j x_j \geq a_o$ (k.3)

Where all the a_j are positive and all variables are 0-1. This is called a Knapsack inequality.

Note that if any of the coefficients a_j are negative, we can use the substitution $x_j' = 1 - x_j$ to bring them to the form above.

If for some $k \in \{1,2,...,n\}$ we have $a_k > a_o$ we can replace a_k by a_o and the inequality

$a_o x_k + \sum_{j \neq k} a_j x_j \geq a_o$ (k.4)

has the same solution set as (k.3) but with more integral solutions in the feasible region.

Eucledian Reduction

This is another way of tightening the LP relaxation by reducing the size of coefficients. In this case we aim at reducing the size of the set of coefficients of a constraint such that the Greatest Common Divisor (GCD) of the resulting set is equal to one.

Eucledian reduction algorithm [H90]

We convert a constraint into the form $\Sigma_i a_i x_i \sim b$. Where $\sim$ is either $\leq$, $\geq$, or $=$ and all a_i are integral.

Both sides of the constraint are then divided by the GCD (of a_i's).

If an equation ('=' type) has a nonintegral right hand side after division, the problem is clearly infeasible.
Otherwise, if a constraint is of the '$\leq$' type ('$>$' type) we truncate the remainder (Truncate the remainder and add one).

This truncation does not eliminate any integral solution, but makes the set of nonintegral solutions to the relaxed problem smaller i.e. tightens the LP relaxation.

Example 7

Consider a constraint $4x_1 + 2x_2 \leq 9$

The GCD of {4,2} is 2.

Dividing the original constraint by the GCD we have;

$2x_1 + x_2 \leq 4.5$

Truncating the right hand side we have

$2x_1 + x_2 \leq 4$.

This is demonstrated by Figure 8 .

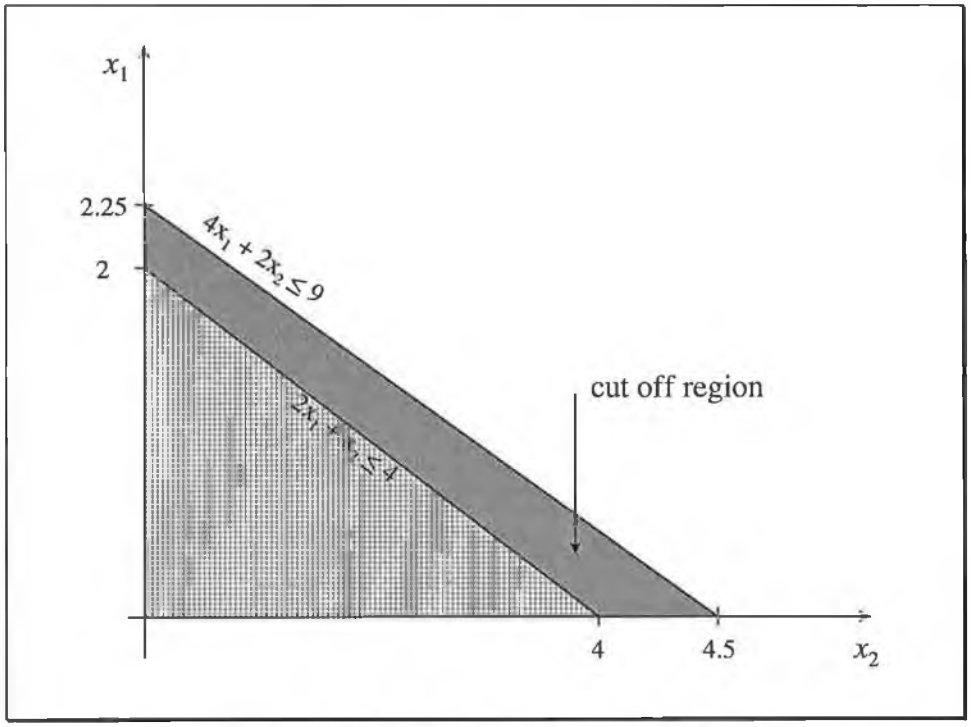


Figure 8: The result of the Eucledian reduction on the feasible region

The cut off region does not include any integral solution. It's elimination led to a polytope, which is defined by only feasible solutions. For a zero-one problem this polytope still needs to be reduced, but the cut off region may lead to a vast improvement in the solution time of an IP algorithm.

Implementation

We present our automatic preprocessing procedure as implemented using a C++ compiler and run on a 486, 50 MHz PC. The input is a formulated problem and the output is a preprocessed version of the problem and stored in MPS format. MPS format is a standard format commonly used by many LP solvers such as the one included in SCICONIC (a mathematical programming package). This package includes a simplex procedure called PRIMAL and Branch and Bound procedure called GLOBAL. We use the SCICONIC package on a VAX machine under the VMS operating system to solve our MPS format problem before and after preprocessing.

```
Procedure Preprocessing ( )
{  Input_Problem ( CoeffsMatrix, RowNumb, ColNumb, RowStatus );
   Eucledian_Reduction ( CoeffsMatrix, RowNumb, ColNumb, RowStatus, Feasible);
        if( NOT Feasible)  return ("Problem infeasible");
        // remove inactive and redundant rows and check infeasibilities //
        NewRowNumb = RowNumb;  Row = 0;
        while (Row < NewRowNumb AND Feasible)
        {   Increment ( Row);
              if( Inactive ( Row, CoeffsMatrix, RowNumb, ColNumb)
                 OR Redundant (Row)) AND  Feasible)
              Remove ( Row, CoeffsMatrix, NewRowNumb, ColNumb);
              } // endwhile //
        if( NOT Feasible) return ("Problem infeasible");
            RowNumb = NewRowNumb;     //new number of rows //
        // Variable fixing process //
        Found = true;
        while ( Found)
        {   Found = false;
            Find_Fixable_Var ( CoeffsMatrix, RowNumb, NewColNumb, ColNumb );
             if( Found)
             FixVariable( CoeffsMatrix, RowNumb, NewColNumb, VarPosition, Value)
        }   // endwhile //
        ColNumb = NewColNumb;  // new number of columns //
        // coefficient reduction //
        Reduce_Coefficients ( CoeffsMatrix, RowNumb, ColNumb, RowStatus);
        // Repeat the inactive and redundant rows and infeasibility checking process //
        // Store the preprocessed problem in MPS format //
        Write_MPS_File ( CoeffsMatrix, RowNumb, ColNumb, RowStatus );
} //end preprocess //
```

We start with the Euclidean reduction process where all the coefficients are transformed into integral values by the following procedure: for any row with rational coefficients a_j denote by EXPON the smallest number such that a_j*10^{EXPON} is an integer for all j. (Thus if all a_j are integral, EXPON = 0).

All coefficients of the row are multiplied by $a_i * 10^{EXPON}$ and thereby transformed to integral values. The resulting integral row is then divided by the GCD.

The next process removes all inactive and redundant rows and check for infeasibility. The procedure stops in case of any infeasibility.

The variable fixing process has two routines. The first routine finds a variable, which can be fixed from the tableau and returns the variable position and fixing value (0 or1). The second routine fixes the variable if found and adjust the tableau to reflect the new problem.

The coefficient reduction routine is then called, where all coefficients are firstly transformed into the form explained in the algorithm. After this process we again check for inactive and redundant rows as these might have been introduced by the previous routines.

The resulting MPS format file is then transferred to the SCICONIC package. Firstly the simplex procedure (PRIMAL) is applied to develop an initial optimal basis. Then we apply the Branch and Bound procedure (GLOBAL) to find an optimal integral solution. The number of iterations and CPU time is compared with those taken when the problem was solved without preprocessing.

Computational results.

Problem	Without preprocessing		With preprocessing	
	Total time (sec.)	Number of iterations	Total time (sec.)	Number of iterations
r3c5.dat	0.3300	6	0.2443	1
r10c6a.dat	0.4800	11	0.4543	10
r7c3.dat	0.2999	2	0.0043	0
r21c16.dat	0.4600	60	0.0054	0
r10c10.dat	0.4699	11	0.2064	6
r10c6b.dat	0.4800	16	0.4699	14
r20c25.dat	0.4699	8	0.0064	0

Table 3: Effect of Preprocessing on the computational time and the number of iterations.

Problem	Variables		Rows	
	Original	Fixed	Original	Fixed
r3c5.dat	5	4	3	2
r10c6a.dat	6	0	10	3
r7c3.dat	3	3	7	7
r21c16.dat	16	16	21	21
r10c10.dat	10	2	10	3
r10c6b.dat	6	0	10	3
r20c25.dat	25	25	20	20

Table 4: Effect of preprocessing on the number of variables and constraints.

	Original LP values	LP value after preprocessing	True IP value	Preprocessing time (sec.)
r3c5.dat	23.80	22.00	22	0.0043
r10c6a.dat	41.34	41.34	38	0.0043
r7c3.dat	1.30	0.00	0	0.0043
r21c16.dat	5.50	0.00	0	0.0054
r10c10.dat	44.42	41.00	41	0.0064
r10c6b.dat	41.57	41.57	38	0.0064
r20c25.dat	2.61	0.00	0	0.0064

Table 5: Effect of preprocessing on the LP value.

Our test problems were generated just for the sake of demonstration. The first part of the problem name stands for the number of rows and the second part represents the number of columns, e.g. the name r3c5.dat means 3 rows (r3) and 5 columns (c5). All problems are maximisation problems.

Table 3- Table 5 summarise the results.
The total times after preprocessing include the preprocessing and Branch and Bound times.

Table 3 reports the computational time and number of Branch and Bound iterations, before and after preprocessing. It is clear from Table 3 that there is a significant improvement in the number of iterations and computational time after preprocessing.

Table 4 shows the effect of preprocessing on the number of rows and variables.

Problems r7c3.dat, r21c16.dat and r20c25.dat were solved completely by the variable fixing procedure. For problems r3c5.dat and r10c10.dat, not all variables were fixed, but the preprocessed LP turned out to be integral. Thus the Branch and Bound procedure was no longer necessary. This is shown in Table 5 whenever the LP value of the preprocessed problem and the true IP value are equal.

The effect of the preprocessing on the relaxed formulation can also be measured by the difference in the objective values between the original and the preprocessed LP. We expect the preprocessed LP value to be closer to the true IP value sought. Table 5 reports the objective function values of the relaxed formulation and the objective function values of the problem after preprocessing.

For problems r10c6a.dat and r10c6b.dat, the LP value did not change. However, the preprocessing procedure removed 3 rows in each, in addition to the effect of the coefficient reduction routines. Thus the total computational time was improved as shown in Table 3.

The times shown are CPU times in seconds on an IBM 486 PC. In all cases, preprocessing took fractions of seconds. For example, the largest problems tested were r20c25.dat and r21c16.dat, where preprocessing took only 0.0064 and 0.0054 seconds respectively.

Summary and concluding Comments

- Our first aim was to describe the general valid inequalities to be applied in a cutting plane procedure as one of the approaches to the solution of a general IP or MIP problem. Although many inequalities were developed and proved to give a solution in a finite procedure, computational efficacy of their algorithms proved to be a problem. Branch and Bound methods are thus preferred.

  However, many IP or MIP problems are very large and Branch and Bound methods alone are not sufficient. Improvements to these types of problems before applying Branch and Bound are therefore essential and can lead to very successful results. One of these improvements is the reformulation and preprocessing of the original problems.
- Our second aim was to describe and implement an algorithm for preprocessing. We presented computational results, which demonstrates how successful this improvement can be on the large IP or MIP problem.

  All of our test problems were on the class of knapsack problems where the coefficients are not restricted to 0, 1 and -1 only. However, there are a large number of real life problems, which fall into this category. There are two types of these constraints, which are of special interest due to availability of special techniques to their solutions. These are special Ordered Sets or SOS of type 1 and type 2 [W78] (p173). These can be included in the preprocessing algorithm.

  For very large IP problems, preprocessing may not be sufficient to define a problem, which can be applied directly to a Branch and Bound procedure. Problem specific methods may be needed to generate facets of the underlying polytope (cut generation). However, preprocessing may vastly improve the computational time of these problems as demonstrated by our computational results.

Chapter 3

MIXED INTEGER PROGRAMMING FORMULATIONS FOR RLP

In this Chapter, we present four different Mixed Integer Programming formulations of the RLP. We conclude that the time indexed formulations give a better lower bound on the linear programming relaxation and hence a better starting point for the Integer Programming algorithms.

Assumptions

- No pre-emption i.e. the processing of an activity must be completed without interruption once started.
- Integral stating time i.e. an activity can start only on integral time units.
- The resource requirement of an activity is known and constant throughout the planning horizon.
- There is an unlimited amount of resources and our task is to determine the minimum amount required to complete the project.
- There is a pre-specified planning horizon within which all activities must be scheduled without violating the given precedence constraints.

Model 1 : Start-time variables and time-indexed resource determining variables.

Let x_i = Start time of activity *i*.

r_i = Resource level corresponding to activity *i*. (the amount of work units required to be performed on an activity per unit time)

p_i = Processing time of activity *i*.

$S_i = [x_i, x_i+p_i]$. A set of points covered by activity *i* on the planning horizon, given that it starts at x_i .

N = {1,2,...,T}. A set of all points on the planning horizon

h_t = Total resource level in the resource profile at time *t*. (Height of the resource profile at time t)

i.e. $h_t = \sum_{\forall i | t \in S_i} r_i$ i.e. for all activities *i* such that *t* lies in the interval $[x_i, x_i+p_i]$.

n = Total number of activities

T = The length of the planning horizon (Project duration) = |N|

H = A set of all arcs (i,j) in the network such that *i* precedes *j* .

Then the RLP is;

$$\text{Min.}\left(\max_{\forall t \in N} \{h_t\} \right)$$

$$\text{s.t. } x_j - x_i \geq p_i \qquad \forall (i,j) \in H$$

$$x \in Z_+^n$$

Or,

$$\begin{aligned} &\text{Min. } w \\ &\text{s.t } h_t \le w && \forall\, t \in N \\ &x_j - x_i \ge p_i && \forall\, (i,j) \in H \\ &x \in Z_+^n, w \in \Re_+, h \in Z_+^T \end{aligned} \qquad \text{(Ia)}$$

Let,

$$y_{it} = \begin{cases} 1 \text{ if } x_i \le t \le x_i + p_i - 1 \\ 0 \quad \text{Otherwise} \end{cases} \qquad \text{(IIa)}$$

Then $h_t = \sum_i r_i y_{it}$ $\qquad t = 1, 2, ..T.$ $\qquad$ (IIIa)

Condition (IIa) means that;

(1) $\sum_{t=x_i}^{x_i + p_i - 1} y_{it} = p_i \qquad i = 1, 2, \ldots n$

and

(2) $y_{it} = 0 \quad \forall t \notin [x_i, \; x_i + p_i - 1], \quad i = 1, 2, .., n.$

We need to define this condition without variable bounds.

We change the bounds in condition (1) and sum for all values of t in the planning horizon and then reinforce condition (2).

(3) $\sum_t y_{it} = p_i \qquad i = 1,2,..n$

(4) $y_{it} = 0 \qquad \forall\, t > x_i + p_i - 1\,, \qquad i = 1, 2, .., n$

(5) $y_{it} = 0 \qquad \forall\, t < x_i\,, \qquad i = 1, 2, .., n$

The following constraint reinforces (4);

$$ty_{it} \leq x_i + p_i - 1 \qquad i = 1, 2, .., n. \qquad \text{(IVa)}$$

Noting that the symbol T stands for the length of the planning horizon, the following condition reinforces (5);

$$(x_i - t) - T(1 - y_{it}) \leq 0$$

$$\text{or} \quad x_i + Ty_{it} \leq (t + T) \qquad i = 1, 2, .., n \qquad \text{(Va)}$$

This gives us the following MIP formulation:

$$
\begin{array}{l}
Min.\ w \\
s.t. \\
(i) \quad \sum_i r_i y_{it} \leq w \qquad t=1,2,...,T \\
(ii) \quad \sum_t y_{it} = p_i \qquad i=1,2,...,n \\
(iii) \quad ty_{it} \leq x_i + p_i - 1 \qquad i=1,2...,n;\ \forall t \in N \\
(iv) \quad x_i + T y_{it} \leq (t+T) \qquad i=1,2,...,n;\ \forall t \in N \\
(v) \quad x_j + x_i \geq p_i \qquad \forall (i,j) \in H \\
(vi) \quad x_i \leq T - p_i \qquad i=1,2,...,n \\
(vi) \quad y \in \{0,1\}^{n,T};\qquad x \in Z_+^n\,,\quad w \in \Re
\end{array}
\qquad \text{(VIa)}
$$

where Z_+ is the set of all nonnegative integral numbers. This formulation has n(T+1)+1 variables and T(2n+1)+2n+|H| constraints, where |H| is the cardinality of H.

Example

Example problems were generated randomly [AM96] and tested for all types of formulations. The problem (C3_02b) has 10 activities and a planning horizon of length 42.

We used the SCICONIC package [S91] (running on a VAX/VMS) to solve the test problems. The Matrix Generator Generator was used to generate the initial problem in MPS format using the two files - MGG problem and MG Data Input, as presented in the appendix A. These files were compiled, linked and run to produce an MPS matrix. The generated matrix was then solved by the SCICONIC programs called PRIMAL (for Linear Programming) and GLOBAL (for Integer Programming - Branch and Bound)[S93].

The MPS matrix generated for this model had 431 variables, 916 constraints with sparsity of 0.64%. The LP value (Lower Bound) which was obtained from the Primal procedure was 13.02.

The Integral solution was obtained after 2778 iterations as shown in the part of the report file presented in Figure 9 which shows only the values of x (start times),

```
          @@@@@@@@
PAGE    1

      PROBLEM @@@@@@@@ - SOLUTION NUMBER    2 - OPTIMAL

      CREATED ON 28-AUG-1996 11:44:52, AFTER     2778 ITERATIONS

      PRINTED ON 28-AUG-1996 11:45:50

      ...NAME...      .....ACTIVITY....  DEFINED AS

      FUNCTIONAL            19.000000   OBJ.....
      RESTRAINTS                        RHSSET01
      BOUNDS....                        BOUND01
1
          @@@@@@@@
PAGE    2

 .NUMBER. ..ROW... AT ...ACTIVITY... .SLACK ACTIVITY. ..LOWER BOUND.. ..UPPER BOUND... .DUAL ACTIVITY..
1
          @@@@@@@@
PAGE    3

  .NUMBER. .COLUMN. AT ...ACTIVITY...      ..INPUT COST.. ..LOWER BOUND.. ..UPPER BOUND... .REDUCED COST..
       917 W.....  BS       19.000000        1.000000          .            NONE
.
       918 X01.....  BS      1.000000          .               .            NONE
.
       919 X02.....  BS      6.000000          .               .            NONE
.
       920 X03.....  BS     12.000000          .               .            NONE
.
       921 X04.....  BS      1.000000          .               .            NONE
.
       922 X05.....  BS      2.000000          .               .            NONE      .
       923 X06.....  BS     19.000000          .               .            NONE      .
       924 X07.....  BS     19.000000          .               .            NONE      .
       925 X08.....  BS     17.000000          .               .            NONE      .
       926 X09.....  BS     32.000000          .               .            NONE      .
       927 X10.....  BS     42.000000          .               .            NONE      .
```

Figure 9: Part of the report file for model 1.

The value W = 19 is the optimal highest level in the resource profile and x01, . . x10 are starting times of activities 1, . . 10 respectively, on the planning horizon.

The corresponding resource profile is shown in Figure 10.

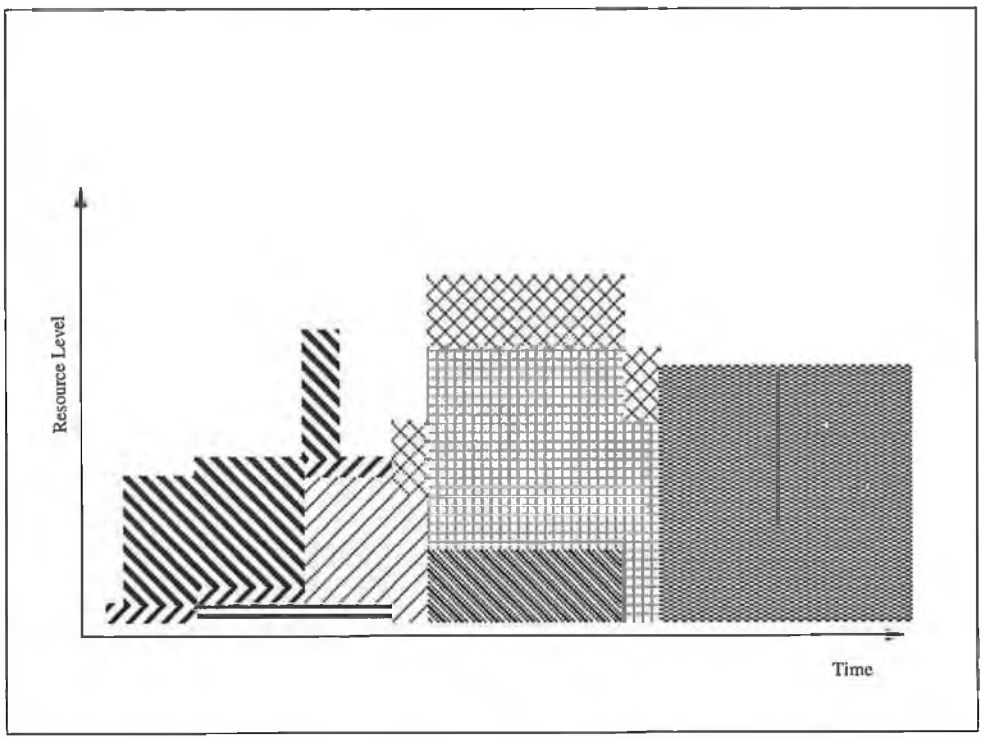


Figure 10: Resource Profile associated with the solution of model 1

Model 2 : Pure time-indexed formulation

Let x, r , p, h, n, S, N, T, H be as defined in the previous formulation,

and the RLP;

$$\text{Min. } w$$

$$\text{s.t. } h_t \leq w \qquad \forall\, t \in N$$

$$x_j - x_i \geq p_i \qquad \forall\, (i,j) \in H$$

$$x \in Z_+^n,\ h \in Z_+^T,\ w \in \Re_+ \qquad \text{(Ib)}$$

Now;

Let $y_{it} = \begin{cases} 1, & \textit{if activity } i \textit{ starts at time } t \\ 0, & \textit{otherwise.} \end{cases}$

Since an activity can start only once, then

$$\sum_t y_{it} = 1 \quad i = 1, \ldots n \tag{IIb}$$

The starting time of an activity i is then given by

$$x_i = \sum_t t\, y_{it} \qquad i = 1, 2, .., n$$

and the precedence relations changes to

$$\sum_t t(y_{jt} - y_{it}) \geq p_i \quad \forall (i, j) \in H \tag{IIIb}$$

We can reinforce the resource constraint by noting that

$\sum_{k=t-p_i+1}^{t} y_{ik} = 1$ only if t lies in the processing time interval of activity i

i.e. $t \in [x_i, x_i+p_i-1]$ and zero otherwise. [CA87]

$$\text{Thus } h_t = \sum_i r_i \Big(\sum_{k=t-p_i+1}^{t} y_{ik} \Big) \qquad \forall\, t \in N \tag{IVb}$$

This gives the formulation:

$$\text{min. } w$$

$$s.t.$$

$$\sum_t y_{it} = 1, \qquad i = 1, \ldots, n$$

$$\sum_t t(y_{jt} - y_{it}) \geq p_i, \qquad \forall (i, j) \in H$$

$$\sum_i r_i \Big(\sum_{k=t-p_i+1}^{t} y_{ik} \Big) \leq w, \quad t = 1, \ldots, T$$

$$y \in \{0,1\}^{nxT}, \quad w \in \Re_+$$

(Vb)

This formulation has $nT+1$ variables, which is less than the previous model ($n(T+1)+1$). It also have $n+|H|+T$ constraints which is far less than the previous model ($T(2n+1)+n+|H|$). The same example was used to test this model, where the corresponding MGG and MG Data Input files were used to generate the MPS format matrix (see Appendix A).

The matrix generated for this case had 421 variables, only 66 constraints and sparsity of 17.99%. The LP value (Lower Bound) obtained after PRIMAL agenda was 11.50.

Figure 11 shows parts of the Report file, which represent the optimal starting times.

1

@@@@@@@@ PAGE 1

PROBLEM @@@@@@@@ - SOLUTION NUMBER 4 - OPTIMAL

CREATED ON 29-AUG-1996 10:49:14, AFTER 4237 ITERATIONS

PRINTED ON 29-AUG-1996 10:56:53
...NAME... ....ACTIVITY.... DEFINED AS
FUNCTIONAL 19.000000 OBJ.....
RESTRAINTS RHSSET01
BOUNDS.... BOUND01

1

@@@@@@@@ PAGE 2

.NUMBER. .ROW.. AT ....ACTIVITY.... .SLACK ACTIVITY. ..LOWER BOUND.. ..UPPER BOUND... .DUAL ACTIVITY..

1

@@@@@@@@ PAGE 3

	NUMBER	.COLUMN.	AT	...ACTIVITY...	...INPUT COST...	.LOWER BOUND..	..UPPER BOUND...	..REDUCED COST..
	67	W..... .	BS	19.000000	1.000000	.	NONE	.
I	68	Y01.01..	LL	1.000000	.	1.000000	1.000000	.
I	69	Y02.01..	BS	1.000000	.	.	1.000000	.
I	70	Y03.01..	BS	1.000000	.	.	1.000000	.
I	71	Y04.01..	LL	1.000000	.	1.000000	1.000000	.
I	72	Y05.01..	BS	1.000000	.	.	1.000000	.
I	213	Y06.15..	LL	1.000000	.	1.000000	1.000000	4.000000
I	235	Y08.17..	LL	1.000000	.	1.000000	1.000000	4.000000
I	254	Y07.19..	LL	1.000000	.	1.000000	1.000000	.
I	386	Y09.32..	LL	1.000000	.	1.000000	1.000000	.
I	487	Y10.42..	BS	1.000000	.	.	1.000000	.

Figure 11: Part of the report file for model 2.

As defined in the formulation of the model, y_{ij} stand for activity i, which starts at point j on the planning horizon. For example Y06.15 means that, activity 6 should start at point 15 on the horizon.

Again the highest resource level is W = 19 and the resource profile is as shown in Figure 12.

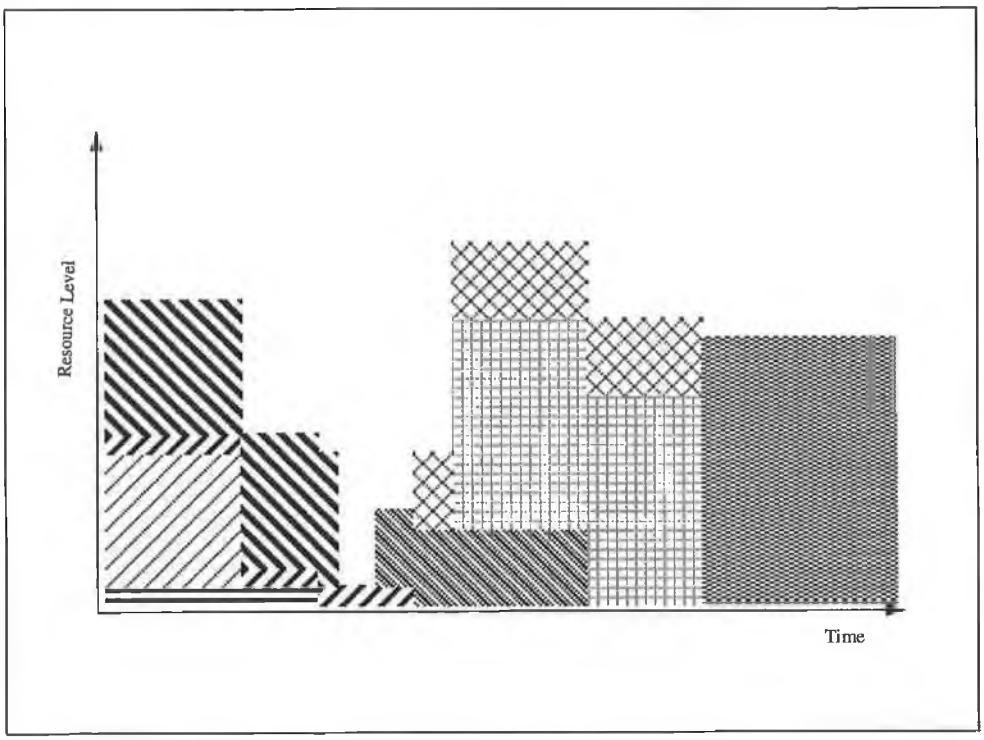


Figure 12: Resource profile representing the solution of model 2.

Model 3 : Reformulation of model 2.

The precedence constraints of model 2 can be replaced by a set of inequalities:

$$\sum_{k=1}^{t-p_i} y_{ik} - \sum_{k=1}^{t} y_{jk} \geq 0, \qquad \forall (i,j) \in H, \ \forall t$$

Although this set of inequalities contains more rows ($T^*|H|$) than the previous formulation ($|H|$), it is *totally unimodular*[W85].

Model 3 is then

$$\min w$$

$$s.t.$$

$$\sum_{t} y_{it} = 1, \qquad i = 1, \ldots, n$$

$$\sum_{k=1}^{t-p_i} y_{ik} - \sum_{k=1}^{t} y_{jk} \geq 0 \quad \forall (i,j) \in H, \ \forall t$$

$$\sum_{i} r_i \left(\sum_{k=t-p_i+1}^{t} y_{ik} \right) \leq w \quad t = 1, \ldots, T$$

$$y \in \{0,1\}^{nT}, \quad w \in \Re_+$$

This model has the same number of variables as the previous case i.e. $nT+1$, but more constraints $(T(|H|+1)+n)$. However, the number of constraints is still far less than the first model (which is $T(2n+1)+n+|H|$)

The model was tested on the same example. The size of the problem in terms of non-zeros was so large (23941) that the Matrix Generator (MG) could not generate the MPS format problem file. Fortunately we managed to generate the MPS matrix by another mathematical programming package called XPRESS. The MPS problem was then solved by SCICONIC and the solution is presented in Figure 13.

1

RLPS3 PAGE 1

PROBLEM RLPS3 - SOLUTION NUMBER 2 - OPTIMAL

CREATED ON 29-AUG-1996 18:12:55, AFTER 615 ITERATIONS

PRINTED ON 29-AUG-1996 18:15:06

...NAME... ....ACTIVITY.... DEFINED AS

FUNCTIONAL 19.000000 Obj
RESTRAINTS RHS00001
BOUNDS.... BOUND001

1

NUMBER	COLUMN		AT	ACTIVITY	INPUT COST	LOWER BOUND	UPPER BOUND	REDUCED COST
I	600	Y 0101	LL	1.000000	.	1.000000	1.000000	-1.000000
I	601	Y 0201	BS	1.000000	.	.	1.000000	.
I	603	Y 0401	LL	1.000000	.	1.000000	1.000000	.
I	644	Y 0505	BS	1.000000	.	.	1.000000	.
I	712	Y 0312	BS	1.000000	.	.	1.000000	.
I	767	Y 0817	BS	1.000000	.	1.000000	1.000000	.
I	786	Y 0719	LL	1.000000	.	1.000000	1.000000	.
I	805	Y 0621	BS	1.000000	.	.	1.000000	.
I	918	Y 0932	BS	1.000000	.	1.000000	1.000000	.
I	1019	Y 1042	BS	1.000000	.	1.000000	1.000000	.
	1020	Z	BS	19.000000	1.000000	.	NONE	.

Figure 13: Part of the Report File for model 3.

The LP lower bound was 14.5, which is higher than the previous tested models. The optimal solution was obtained after only 615 iterations. The solution profile is as presented in Figure 14.

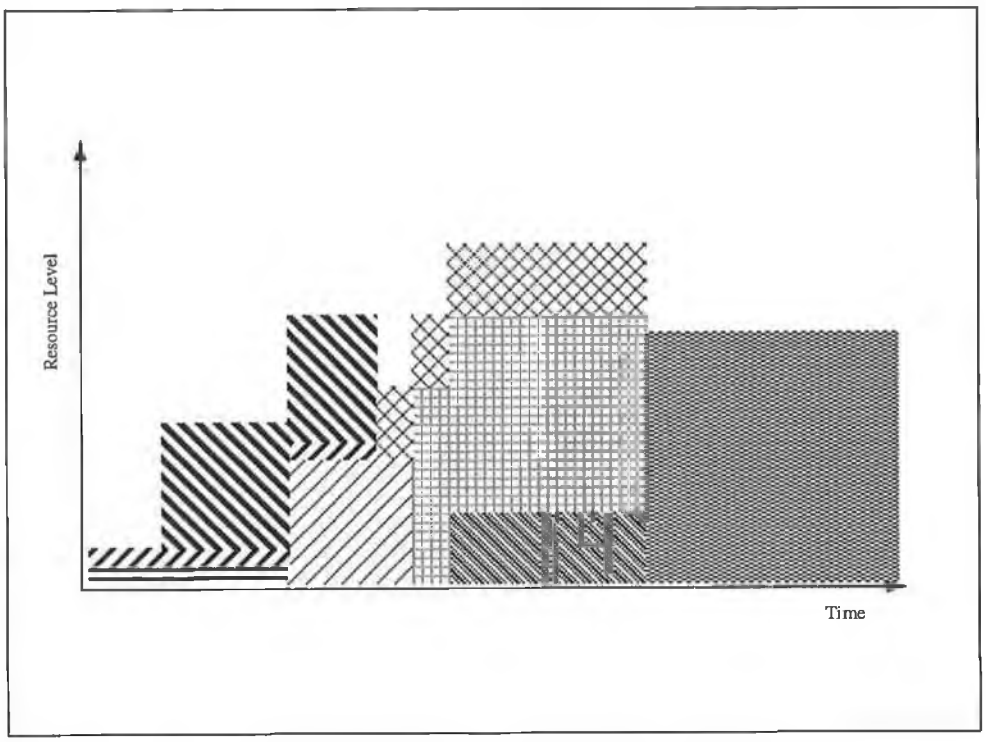


Figure 14 : Resource profile representing the solution of model 3.

Model 4: Start-time and sequence determining variables

Let p, w, H, T be as defined in the previous models.

Let x_i = Start time of activity *i*

We observe that the interval $[x_i, x_i + 1]$ for each *i* is sufficient to define all the *necessary* high points of the resource profile. Figure 15 illustrates this idea.

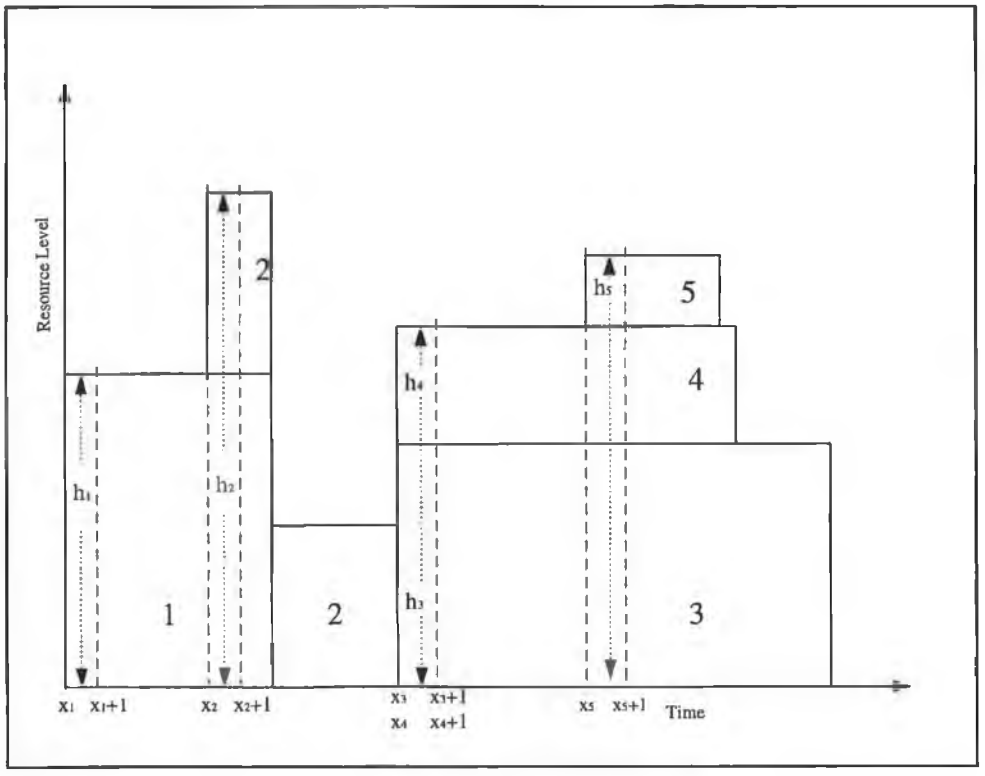


Figure 15: An example showing the necessary highest points for a given set of five activities.

The height h_2 is the highest peak that we intend to minimise and is a member of $\{ h_1, h_2, h_3, h_4, h_5 \}$.

If we sum the resource levels of all activities that falls into these intervals for each *i* (height at point *i*) we are guaranteed that the highest point will be one of these heights. The advantage of this model is that we only have to investigate *n* high points instead of all T points of the planning horizon.

Let $y_{ij} = \begin{cases} 1, & \text{if } x_j \le x_i \le x_j + p_j - 1 \\ 0, & \text{Otherwise.} \end{cases}$

Then for each i, the height is given by $\sum_j r_j y_{ij}$. We note that the number of binary variables in this case is n^2 instead of nT as in the previous models.

The MIP model 4 is therefore;

Min w

s.t. (i) $x_j - x_i \ge p_i$ $\quad \forall\ (i,j) \in H$

(ii) $\sum_j r_j y_{ij} \le w$ $\quad \forall\ i$

(iii) y_{ij} reinforcement constraints.

y_{ij} reinforcements.

We need to define the condition $x_j \le x_i \le x_j + p_j - 1 \Rightarrow y_{ij} = 1$

or $(x_i - x_j \ge 0) \wedge (x_j + p_j - 1 - x_i \ge 0) \Rightarrow y_{ij} = 1$.

We define two binary variables u and v such that;

$x_i - x_j \ge 0 \Rightarrow u_{ij} = 1$, and (1d)

$x_j + p_j - 1 - x_i \ge 0 \Rightarrow v_{ij} = 1$. (2d)

Then

$$(x_i - x_j \geq 0) \wedge (x_j + p_j - 1 - x_i \geq 0) \Rightarrow y_{ij} = 1 \equiv (u_{ij} + v_{ij} = 2) \Rightarrow y_{ij} = 1 \quad (3d)$$

The following constraint reinforces (1d)

$\boxed{(x_i - x_j) + \varepsilon - Tu_{ij} \leq 0,}$ where ε is a sufficiently small value ($\varepsilon = 1$ suffices our case) [W85].

i.e. $x_i - x_j - Tu_{ij} \leq -1$. (4d)

The following inequality reinforces (2d);

$(x_j + p_i - 1 - x_i) + \varepsilon - Tv_{ij} \leq 0$ for a sufficiently small value of ε ($\varepsilon = 1$) .

i.e. $\boxed{x_j - x_i - Tv_{ij} \leq -p_j \ .}$ (5d)

Similarly, (3d) is defined as

$(u_{ij} + v_{ij} - 2) + \varepsilon - y_{ij} \leq 0$

or $\boxed{u_{ij} + v_{ij} - y_{ij} \leq 1}$ (6d)

MIP model 4 is then,

Min w

s.t.

(i) $x_j - x_i \geq p_i$ $\quad \forall (i,j) \in H$

(ii) $\sum_j r_j y_{ij} \leq w$ $\quad \forall i$

(iii) $x_i - x_j - T u_{ij} \leq -1$ $\quad \forall i,j$

(iv) $x_j - x_i - T v_{ij} \leq -p_i$ $\quad \forall i,j$

(v) $u_{ij} + v_{ij} - y_{ij} \leq 1$ $\quad \forall i,j$

$y,u,v \in \{0,1\}^{n \times n}$, $w \in \Re_+$, $x \in Z_+^n$,

This model contains $3n^2 + n + 1$ variables and $3n^2 + n + |H|$ constraints. Thus the problem matrix for our example is made up of 311 columns and 324 constraints with 997 non-zeros (sparsity of 0.99). The LP lower bound was very poor (zero) and the optimal solution is as shown in Figure 16.

@@@@@@@@ PAGE 1

PROBLEM @@@@@@@@ - SOLUTION NUMBER 7 - OPTIMAL

CREATED ON 23-AUG-1996 15:54:59, AFTER 6654 ITERATIONS

PRINTED ON 23-AUG-1996 15:59:50

..NAME... ....ACTIVITY.... DEFINED AS

FUNCTIONAL 19.000000 OBJ.....

RESTRAINTS RHSSET01

BOUNDS... BOUND01

@@@@@@@@ PAGE 2

NUMBER	..ROW..	AT	....ACTIVITY....	.SLACK ACTIVITY.	..LOWER BOUND..	..UPPER BOUND..	..DUAL ACTIVITY..

@@@@@@@@ PAGE 3

NUMBER	.COLUMN.	AT	...ACTIVITY...	...INPUT COST...	...LOWER BOUND...	...UPPER BOUND...	..REDUCED COST..
325	W.....	BS	19.000000	1.000000	.	NONE	.
326	X01.....	BS	.	.	.	NONE	.
327	X02.....	BS	1.000000	.	.	NONE	.
328	X03.....	BS	2.000000	.	.	NONE	.
329	X04.....	LL	.	.	.	NONE	.
330	X05.....	BS	3.000000	.	.	NONE	.
331	X06.....	BS	20.000000	.	.	NONE	.
332	X07.....	BS	18.000000	.	.	NONE	.
333	X08.....	BS	16.000000	.	.	NONE	.
334	X09.....	BS	31.000000	.	.	NONE	.
335	X10.....	BS	41.000000	.	.	NONE	.

Figure 16 : The solution file for model 4.

The corresponding resource profile is shown on Figure 17.

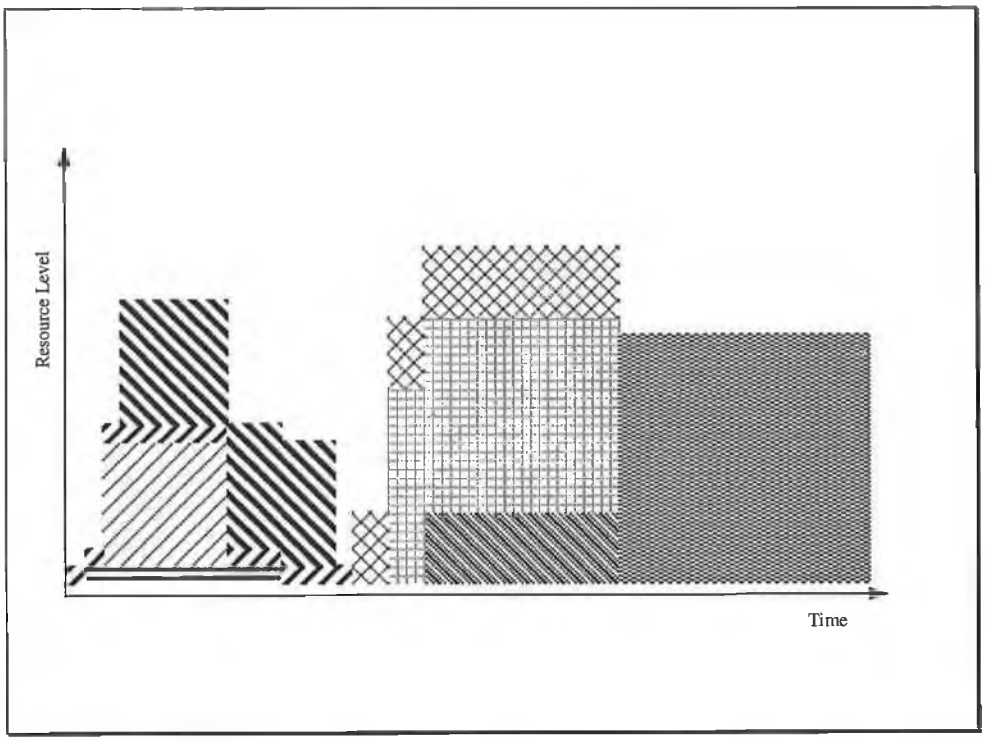


Figure 17 : Resource profile corresponding to model 4.

The size of all four developed models are:

	Model 1	**Model 2**	**Model 3**	**Model 4**
Rows	$2n(T+1)+T+\|H\|$	$n+T+\|H\|$	$n+T+T\|H\|$	$3n^2+n+\|H\|$
Columns	$n(T+1)+1$	$nT+1$	$nT+1$	$3n^2+n+1$

The experimental results for the tested cases are presented in the summary of results. We define the deviation of the LP lower bound (LP) from the optimal integral solution (IP) as ;

$$Dev = \frac{(IP-LP)}{IP} * 100$$

The smaller the value of Dev, the better is the lower bound. If Dev = 0, then IP = LP and in this case the LP relaxation defines an integral polytope which gives the IP optimal solution just by solving the LP problem. The worst case is when Dev = 100, because that means LP = 0 which is the least lower bound.

Summary of Results

Model	Rows	Cols	non-zeros	% Sparsity	LP Solutn	Iters (MIP)	% Dev.	Time (Sec.)
1	916	431	2515	0.64	13.02	2778	31.47	15855
2	66	421	5000	17.99	11.50	4237	39.47	611
3	599	421	23941	9.49	14.50	615	23.68	67
4	324	311	997	0.99	0.00	6654	100.0	3651

Table 6: Problem: c302b; n = 10; T = 42; Optimal Solution (IP) = 19

Model	Rows	Cols	non-zeros	% Sparsity	LP Solutn	Iters (MIP)	% Dev.	Time (Sec.)
1	827	391	2281	0.71	18.14	194667	24.42	24008
2	60	378	3814	16.82	14.64	2205	39.00	1665
3	536	378	15197	7.50	18.79	1266	21.71	44
4	538	521	1686	0.60	0.00	72059	100.0	24615

Table 7: Problem: c302a; n = 13; T = 29; Optimal solution (IP) = 24

Model	Rows	Cols	non-zeros	% Sparsity	LP Solutn	Iters (MIP)	% Dev.	Time (Sec.)
1	1117	529	3100	0.52	18.19	8266	41.32	3737.0
2	73	517	6445	17.08	15.03	3285	51.52	621.41
3	787	517	32724	8.04	18.68	2306	39.74	2935.97
4	462	445	1439	0.70	0.00	1413	100.0	36552.06

Table 8: Problem: c204b; n = 12; T = 43; Optimal Solution (IP) = 31

Model	Rows	Cols	non-zeros	% Sparsity	LP Solutn	Iters (MIP)	% Dev.	Time (Sec.)
1	1125	534	3130	0.52	20	671	39.39	80064.83
2	72	521	6229	16.61	15.67	3237	52.52	481.64
3	774	521	30081	7.46	21.90	1171	33.64	80.82
4	539	521	1688	0.60	0.00	8841	100	75297.80

Table 9: Problem: c304a; n = 13; T = 40; Optimal Solution (IP) = 33

Model	Rows	Cols	non-zeros	% Sparsity	LP Soln	Iters (MIP)	% Dev.	Time (Sec.)
1	1474	703	4127	0.40	14.81	3269	25.95	6115.00
2	83	690	7244	12.65	14.37	5510	28.14	2032.86
3	915	690	46535	7.37	19.5	1893	2.50	713.00
4	537	512	1684	0.61	0.00	13409	100	72779.71

Table 10: Problem: c402a; n = 13; T = 53; Optimal Solution (IP) = 20

In all cases the least deviation from the optimal solution (IP) is given by model 3. Due to its higher lower bound, model 3 took fewer iterations and less time to find an IP optimal solution than the rest of the cases (except problem c204b) . This is so despite of the fact that, model 3 involves a larger number of non-zeros, which results into a larger MPS problem.

Problem c204b is an exception because the LP lower bound was too far from the optimal IP value (39.74%). Although this lower bound (18.68) was better than other models, the optimal value was 31 which means that much iteration had to be performed before reaching optimality. Each iteration took more time due to the large size of the problem and hence more time was spent on the branch and bound procedure.

Conclusion

- Although model 3 has more non-zeros and therefore a larger MPS matrix than the rest of the models, the LP lower bound was better in both cases. The optimal solution was obtained after fewer iterations than the other models (except for problem c204b).

- The only model which did not involve the time-indexed variables was model 4, and the lower bounds obtained from this case were the worst. The *time-indexed* formulations are therefore better than *start-time variable* formulations.

Future work

- The best LP lower bounds are still not very close to the IP optimal value. Further research on other formulation techniques may lead to a better lower bound.

- Development of strong valid inequalities, which would trim off infeasibilities from the LP relaxation. This would increase the value of the lower bound or (hopefully) result in an integral polytope.

- Although all cases produced the same optimal solution, the corresponding resource profiles differ considerably. This indicates that there may be many optimal solutions for each problem. It may be possible to find another objective function, which would not only give us the minimum height but also, the most evenly distributed amongst those minimum height profiles.

Chapter 4

SIMULATED ANNEALING HEURISTIC TECHNIQUE

Simulated Annealing is an optimisation algorithm, which originates from the physical simulation of a gaseous state being slowly heated and cooled into a solid state.

Traditional forms of local search for minimisation problems employ a 'descent' strategy, whereby the search for a better solution always moves in the direction of improvement. However, employing such a strategy often results in convergence to local optima rather than to a global optimum.

Simulated annealing tries to overcome this drawback by using a random element in conjunction with the temperature parameter to generate a probability curve which determines when to accept a bad move in anticipation of a better convergence afterwards.

Simulated Annealing has been successfully applied to a variety of optimisation problems, spreading over many disciplines. These include combinatorial problems such as Travelling Salesman Problem, Circuit design, Data analysis problems, Imaging and Neural networks [I93].

The pseudo-code below (based around C++) represents the general Simulated-annealing algorithm: We assume that we have a minimisation problem with solution space S, objective function f and neighbouring structure N

```
Simulated_Annealing ( )
{   Select an initial solution s_o ;
    Select an initial temperature t_o > 0;
    Select a temperature reduction function α;
    While ( Not Freezing_Condition )
    {   Initialise( Iteration_Count );
        While( Iteration_Count <= Number_of_Repetitions )
        {   Randomly select s ∈ N(s_o);
            σ = f(s) - f(s_o);
            If( σ < 0 )
                s_o = s;
            else
            {   generate random x uniformly in the range (0, 1)
                if( x < exp( -σ/t ) s_o = s;
                }
        Increment( Iteration_Count );
        }
        Set t = α( t );
        Check_Freezing_Condition;
    }
    s_o is the approximate solution.
} // End Algorithm //
```

Figure 18: *Simulated Annealing Heuristic*

In adapting this general algorithm to our problem, a number of general and specific decisions need to be made. The general decisions are concerned with the parameters of the algorithm itself - the initial temperature and the cooling schedule. Problem specific decisions include the choice of the space of feasible solutions, the neighbourhood structure and the form of the cost function.

General decisions

Selection of the initial temperature (T_o)

In order for the final solution to be independent of the initial solution, T_o must be 'hot' enough to allow a free exchange of neighbouring solutions.

Definition

The acceptance ratio χ is the ratio between the number of moves accepted by the algorithm and the total number of moves generated by the algorithm for a given temperature.

Initially, the value of χ should be close to one. That is, T_o must be high enough such that virtually all generated moves are accepted.

Laarhoven & Aarts [HB91] suggested the following procedure for the determination of T_o.

The algorithm is run for a selected number of iterations (M). T_o is calculated from the following formula;

$$T_o = u^{>}\left[\ln\left(\frac{m^{>}}{m^{>}\chi-(1-\chi)(M-m^{>})}\right)\right]^{-1}$$

Where,

$m^{>}$ = the number of moves which showed an increase in cost ($\sigma > 0$)

$u^{>}$ = the average cost increase over the generated moves with $\sigma > 0$.

However, this procedure can be done reliably in those cases where the values of the cost function for different moves are sufficiently uniformly distributed. In our experience, we have found that the cost function values for RLP are not uniformly distributed.

A more natural procedure could be the following [C93];

> The value of χ is fixed to a high value and then the system is heated (increase temperature) until the probability of acceptance ($x = e^{-\sigma/T}$) reaches χ. At this point, T can be taken as the initial temperature for the tested problem. This is analogous to the physical cooling process where a substance is heated quite rapidly to its liquid state before being cooled slowly to its ground state (thermal equilibrium).

However, we found that this procedure was too time consuming as it entails the repetition of the stated process for each problem before cooling can commence.

We applied the formula below, which showed better results than the previously stated methods.

$$\frac{-\left(C_0/T_f\right)}{\ln(\chi_o)}$$

Where;

C_o = Initial cost value of the problem (Sum of the squares of the initial resource profile)

T_f = The final value of temperature i.e. freezing point (0.01 in our case).

The formula was derived as follows;

Since the probability of acceptance is given by the formula,

$x = e^{-\sigma/T}$, then,

$T = -\sigma/\ln(x)$;

If we assume that the probability of acceptance at the initial stage of the algorithm is χ_o we have,

$T_o = -\sigma/\ln(\chi_o)$;

We want to substitute σ with the maximum value of cost difference in the problem i.e. the difference between the maximum and minimum values of cost. Since this value is not known, we estimate it by multiplying the initial cost (C_o) by a factor greater than one, which we have chosen to be $1/T_f$

i.e. $\sigma = C_o/T_f$.

The cooling schedule

The rate at which temperature is reduced is vital to the success of the algorithm. This is dependent on the number of iterations at each temperature (*nrep*) and the temperature reduction function $\alpha(t)$.

Theory suggests that the system should be allowed to move very close to its stationary distribution at the current temperature before temperature reduction. However, the theory also suggests that this requires an exponential number of repetitions [C93], and therefore it is necessary to find a way of reducing the number of iterations. This may be achieved by applying a large number of iterations at few temperatures or a small number of iterations at many temperatures.

⇒ One of the most widely used schedules [C93] involves a geometric reduction function;

$$\alpha(t) = at, \text{ where } a < 1$$

From experience, the most successful values of *a* are between 0.8 and 0.99 with a bias to the higher end of the range. The algorithm stops when the change in temperature is less than a given value (0.01 in our case).

However this purely theoretical condition is often tempered with more pragmatic factors such as the amount of time spent by the algorithm, or the amount of time since the last most optimum solution was discovered. The reason for taking this approach is that, the algorithm may continue searching for a long time after the stable optima has been achieved. Counting the number of times which the solution is repeated without any change we can check if this number exceeds a selected value (m^3 in our case, where m is the length of the planning horizon) and stop the algorithm.

Number of repetitions at each temperature (nrep)

It is important to spend more time at lower temperatures so as to ensure that a local optimum has been fully explored. This can be done in various ways including the following two;

i) Starting with a small value of *nrep* at high temperature, we increase this value arithmetically by a small factor (0.05 in our case) at each new temperature.

ii) The feedback from the annealing process is used by accepting a fixed number of moves before decreasing the temperature. This implies that a short time will be spent at high temperatures when the rate of acceptance is high. Because the number of acceptances at very low temperatures will be low, it may take an infeasible amount of time to reach the required total, and it is therefore necessary to impose a maximum number of iterations per temperature as well (m^3).

⇒ Another commonly used temperature reduction function suggested by Lundy and Mees [LM86], executes just one iteration at each temperature, but reduces temperature very slowly according to the formula,

$$\alpha(t) = t / (1+\beta t)$$

where β is a suitably small value (in our case $\beta = 0.01$)

We collected data for both functions and compared the results.

Problem Specific decisions

Solution space

Our solution is a vector of starting times of each activity, which corresponds, to the near optimal even distribution of resource levels in the resource profile. The initial solution is a vector of the Earliest starting times of each activity. Since each activity can be scheduled anywhere within its 'total floating time' without affecting the completion time of the schedule, then the 'total solution space' is the set of all feasible combinations of these activity starting times over their total floats.

'Free floating time' (FreeF) of an activity is a part of its total floating time in which an activity can be scheduled without affecting the schedule of other activities in the activity network. The role of the *total* and *free* float in determining a feasible solution can be explained in terms of two general rules [H92]:

i) If the total float *equals* the free float, the noncritical activity can be scheduled anywhere between its earliest starting and latest completion times.

ii) If the free float is *less than* the total float, the starting of the noncritical activity can be delayed relative to its earliest start time by no more than the amount of its free float without affecting the schedule of its immediately succeeding activities.

Thus if an activity's starting time does not fall into a free float range, the immediately succeeding activities must be pushed forward by at least the amount of deviation of this activity from the free float region (i.e. ∇ as shown in Figure 19).

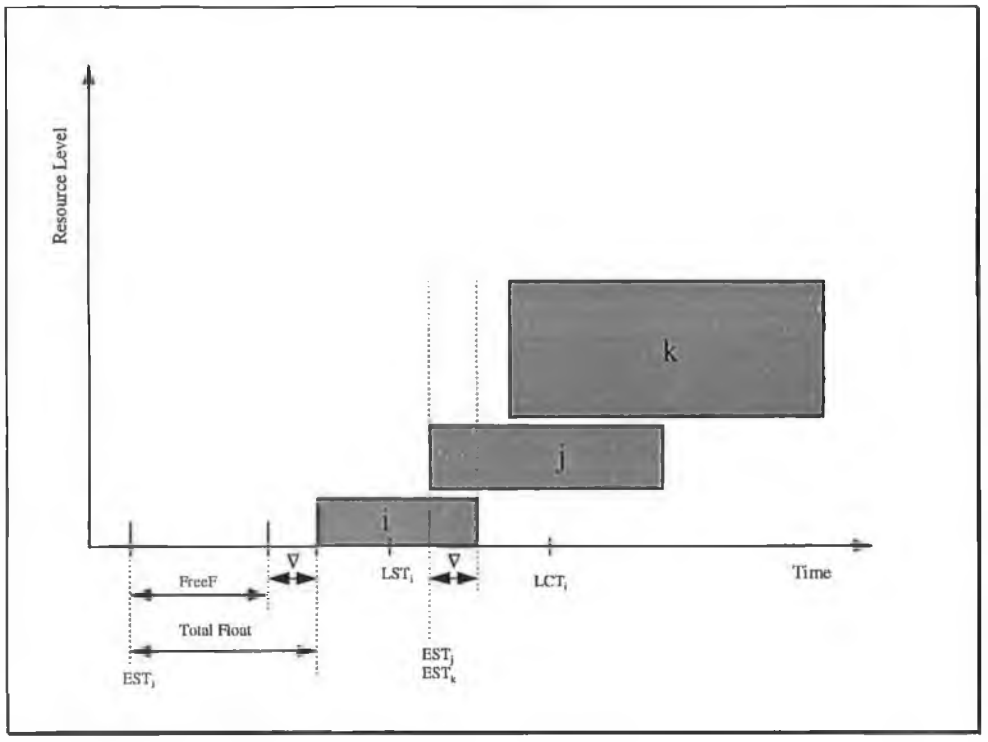


Figure 19: *The effect of activity* **i** *on its immediate successors* **j** *and* **k**.

In Figure 19, activities *j* and *k* are immediate successors of activity *i* i.e. *j* and *k* can start as earlier as *i* is completed. If *i* starts anywhere within the FreeF region, then it will always be completed before EST_j and EST_k which are the earliest starting times of activities *j* and *k* respectively. Figure 19 shows that activity *i* deviates from the FreeF by an amount ∇ and thus activity *j* must be pushed forward by at least the same amount ∇. It is noted that, we only push those activities that will otherwise violate the precedence relations.

For example, activity *k* does not need to be pushed forward because it starts at a point beyond the completion of activity *i*.

Neighbourhood structure

Three strategies were used:

I. **Fixed Neighbourhood.**

Since the aim is to minimise the highest resource level in the resource profile, it seems reasonable to concentrate our search on the neighbourhood of this highest point. At each iteration, we find the highest point and search for the next solution by selecting random starting times for the activities which lies in the neighbourhood of its corresponding time (*High*) by a predetermined distance *NbrHood*). That is those activities which lie in the interval [*High-NbrHood, High+NbrHood*] as shown in Fig. 20 provided that this range is feasible. This range is not feasible if ;

- (High - NbrHood) < 0 in which case 0 is elected as the lower bound of the range.
- (High + NbrHood) > *m*, in which case *m* is selected as the upper bound of the range.

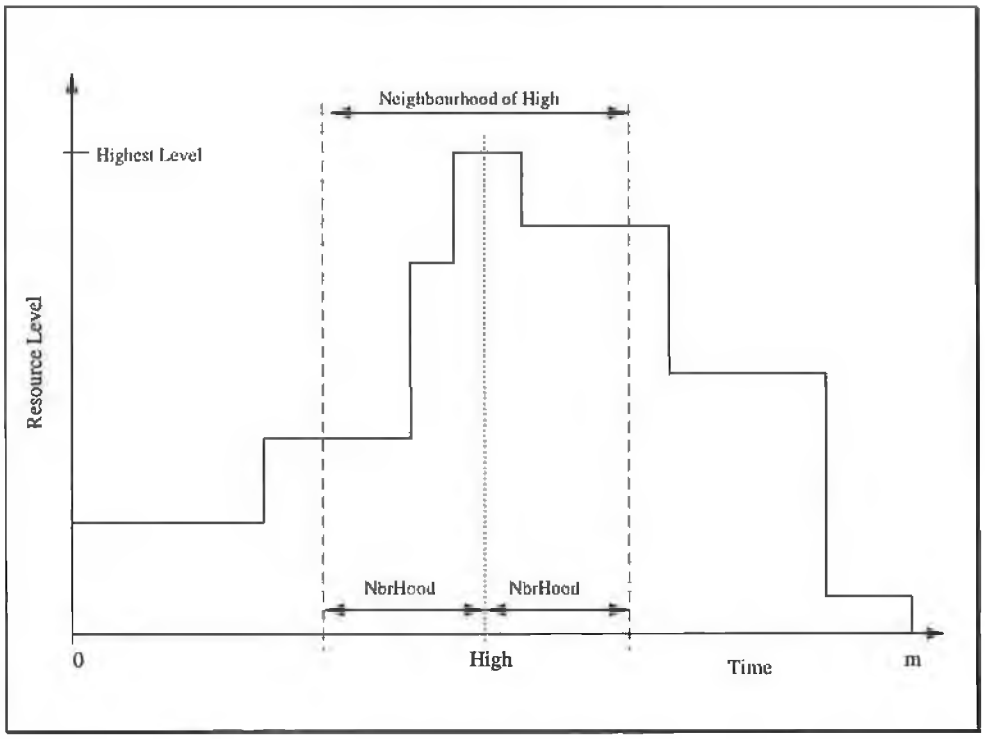


Figure 20: *Neighbourhood structure.*

This structure was found to work well only on the cases where there was a guarantee that one *High* point will not dominate the search. That is, if the algorithm could move freely through the planning horizon. However this is not always the case. If at the beginning, a certain selected neighbourhood fails to drop below the next highest point which is not in its neighbourhood, then the algorithm will be stuck on this neighbourhood (see Figure 21).

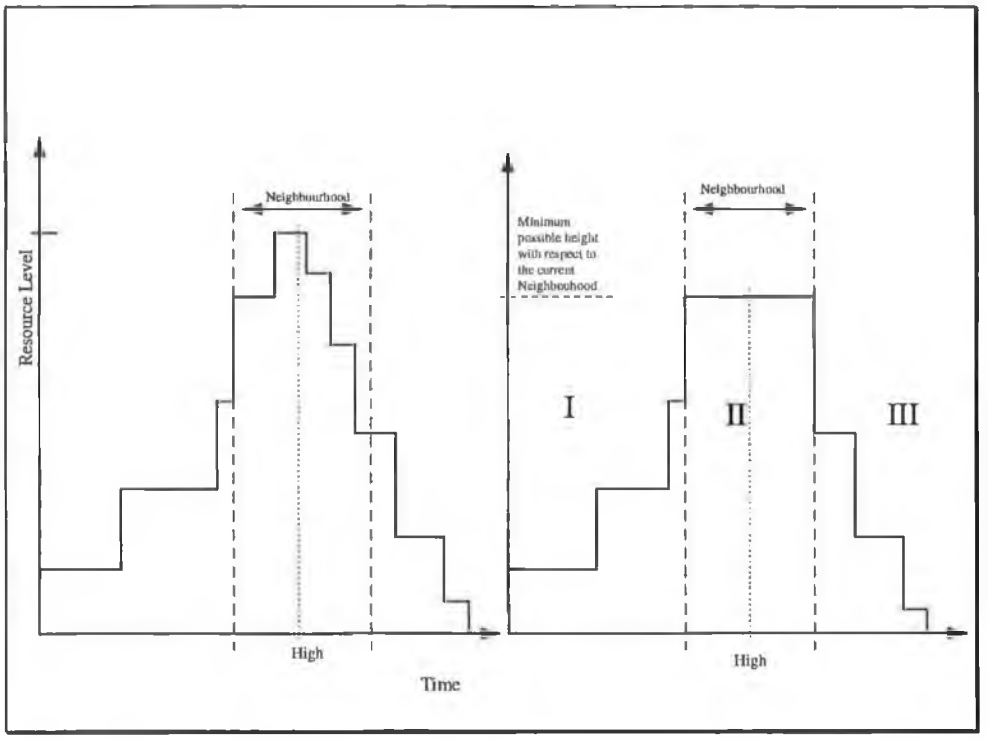


Figure 21*: Limitations of the fixed neighbourhood structure.*

Surely, the movements of activities in region II and I may result in more freedom of movements for activities in the current neighbourhood and lead to a better solution. To overcome this drawback, we implement a second strategy as explained below.

2. Increasing Neighbourhood.

We start with a small neighbourhood interval and slowly increase its value with temperature to a maximum value of *m*. Clearly, this is a more expensive strategy in terms of time, but it enables the algorithm to search in a wider space when the highest point is about to reach its stable position. That is, at each new temperature, we increase the searching space so as to increase the chances of an improvement in the solution at hand.

3. **Decreasing Neighbourhood.**

In [HB91](p 16), it is suggested that Large moves should be made at high temperatures so as to allow the algorithm to avoid local minimum, but as the temperature is lowered, smaller moves should be made to support the convergence to an equilibrium point. We implement this case by varying the neighbourhood interval from m to a minimum chosen value (2.0 in our case). This allows the algorithm to start by examining all activities within the planning horizon and then narrow the search as *High* moves to a more stable position. In this way, we minimise the possibility of being prematurely stuck on a fixed neighbourhood structure

The cost function

One of the ways of detecting the most even distribution is to minimise the sum of the squares of the activity levels in the resource profile [BK62]. We divide our resource level profile into unit intervals, and find the sum of squares of the resource levels over all units. The approximate solution corresponds to the minimum of these sums.

Simulated annealing for the Resource Levelling Problem then looks as shown in Figure 22;

```
Resource_Levelling_S_Annealing_1 ( )
{   Input_Problem( Activities, s_o );
    Current_Sum = Sum_Of_Squares( Activities , s_o );
    Calculate initial temperature t_o = g(Current_Sum);//g= initial temperature function
    Initialise (nrep_Criteria); //either nrep or number of acceptances //
    Initialise_Neighbourhood(N, s_o);
    While ( t > Freezing_Point ) // where Freezing point is a sufficiently small value
    {  Initialise( Iteration_Count );
       Change(nrep_Criteria);  //increase if not fixed
       Change_Neighbourhood(N, s_o); //increase or decrease if not fixed
       While( Iteration_Count <= nrep_Criteria )
       {   Randomly select s ∈ N( s_o );
           New_Sum = Sum_Of_Squares( Activities, s );
           σ = New_Sum - Current_Sum;
           If( σ < 0 )
           {    s_o = s;  Current_Sum = New_Sum;  }
           else
           {    generate random x uniformly in the range (0, 1)
                If( x < exp( -σ/t ) { s_o = s;  Current_Sum = New_Sum; }
                }
       Increment( Iteration_Count );
       }
    t = α(t);  // where α (t) = at 0.8 ≤ a ≤ 0.99 or α (t) = t/(1 +β t) //
    }
s_o is the approximate solution.
}   // End Algorithm //
```

Figure 22: *Simulated annealing heuristic for the Resource Levelling problem.*

Generation of the resource levelling problems

In order to test our algorithm, problems with varying features are needed and these are described in this section. The input to our heuristic is the output of the Critical Path Methods (CPM) which in turn uses the BOM as input. That is, the BOM tree is converted into an activity network and then solved by the CPM. We firstly describe how the BOM tree is generated, then a description of how the BOM is to be converted into an activity network is given.

Generating BOM.

We generate random BOM trees of varying distributions, which are governed by the following features;

- *Branching Factor (BF):* This factor represents the number of branches emanating from each new parent node. In order to avoid the uniformity of branches on each parent node, we represent BF in terms of a range of values in which the number of branches for a particular parent node can fall. Thus different ranges of BF correspond to different distributions of nodes in the BOM tree.

- *Number of Levels*: This determines the height of the BOM tree and its variation will result in different distributions of BOM nodes.

- *Duration and Resource Levels* of each node are also generated at random at specified upper and lower bounds.

Figure 23 shows a generated example of two different representations of a tree with the same number of nodes.

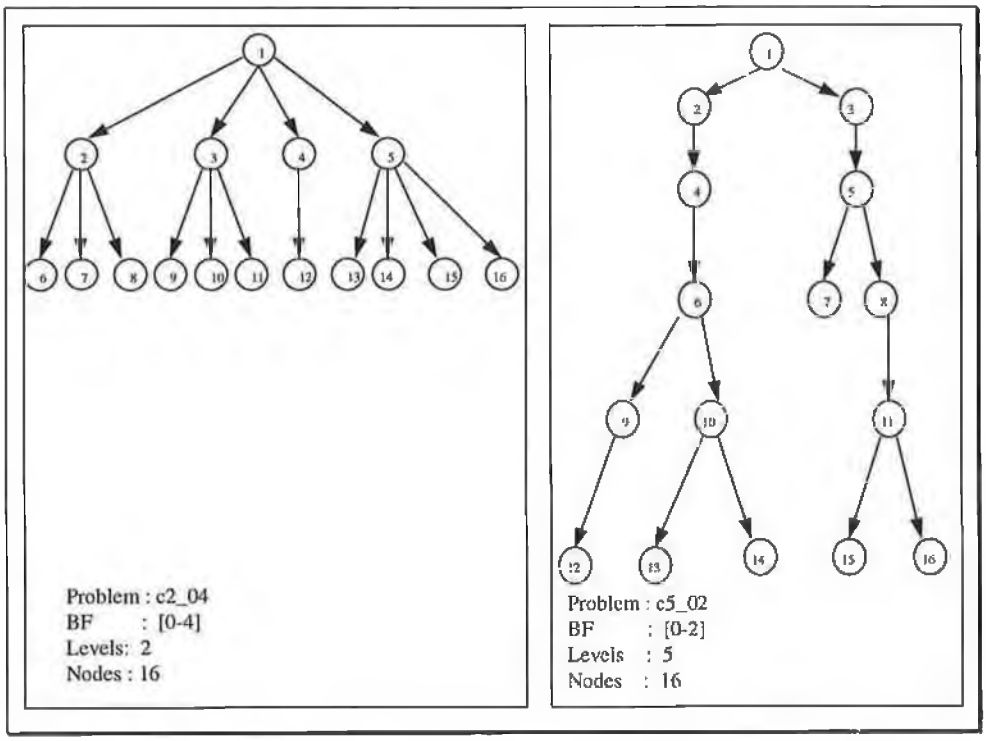


Figure 23: *Examples of trees generated from different BF and number of levels.*

The algorithm then looks as shown in Figure 24.

```
Generate_Random_BOM ( )
{
    Get_Number_Of_Levels ( Numb_Of_Levels );
    Get_Branching_Factor_Range ( MinBF, MaxBF ) ;
    Previous_Level_End = 0;
    arc_i = 0;
    Level  = 0;
    Current_Node   = 0;
    Next_Level_Nodes = 0;
    while( Level < Numb_Of_Levels )
    {
        Current_Level_Nodes = Next_Level_Nodes;
        Next_Level_Nodes = 0;
        while( arc_i <= (Previous_Level_End + Current_Level_Nodes) )
        {
            BFactor = Random( MinBF, MaxBF );
            for(arc_j=Current_Node+1; arc_j<=(Current_Node+BFactor); arc_j++ )
            {  Generate_Random ( Duration, Resource_Level );
               Store_Arc ( arc_i, arc_j , Duration, Resource_Level, BOM ); }
            Current_Node += Bfactor;
            Next_Level_Nodes += BFactor;
            arc_i++;
            }
        Level++;
     Previous_Level_End += Current_Level_Nodes;
     }
} // End Algorithm //
```

Figure 24: *Generating the BOM tree.*

Converting BOM to activity network

Our CPM algorithm applies the *Activity-on-Node* model of an activity network. From BOM, we add two dummy nodes, the starting (S) and the end nodes (T) respectively. Node S connects to all leaf nodes of BOM and assigned the duration of zero. Node T is assigned to the first node in BOM i.e. the final product of the assembly, also with zero duration. All arcs then point from S to T i.e. all arcs are reversed. We use an example of BOM in Fig. 1 for illustration as shown in Figure 25 below.

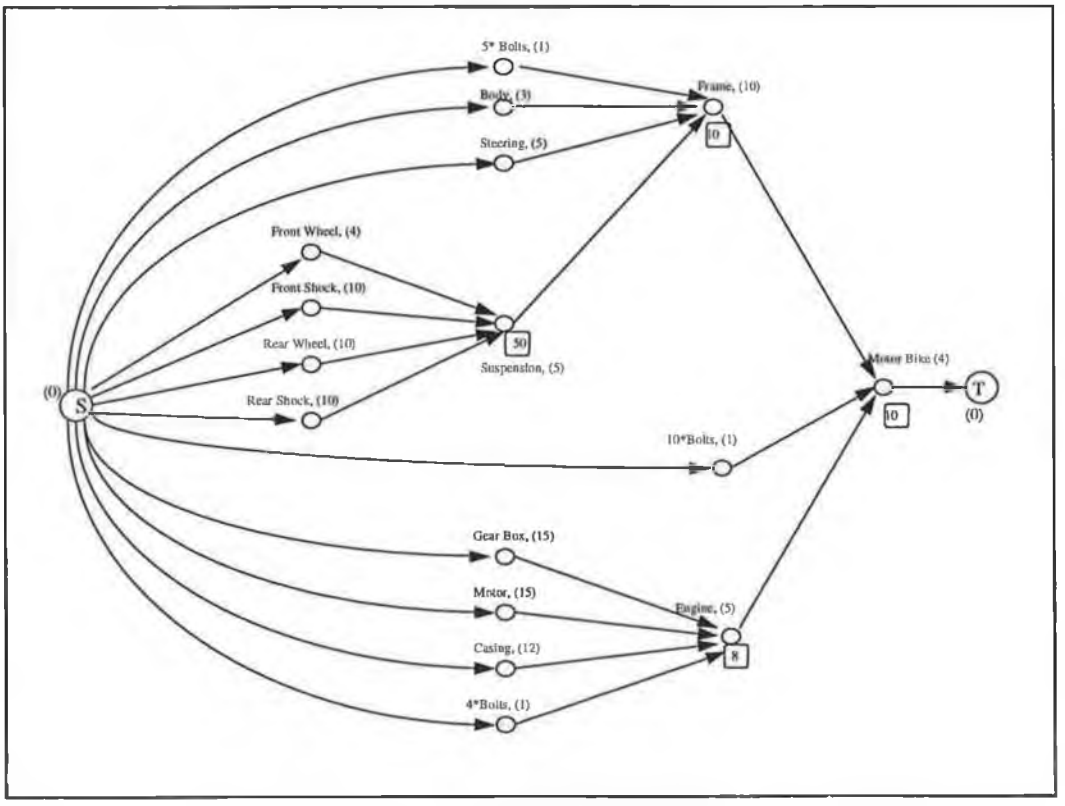


Figure 25: *Activity Network extracted from BOM in Fig 1.*

The rounded square boxes represent resource levels and the brackets represent activity duration. The activity network is then input to the CPM procedure where all time-oriented parameters as shown in table 1 are calculated.

Now, there are different orders of these assembly products from the Master Production Schedule, each with a different due date and quantity of the product required. Therefore, the BOMs for different orders have different load levels and time parameters. In this project we assume that all orders are of the same assembly product, i.e. no order requires the extraction of a subtree from the original BOM tree.

A data structure is needed to store the different trees for each order. These trees are used to create a single resource profile. We use an array of adjacency lists, where each array element represents the tree of a single order. To minimise the amount of memory required by the data structure, dynamic memory is implemented by using linked lists.

Since the MPS is generated at random, we make sure that all the due dates generated are not less than the critical path duration of the BOM tree. The BOM tree of each order is modified as follows;

- The load level of each activity in the BOM tree is multiplied by the number required for that order from MPS table.

- If the due date from MPS table for a given order is greater than the critical path duration, this due date becomes the new LCT for that tree. In this case, the difference between the two times has to be added to the latest starting (LST), free float and total floating times of each activity.

All trees generated i.e. one tree for each order, are used to build an initial resource profile.

The solution is stored in a 2-dimensional array, where each row stores the starting times of activities of a single order.

A more detailed version of the simulated annealing algorithm as was applied to the Resource Levelling Problem is now presented (Figure 26).

```
Resource_Levelling_S_Annealing_2 ( )
{
   Input_BOM_Tree( BOM );
   Create_MPS_Table( MPS, Number_Of_Orders );

   // for each order, modify the corresponding activity tree and Initialise the resource profile.
   for (order = 1; order <= Number_Of_Orders; order++ )
   {  Copy_BOM( BOM, Activity[order] ); // copy BOM to activity tree of that order
      Modify_Tree( MPS, Activity[order], s_o[order] );
      Initialize_Resource_Profile( Res_Profile, Activity[order] );
      }

Current_Sum = Sum_Of_Squares( Res_Profile, s_o );
Initialise(nrep_Criteria); //either nrep or number of acceptances //
Initialise_Neighbourhood(N, s_o);

Calculate initial temperature t_o = g(Current_Sum); //g = initial temperature function
//
While ( t > Freezing_Point )   // where Freezing point is a sufficiently small value //
{
   Initialise( Iteration_Count );
   Change(nrep_Criteria);  //if not fixed //
   Change_Neighbourhood(N, s_o); //if not fixed //
   While( Iteration_Count <= nrep_Criteria )
   {
      Randomly select s ∈ N( s_o );
      New_Sum = Sum_Of_Squares( Res_Profile, s );
      σ = New_Sum - Current_Sum;
      If( σ < 0 )
      {    s_o = s;  Current_Sum = New_Sum;  }
      else
      {    generate random x uniformly in the range (0, 1)
           if( x < exp( -σ/t ) { s_o = s;  Current_Sum = New_Sum; }
           }
   Increment( Iteration_Count );
   }
   t = α(t);  //α (t) = at, 0.8 ≤ a ≤ 0.99 or α (t) = t/(1+β t) //
   }
s_o is the approximate solution.
}   // End Algorithm //
```

Figure 26: *Simulated annealing heuristic for the Resource Levelling problem (detailed).*

Experimental Results

We implemented our algorithm and ran it on a 486 PC 50Mhz, under DOS operating system version 6.2. The first digit in each of the names of problems listed stands for the number of levels and the last two digits stands for the branching factor. For example the problem *C3_04* has 3 levels and a branching factor range of 0 - 4. Two problems were generated for each set of parameters e.g. problems C3_04A and C3_04B.

We tested 9 cases as explained in table 11 with the abbreviations used for each case.

Case	**Reduction function**	**Iterations (*nrep*)**	**Neighbourhood**	**Abbreviation**
1	Geometric (G)	Variable (V)	Fixed (F)	GVF
2	Geometric	Variable	Increasing (I)	GVI
3	Geometric	Variable	Decreasing (D)	GVD
4	Geometric	Fixed	Fixed	GFF
5	Geometric	Fixed	Increasing	GFI
6	Geometric	Fixed	Decreasing	GFD
7	Lundy & Mees (L)	1	Fixed	L1F
8	Lundy & Mees	1	Increasing	L1I
9	Lundy & Mees	1	Decreasing	L1D

Table 11: *The tested cases with their abbreviations.*

Figure 27 shows graphical representations of one of the problems and solution as generated by the algorithm. The example presented (C3_04B) is generated from the GVI case. There were 3 orders in its corresponding Master Production Schedule. Each order is a BOM tree with 18 nodes, and thus the profiles presented in Figure 27 and Figure 28 were made from 54 nodes of the 3 orders.

Figure 27 shows the initial profile and Figure 28 shows the generated final profile, which represent an approximate solution. Clearly, Figure 28 is more evenly distributed compared to Figure 27. It is also noted that the height of the profile has been reduced considerably. The difference between the initial and the final cost is 412468, which is an improvement of 14%. The highest peak of this initial profile is 427 units. The highest peak of the final profile represented in Figure 28 is 328 units.

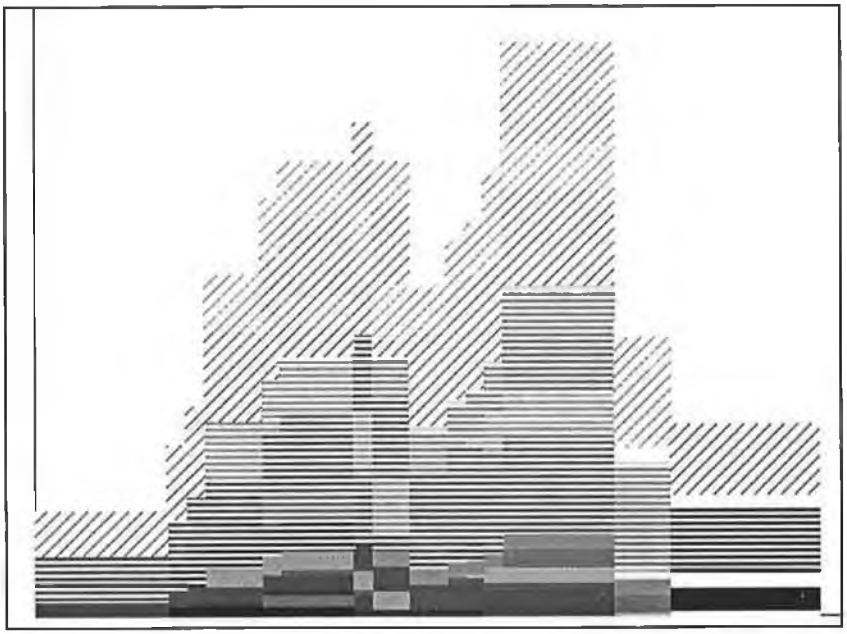


Figure 27:: The initial Resource Profile for problem c3_04B. Sum of squares(cost): 2961511.

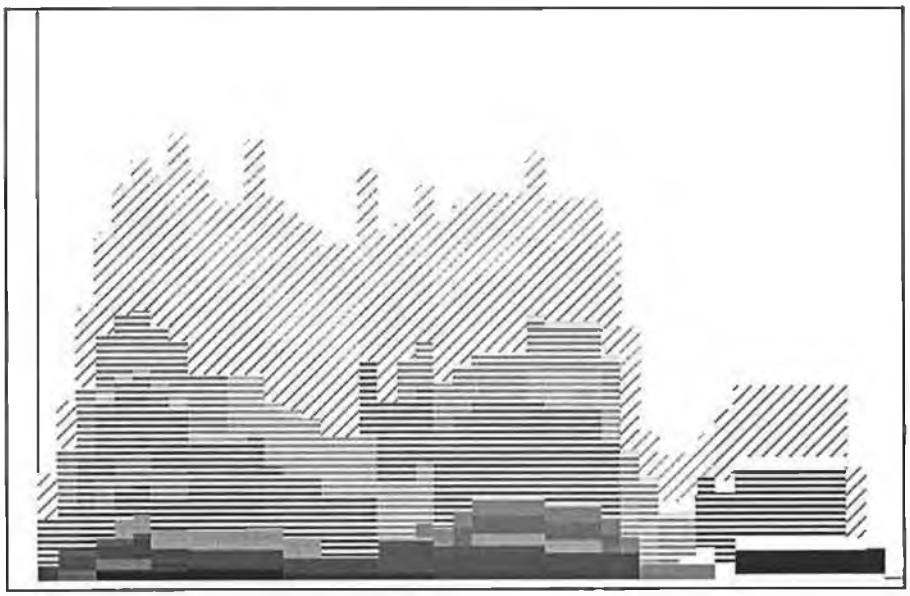


Figure 28: Final Resource Profile for the problem c3_04B. Sum of Squares (cost): 2549043.

Table 12 presents the generated summary of results for problem c3_04B.

			Order 1		Order 2		Order 3	
Activity	Duration	Initial Start	Resource Level	Final Start	Resource Level	Final Start	Resource Level	Final Start
1	0	0	0	11	0	11	0	11
2	10	0	7	7	21	7	21	7
3	5	0	0	12	0	12	0	12
4	5	0	7	12	21	12	21	12
5	16	0	4	8	12	8	12	8
6	18	0	11	2	33	3	33	1
7	5	0	4	7	12	7	12	12
8	11	0	14	3	42	9	42	4
9	14	10	19	17	57	17	57	17
10	9	0	5	22	15	22	15	22
11	7	16	9	24	27	24	27	24
12	11	0	2	14	6	3	6	1
13	9	0	14	15	42	3	42	24
14	16	18	11	21	33	19	33	19
15	14	11	0	13	0	20	0	17
16	11	24	3	30	9	32	9	29
17	8	34	17	34	51	36	51	35
18	0	42	0	45	0	43	0	44

The Starting Cost : 2961511
The Final Cost : 2549043
Planning Horizon : 45
CPU Time (Sec) : 21

Table 12: Summary of results for problem c3_04B.

For brevity, tables of results for both cases are listed in appendix B. It should be noted that, solutions obtained by applying the same set of parameters (same case) to the same problem might differ slightly due to the random selection of possible solutions.

The *performance* of a solution is the percentage of improvement in cost value associated with that solution in relation to the cost value associated with the initial solution.

$$\text{i.e.} \quad Performance = \left(\frac{\text{Initial cost - Final cost}}{\text{Initial cost}} \right) x100$$

Cases L1I and GVI showed significantly better results (performance) compared to the other cases as shown in Figure 29. We have only shown the graphs of GVI, L1I, L1D and GFI for clarity. The graph of the performances of all cases is presented in appendix B.

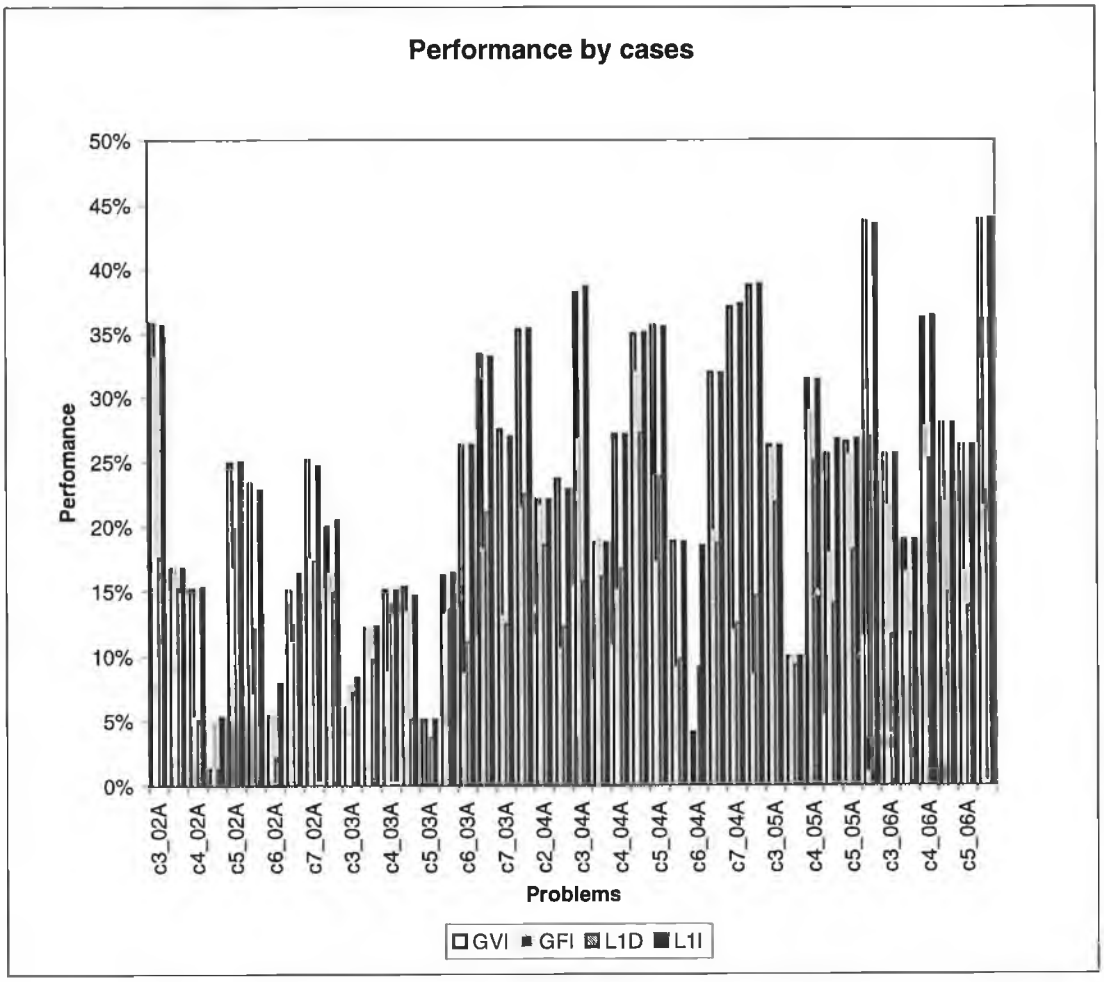


Figure 29: *Performance of the algorithm by cases*

For most cases, the time complexity of the algorithm shows an increase with the number of activities (nodes) involved (Figure 30).

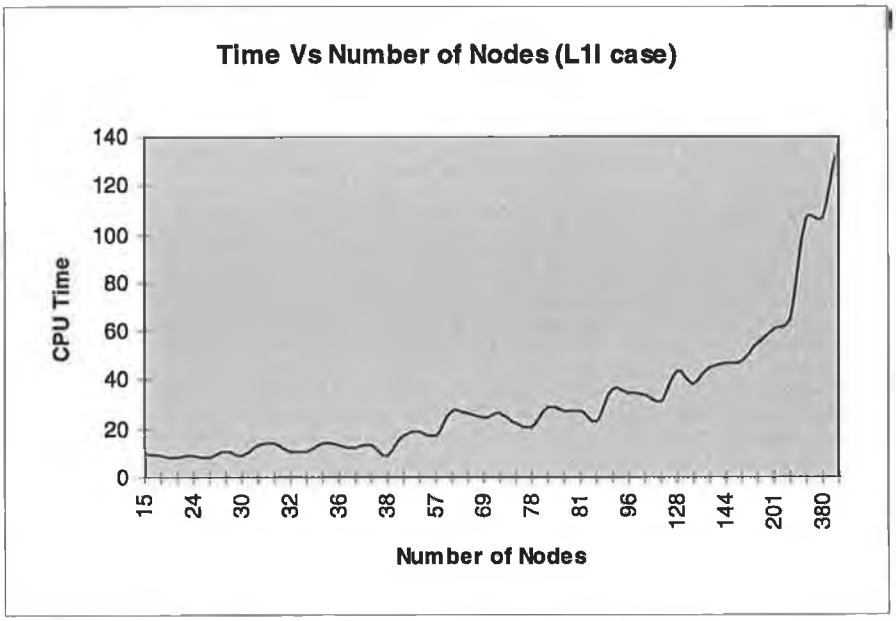


Figure 30: *CPU Time by number of Nodes for L1l case.*

However, this increase is not gradual because the CPU time can also be affected by other factors such as the magnitude of the cost value. Higher values of the initial cost means higher values of the initial temperature and hence more iterations of the algorithm. The dependency of CPU time on the magnitude of the cost value can be seen from Figure 31, where a considerable number of problems shows an increase with the cost value. It is important to note that, the CPU time does not necessarily increase with the magnitude of the cost value, the cost value is just one of the factors which can cause this variation.

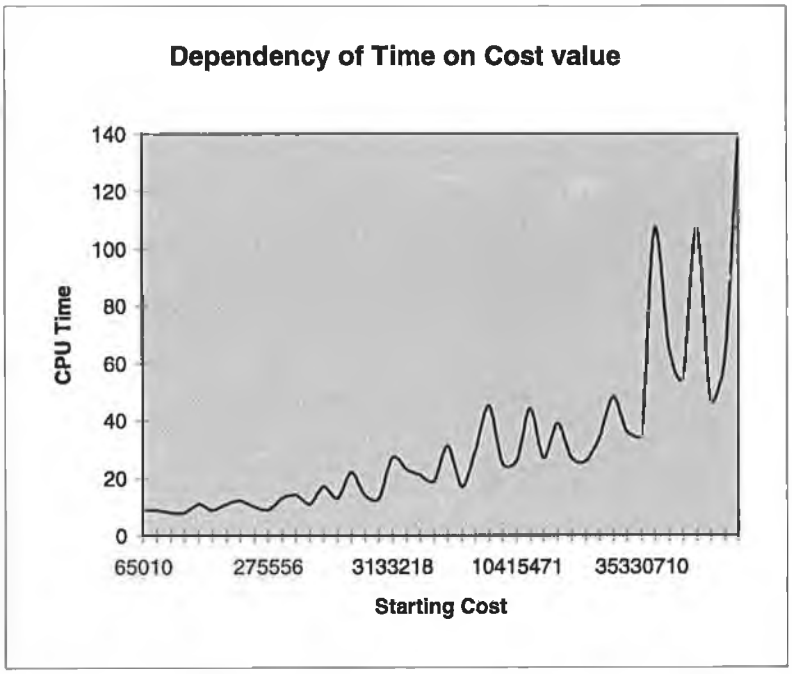


Figure 31: *Time V's Starting Cost for L1I case*

The time complexity was also found to increase with the concentration of the nodes on the planning horizon i.e. the ratio of the number of nodes in the project to the length of the planning horizon as shown in Figure 32.

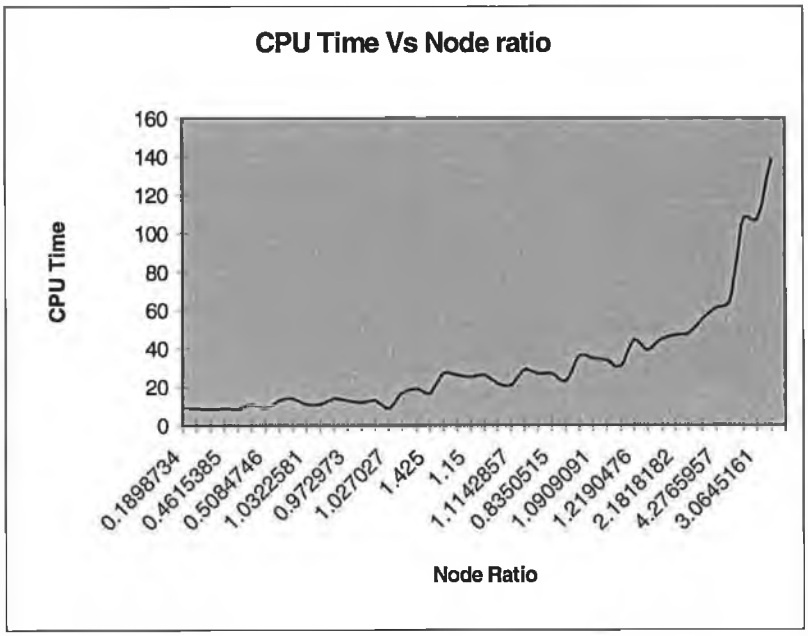


Figure 32*: Dependency of Time on node ratio (case L1I)*

In general, given that the best set of parameters has been selected, the efficiency of the algorithm depends on a number of factors. These include the following;

- **Initial solution**: The closer the initial solution is to the final solution, the faster the algorithm will converge. That is, the algorithm stops before completion of all iterations if there is no improvement in the solution after m^3 number of iterations (Convergence test).
- **Slack** (Floating time): The amount of slack available in each activity determines the size of the searching space.
- **Node ratio**: The ratio between the number of activities (N) and the length of the planning horizon (H) i.e. $^{N}/_{H}$. The higher the ratio the higher the concentration of activities in the planning horizon and hence the trend shown in Figure 32.
- **Resource Levels**: The magnitude of the cost value can affect the number of iterations as explained previously.

Conclusion

- Simulated annealing technique has shown promising results with considerable improvement in the solution within a reasonable time scale.
- The time complexity of the algorithm depends on the structure of the problem i.e. the number of nodes and length of the planning horizon. Also, (in a considerable number of cases) it depends on the magnitude of the cost values (resource levels) involved.
- Careful selection of the heuristic parameters is necessary for the success of the algorithm. Cases L1I and GVI were found to yield close and significantly better results.

Further Research

- Application of other heuristic techniques such as Tabu search may lead to better results. Since the problem is NP-Hard, we could not get optimal solutions to problems of realistic size for comparisons. However, the use of the Integer Programming (IP) formulation on small sized samples, may help to give us an idea of the efficiency of our heuristic.
- We tested a total of 44 problems, which were generated randomly. More problems and specifically real life ones may lead to a better and clearer picture about the characteristics of the problem.
- In the extraction of orders from the MPS table, we assumed that each order involved the whole BOM tree. This is not always the case as is shown in the MPS table (Table 2), where some of orders are placed for only parts of the BOM tree (such as Engine or Frame). We need to incorporate this fact into our algorithm in order to get a more realistic view of the problem.

- Despite the current promising results, there are still some improvements to be made to our program code. At present, a considerable amount of time is used in storing and retrieving data from their structures. More optimisation of the structures in terms of storage and time are bound to provide an improvement in the general performance of the heuristic.
- Search for the exact algorithm for this problem by applying the theory of *polyhedral combinatorics*. This technique has been successfully applied to some NP-Hard combinatorial problems such as the Travelling Salesman and Linear Ordering problems [KC95] .

Chapter 5

TABU SEARCH AND THE STOCHASTIC HEURISTIC TECHNIQUES FOR RLP

Tabu search (TS) has its antecedents in methods designed to cross boundaries of feasibility or local optimality normally treated as barriers and systematically to impose and release constraints to permit exploration of otherwise forbidden regions. The algorithm is named after a table used to record the states visited by the algorithm and which is used to prevent the algorithm from cycling through the previous states.

Notations:

- X : A set of all feasible solutions
- c(x): The cost associated with the solution $x \in X$
- TABU: A set of history records of solutions processed with respect to a particular attribute. This attribute can be recency, frequency, quality, influence or a mixture of some of them.
- N(TABU, x): A set of solutions that can be selected from X given that the current solution is x.
- History (TABU) determines which solutions can be reached by a move from the current solution i.e. selecting x^{next} from N(TABU, x^{now}).
- N(x^{now}): A subset of N(TABU, x^{now}) which determines a neighbourhood of x^{now} where a selection of the next move is to be made.
- MAXITER: Fixed number of iterations that the algorithm is set to be repeated.

The pseudo-code given in Figure 33 shows how the TS algorithm operates [C93]

```
// Initialisation
i) Select a starting solution x^now ∈ X.
ii) Record the current best solution by setting x^best = x^now.
iii) Define best_cost = c(x^best).
iv) Initialise a history record TABU to empty
// Choice and termination
while( not termination criteria)
{   i) Determine a neighbourhood N(x^now) ∈ N(TABU, x^now).
    ii)Select x^next∈N(x^now) to minimise c(TABU,x) over this set of neighbours
// Update
        i) Re-set x^now = x^next
        ii) if( c(x^now) < best_cost )
                a) Set x^best = x^now
                b) Define best_cost = c(x^best)
        iii) Terminate by a chosen iteration cut-off rule.
}
```

Figure 33. General Tabu Search Algorithm.

Tabu Search approach to RLP

Choice of a move

Many of the TS heuristics employ swap/insert moves, such as Lin's 2-Opt Swaps. Normally, swap/insert moves would require us to change the sequence of the jobs. However as we have a structured list of jobs (BOM) for each order, as defined by the precedence relationships, there is no possibility of carrying out sequence changes within orders.

The neighbourhood definition, below, allows the following moves.

- Increase the starting time of a job by one, i.e. move the starting time one step forward. This is only possible if the job is not currently at the end of the feasible starting time interval. A job can be scheduled to start between the earliest starting time (EST) and EST+TFloat, where TFloat is the total floating time of the job as calculated from the critical path method.

- Move the starting time of a job one step backward in time. This is only possible if the job is not currently at the beginning of its feasible starting interval.

Neighbourhood structure

Suppose that we have n jobs and x_i stands for the current starting time of job i.

Then $x^{now} = \{x_1, x_2, ..., x_n\}$ and the change in starting time of one of these jobs (one step) constitute a move. The neighbourhood is chosen to be the set of all separate single step moves of each job that are not restricted (not 'tabu'). Suppose we denote a single step move of job i from the current solution by x_i^*, then $x^{next} \in \{ (x_1^*, x_2, ..., x_n), (x_1, x_2^*, ..., x_n), ..., (x_1, x_2, ..., x_n^*) \}$ provided that the attribute of the selected solution is not in the TABU list. The neighbourhood therefore consists of at most *n* moves.

Solution attribute

We use the "recency" criteria by restricting the solution to avoid points that have been recently selected. This is done by maintaining a queue structure of a fixed length (TL) and a move can only be selected if it is not in the queue. The selected move is then placed in the queue and the last element deleted (released) if the queue is full. Our input consists of jobs from different orders[AM96]. Therefore an entry in the TABU list consists of three values;

- The number of the order in process
- The number of a job in that order that has moved
- The new starting time of the moved job.

These three values are unique for each move and therefore sufficient to identify any move.

This attribute is not absolute as we allow a special type of move to violate the 'tabu' rule (aspiration criterion). We keep track of the best move i.e. a move that has produced the best-cost reduction so far (great_change). If the next move results into a cost reduction that is better than great_change, we select this move even if it is in the TABU list.

Cost criteria

We minimise the sum of squares of the resource levels at each time unit in the planning horizon [BK62].

Each move must correspond to a decrease in cost from the current resource in order to be considered for selection. The cost change from the current solution is the value;

$$c(x^{next}) - c(x^{now})$$

The best move in the neighbourhood is the one that corresponds to the most negative cost change.

Termination rule

Since the move selection process is deterministic, if no move in the neighbourhood results in an improvement in solution from the current profile, the algorithm would repeat exactly the same move selection process with no improvement. If this happens then the algorithm is terminated to avoid wasting time unnecessarily on this cycling process. If no cycling occurs, the algorithm stops after a fixed number of iterations MAXITER.

The use of Long Term Memory technique

Early cycling can result in a very poor solution, due to premature termination of the algorithm. Furthermore we found, in practical data sets, that most of the solutions were obtained from the first few iterations of the algorithm (see Table 13).

This triggered us to improve the algorithm by repeating the procedure at different starting points called stages. A new starting point is selected at each stage depending on the current structure of the resource profile and is generated by the *'Perturbation Process'* described below. We keep a second list of starting points so as to avoid the repetition of the same stating points at different stages. We call this procedure a perturbation process because it helps to shake up the resource structure of the current stage and therefore avoid cycling through the same series of feasible solutions.

The Perturbation Process.

We identify the point in time on the planning horizon that corresponds to the highest resource level of the current profile (HighPos). We then define an interval of time (Nbr) to the left and right of this point and process the set of jobs that falls within this interval of time (see Figure 34). To avoid spending too much time on this procedure we do not explore all possible moves on this interval but rather stop after the first good move (a move with negative cost change). This move is sufficient to jump out of the cycling process.

This long-term memory process is repeated LT times, i.e. LT stages.

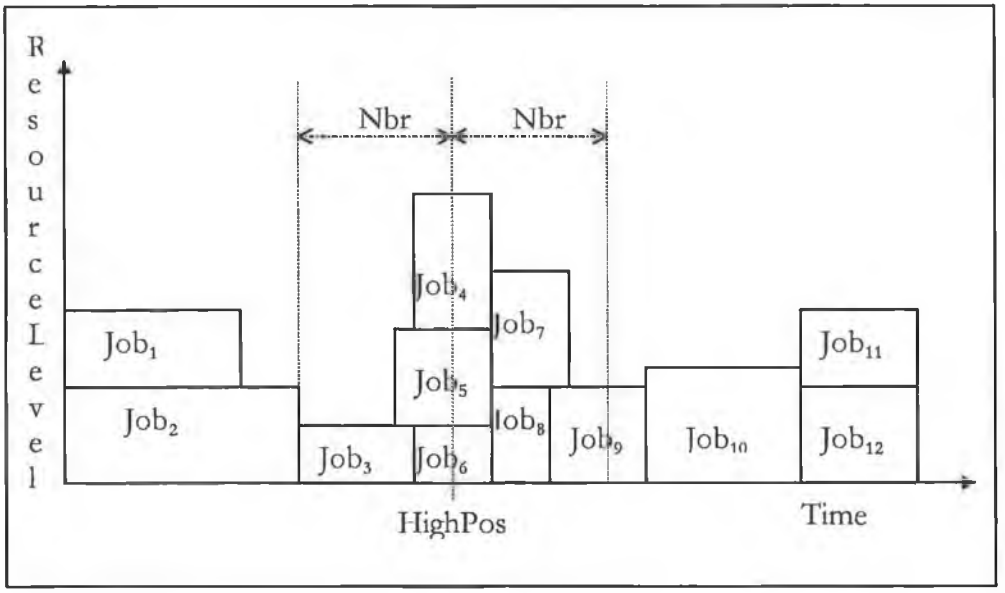


Figure 34: 'Shake up' of the jobs around the highest point

Implementation

```
Tabu-Search_RLP( )
{// Initialisation
  i) Select a starting solution x^now ∈ X.
  ii) Record the current best solution by setting x^best = x^now.
  iii) Define best_cost = c(x^best).
  iv) Initialise a history record TABU to empty
// Choice and termination
Terminate = False; Iterations = 0; cost_change = 0; great_change = 0;
while( iterations <= MAXITER and not Termination )
{   Move = False; iterations++;
    // Get all possible one step moves from the current solution (x^now) and peek the best //
    For each order
        For all jobs in the order
          {cost_change = Get_A_Move( order, job, newStart);//c(x^next)-c(x^now)
            // Move a job by one step (x^next)and return the resulting cost change. //
            if( cost_change < best_change)  // Find Minimum x^next ∈  N(x^now)
              {if(not Taboo(TABU,order,job, newStart) or cost_change<great_change)
                {//Collect the best move x^now on the neighbourhood //
                    best_change      = cost_change;
                    best_moveJob     = job;
                    best_moveOrder = order;
                    best_jobStart    = newStart;
                    Move = True;
                    }
                }
          }
// Update //
// if a move with cost improvement was made i.e. x^now = min{x^next ∈  N(x^now) and c(x^now)< best_cost //
if( Move)
{     best_cost = best_cost + best_change;
      Re-set x^best = x^now;
      if( best_change < great_change) great_change = best_change;
      best_change = 0;
      Push(TABU, best_orderJob, best_jobStart, best_moveJob);
      }
      else  Terminate = True;
}
```

Figure 35: Tabu Search procedure for the RLP.

Implementation of the Long Term Memory Technique

This is done by repeating the Tabu procedure TL times as shown in Figure 36.

```
Tabu_Search_RLP_LT ( )
{
    Best_Height = Big_Value;

    for  stage = 1 to TL do
    {  Tabu_Search_RLP ( Profile, Height );
       if( Height < Best_Height)
           Best_Height = Height;
           ShakeUp ( Profile ); // New starting point //
           }
}
```

Figure 36: Long term memory TS

Get_A_Move function

Before moving a job forward we must check the successors of the job. We must push the successors forward if necessary in order to maintain the feasibility of the resulting solution. These moves might involve a recursive procedure (we employ Depth First Search) to explore all the successors that need to be pushed forward before making a move on the job. Likewise the backward move must first be preceded by the exploration of the predecessors that need to be pushed back before making a move. Thus a single move may sometimes result in propagating large changes in the whole structure of the resource profile.

This function calculates the result of the change and returns the associated value of cost_change.

Perturbation Function

```
Boolean Perturbation (Profile )
{  Set x^now to the cycling solution;
   Set cost_change = 0;
   Determine the width of the interval (Nbr);
   Find the highest point ( Profile, HighPos);
   while( More orders and cost_change ≥ 0 || LT_Tabu)
   { while( More jobs in the order and cost_change ≥ 0 || LT_Tabu)
        { if( HighPos-Nbr ≤ job.start ≤ HighPos+Nbr)
             { start = job.EST;
               while( start <= job.start+job.TFloat and cost_change ≥ 0)
               {// get x^next ∈  N(x^now); //
                 cost_change = Get_A_Move( Profile, order, job, start);
                 start++;
                 Re-set x^next = x^now;
                 Push( TABU, order, job, start);
                 }
             }
        }
    }
   }
if( cost_change < 0 ) return True;
else return False;
}
```

Figure 37: Perturbation Function. Helps to jump out of the cycling process.

Summary of Results

The test problems were generated randomly as described in [AM96]. The name of each problem denotes the number of levels of the BOM tree, the *branching factor* (number of children) on each node and the number of jobs in the problem. For example, the problem A312a14 denotes a BOM tree with 3 levels, a branching factor in the range from 1 to 2 and having 14 jobs. The algorithms were implemented on a Borland C++ compiler and run on a Pentium 120Mhz machine. We compare the results with the optimal solutions obtained by the MIP formulation [AM96b] where the model was generated and solved by the XPRESS-MP package [D94] . We assumed that all tested problems are generated from one order so as to decrease the size of the Linear Programming formulation and hence be able to get an optimal solution from the MIP procedure.

The following table shows the solutions of different problems and the iteration at which each solution was found. The value of MAXITER used in this case was 1000 and Nbr = 1 and the number of stages LT = n.

Problem	Tabu Solution	Optimal solution	Iteration count.
A213a10	67	67	1
a213b12	62	62	7
a214a13	56	56	1
a214b20	108	108	3
a215a20	97	87	19
a215b20	104	88	3
a216a21	137	116	10
a216b15	72	63	3
a217a22	154	125	10
a217b13	87	81	2
a218a22	136	109	1
a218b16	137	129	3
a312a14	72	61	1
a312b10	59	53	1
a313a15	67	66	1
a315b11	48	48	18

Table 13: Iteration at which the solution was found.

The maximum number of iterations at which a solution was found is 19 and most of the solutions were found in the first few iterations. Consequently, the value of MAXITER was chosen to be 30. Increasing the LT beyond n did not seem to yield better results and therefore LT was fixed at n.

Further Improvement

It was also noteworthy that, the solution improvement depended considerably on the value of Nbr. After several tests on different problems, it was found that these values of Nbr vary between 0.5 and 5 with a bias to the lower end. We decided to repeat the procedure for values of Nbr in that interval in an increment of 0.5 and pick the best solution.

Problem	Number of jobs	Approx. Minimum height (from TS)	Optimal Height (from MIP)	Time taken by MIP (Sec)	Time taken by TS (Sec)
a213a10	10	67	67	0.5	0.06
a213b12	12	62	62	14	0.11
a214a13	13	56	56	1	0.10
a214b20	20	108	108	2	0.33
a215a20	20	97	87	474	0.27
a215b20	20	104	88	522	0.65
a216a21	21	119	116	25	0.44
a216b15	15	72	63	6	0.16
a217a22	22	146	125	84	0.44
a217b13	13	87	81	27	0.05
a218a22	22	119	109	3191	0.44
a218b16	16	137	129	29	0.11
a312a14	14	61	61	7	0.11
a312b10	10	53	53	1	0.06
a313a15	15	67	66	24	0.22
a315b11	11	48	48	10	0.11

Table 14: Comparison of TS to optimal results.

Problems a216a21, a217a22, a218b16, a312a14 and a312b10 were considerably reduced. Figure 38 summarises the performance of the two sets of results in comparison with the optimal solutions.

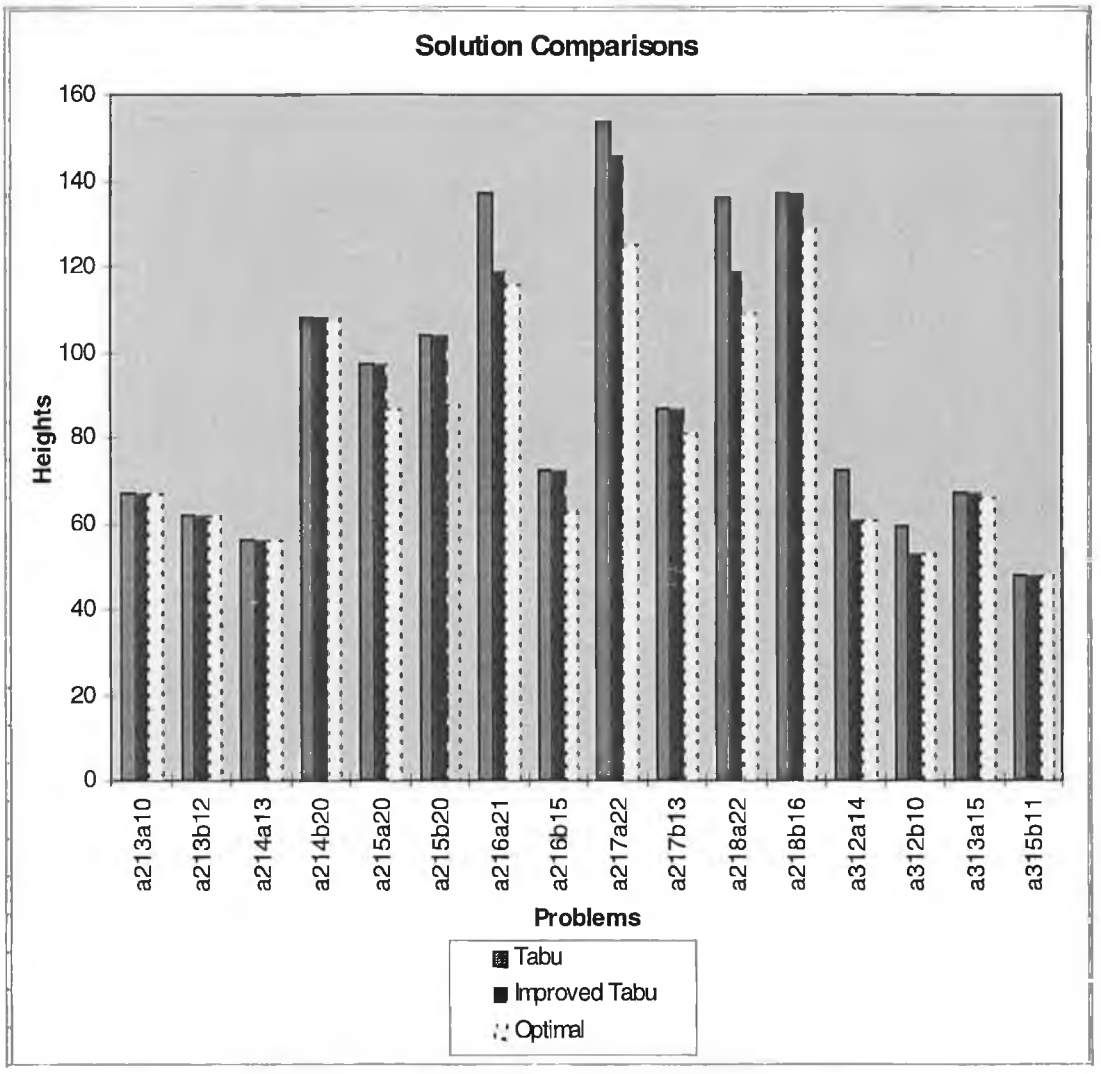


Figure 38: Comparison of solution performances between TS and Improved TS

Improved Tabu Search is clearly better than the Tabu Search as shown by the heights of their corresponding histograms in Figure 38.

Stochastic algorithm

When the deterministic algorithm stops at a local optimum, we select a set of objects at random, multiply their weightings by a factor (M) and reapply the deterministic algorithm. This may completely change the structure of the current solution and lead to a different starting point in the searching space. The concept of stochastic perturbations can be applied to a wide range of global optimisation problems such as the crossing number problem [GM93].

We can convert a deterministic heuristic into a stochastic algorithm as follows;

- Select a set of solution attributes say jobs, from the current solution and give them weighting of M (where M is a sufficiently large number).
- Find a local optimum, set M = 0 and re-optimise to get a candidate solution.
- Repeat the procedure until a CPU time bound is reached and print the best candidate solution found.

Stochastic algorithm applied to RLP

Improved Tabu Search (ITS) was selected as the deterministic algorithm. We applied the perturbation by selecting jobs at random and adding a big M value (Initial height on the profile) to the resource levels of the selected jobs. We re-optimised the problem and restored the resource levels to their correct values. This new profile was then re-optimised to find a new solution. We repeated the procedure a fixed number of times (steps), say S and recorded the best solution.

```
Stochastic_RLP ( )
{    Improved_Tabu_Search( Profile, High); // Perform the Tabu search //
     Best_Profile = Profile;
     for  step = 1 to S do
     {
          Select_random( Profile, jobs);
          Add_weights ( Profile, jobs, Initial_Height);
          Improved_Tabu_Search( Profile, High); //re-optimise with weights//
          Remove_Weights( Profile, jobs, Initial_Height);
          Improved_Tabu_Search ( Profile, High); // optimise //
          if( High < Best_Height)
          {    Best_Height = High;   // collect the best height //
               Best_Profile = Profile; // collect best solution //
                 }
       }
}
```

Figure 39: Stochastic RLP.

Summary of results (Stochastic case)

The following tables show the effect of perturbations on each step of the stochastic algorithm for the problem a217a22 with the planning horizon of 18. The last column of both tables shows the percentage improvement of the solutions vis-à-vis the initial heights.

Step	Perturbed job	Resulting Height	%Improvement
Initial (0)	0	146	0%
1	15	188	-29%
2	2	146	0%
3	3	146	0%
4	7	170	-16%
5	10	150	-3%
6	14	150	-3%
7	19	155	-6%
8	11	142	3%
9	10	136	7%
10	14	146	0%
	Best Height	136	
	Optimal Height	125	

Table 15: Steps **in a** 1-job **stochastic perturbation process**

The best height in this case (136) was obtained on the 9th step, which is the penultimate step. More steps might have produced a better solution. The best height found (136) is still significantly far from the optimal value (125) but it is a good improvement from the ITS value which was 146.

Step	Perturbed job	Resulting Height	%Improvement
Initial (0)	0	146	0%
1	10, 8	153	1%
2	19, 4	149	3%
3	20, 17	168	-9%
4	16, 4	156	-1%
5	3, 6	172	-12%
6	15, 2	153	1%
7	3, 14	149	3%
8	20, 19	157	-2%
9	2, 4	146	5%
10	14, 15	155	-1%
	Best Height	146	Total time
	Optimal Height	125	

Table 16: Steps **in a** 2-jobs **stochastic perturbation process.**

Two-job perturbations (Table 16) did not produce any improvement in the solution values after 10 steps. This means that all the selected jobs for perturbation so far only succeeded in changing the structure of the initial profile to a worse starting point. However, we have only presented 10 steps of the algorithm and more steps might result in a better solution.

Iterations	Perturbed job	Resulting Height	%Improvement
Initial (0)	0	146	0%
1	9, 1, 4	131	15%
2	3, 10, 19	144	6%
3	11, 7, 5	150	3%
4	1, 17, 5	162	-5%
5	4, 5, 6	177	-15%
6	19, 4, 16	204	-32%
7	10, 11, 8	159	-3%
8	4, 9, 2	149	3%
9	12, 15, 6	150	3%
10	17, 2, 14	149	3%
	Best Height	131	Total time
	Optimal Height	125	

Table 17: Steps **in** 3-jobs **stochastic perturbation process.**

The best height found in this case was better than the rest of the tables above (131) and was found at the 1st step.

The question arises as to what number of steps and perturbations should the algorithm perform in order to guarantee the best solution. Obviously, the larger the number of steps, the better is the chance of improving the solution, albeit at the expense of higher CPU times. Table 15, Table 16 and Table 17 indicate that some of the "number of perturbations per job" are better than others (3 jobs perturbations gave a better solution in this case). We are not only interested in the best solution but also in the number of perturbations that will give us the best solution using the least number of steps. Since this is a stochastic process, we need to look at the set of parameters (number of steps and perturbations) which will result in a more stable solution i.e. a solution with more repetitions of the best value than others.

Figure 40 and Figure 41 present the distribution of problem a414b22 for 1-job and 4-jobs perturbations respectively. Figures for the rest of perturbations are presented in the appendix C. The heights at each step have been sorted in increasing order and 20 steps were performed on each set of perturbations.

In Figure 40 the best solution was found at step 2 with a solution value of 74. However, the distribution of solutions was uneven with a standard deviation (std) of 11.13 and a mean of 87.2, which is far from the optimal value of 71. In Figure 41 the best solution was 73 which is better than in the previous Figure and was found at step 15. The heights are better distributed than Figure 40 as shown in the figure with the mean of 86 and std 7.62.

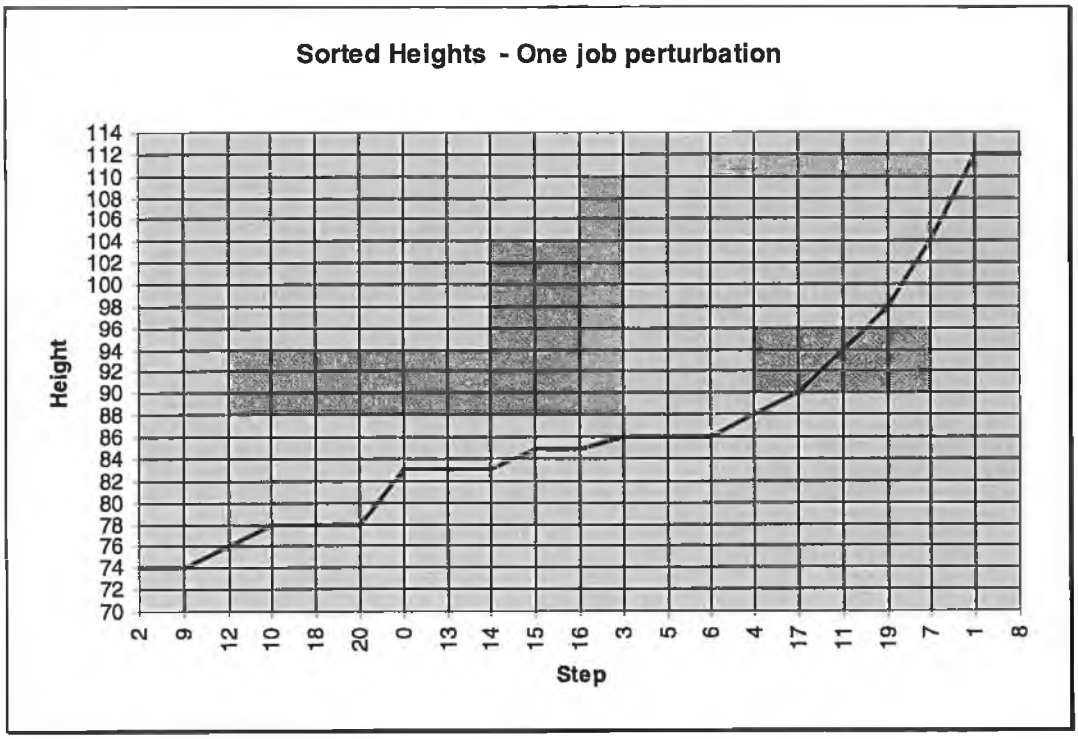


Figure 40: Heights at each step for Problem a414b22. Mean = 87.28, std = 11.13

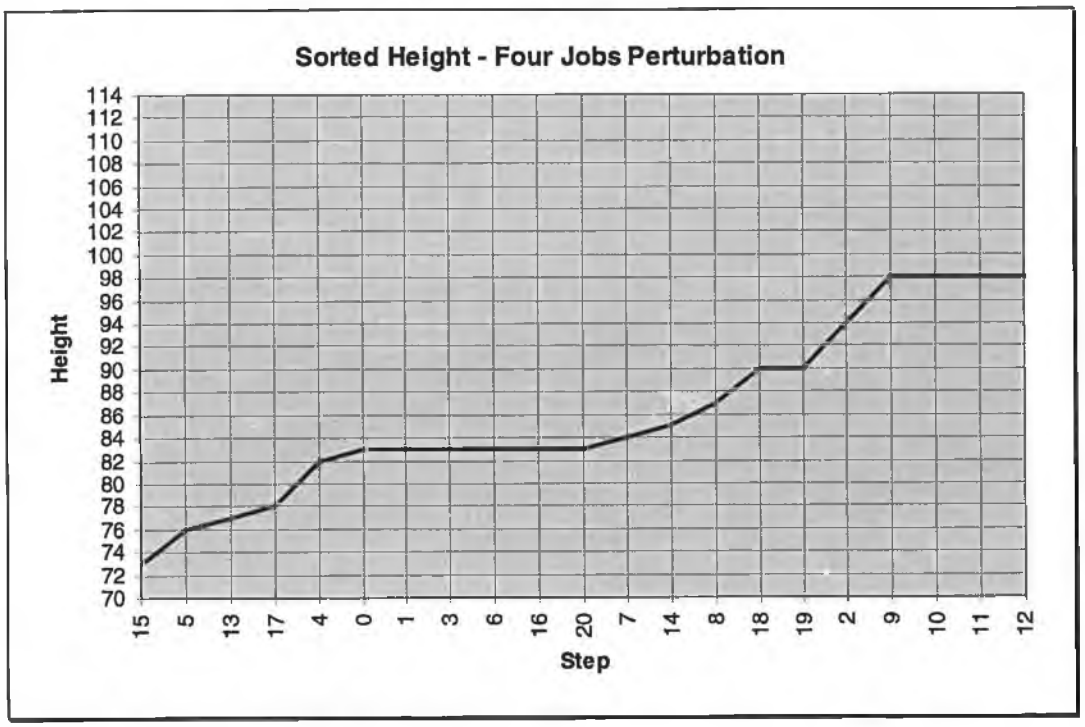


Figure 41:: Heights at each step for problem a414b22. Mean = 86 and std = 7.62

We experimented with different problems and with up to 10-jobs perturbations. (See figures in the appendix C). We did not find any relationship between the number of steps and the number of jobs perturbed. The search space was very uneven as shown in Figure 42.

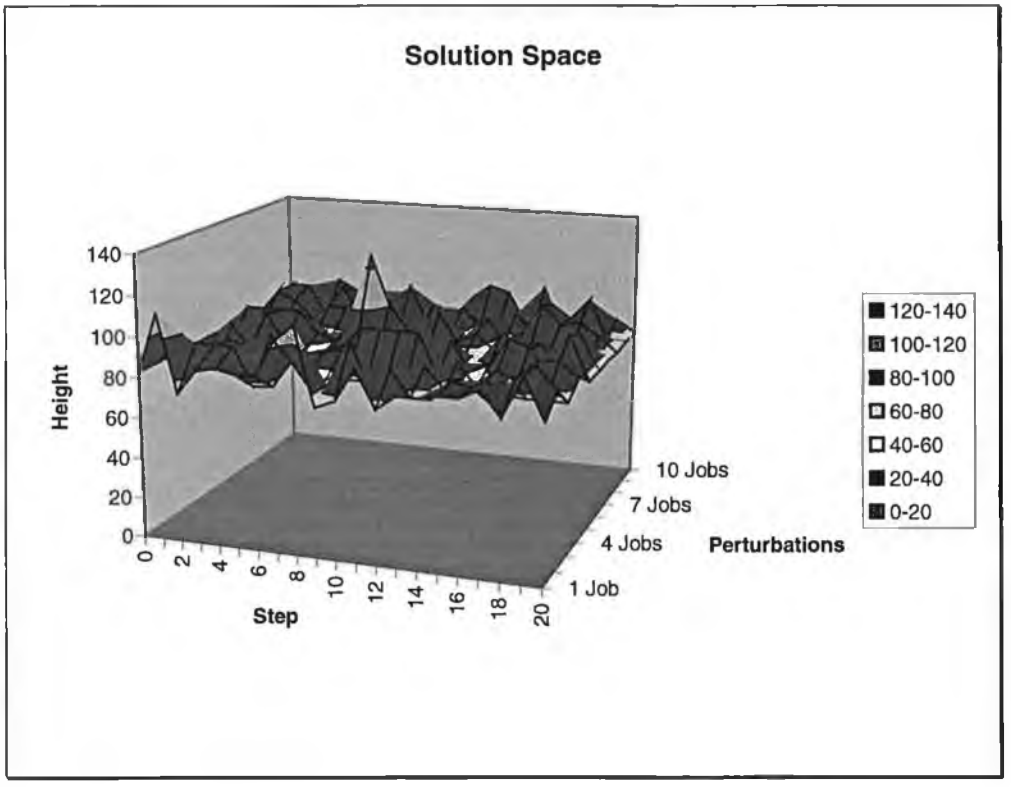


Figure 42: Solution space for problem a414b22.

Since there are so many local minima (Figure 42), the solution value largely depends on the chosen starting point. There are many starting points, each of which can lead us to a completely different local optimum.
However, no good solution was found at a step count larger than the number of jobs in the problem. It was consequently decided that the number of steps for each problem should be set to the size of the problem (n).

For the sake of comparison to the ITS case, we present the summary of results for 16 test problems. We show these results for 1-job and 2-job perturbations and a fixed number of steps (n).

1 - job Perturbation					
Problem	**Initial Height**	**ST Height**	**ST Time(Sec.)**	**Optimal**	**% Improvement**
a213a10	71	67	0.71	67	6
a213b12	78	62	1.65	62	21
a214a13	81	56	2.36	56	31
a214b20	123	108	13	108	12
a215a20	150	92	16.1	87	39
a215b20	147	91	14	88	38
a216a21	231	119	12.57	116	48
a216b15	112	63	4	63	44
a217a22	231	130	24	125	44
a217b13	130	81	3	81	38
a218a22	177	118	27	109	33
a218b16	204	136	11.9	129	33
a312a14	72	61	2.75	61	15
a312b10	59	53	1	53	10
a313a15	98	66	6	66	33
a313b11	94	48	3.52	48	49

Table 18: Summary of results for the ST algorithm, 1 - job perturbation.

2 - Job Perturbation					
Problem	**Initial Height**	**ST Height**	**ST Time (Sec.)**	**Optimal**	**% Improvement**
a213a10	71	67	0.71	67	6
a213b12	78	62	2	62	21
a214a13	81	56	3	56	31
a214b20	123	108	12	108	12
a215a20	150	89	16	87	41
a215b20	147	93	13	88	37
a216a21	231	119	13.4	116	48
a216b15	112	63	4.18	63	44
a217a22	231	131	22	125	43
a217b13	130	81	2.69	81	38
a218a22	177	117	17.14	109	34
a218b16	204	137	7	129	33
a312a14	72	61	2.64	61	15
a312b10	59	53	1.5	53	10
a313a15	98	66	5.5	66	33
a313b11	94	48	1.92	48	49

Table 19: Summary of results for the stochastic algorithm, 2 - jobs perturbation.

The solution times were reasonable compared with the amount of improvement of each case from the initial value (10 problems were solved to optimality in the first table and 7 in the second table). Figure 43 shows the comparison of ST to the TS case.

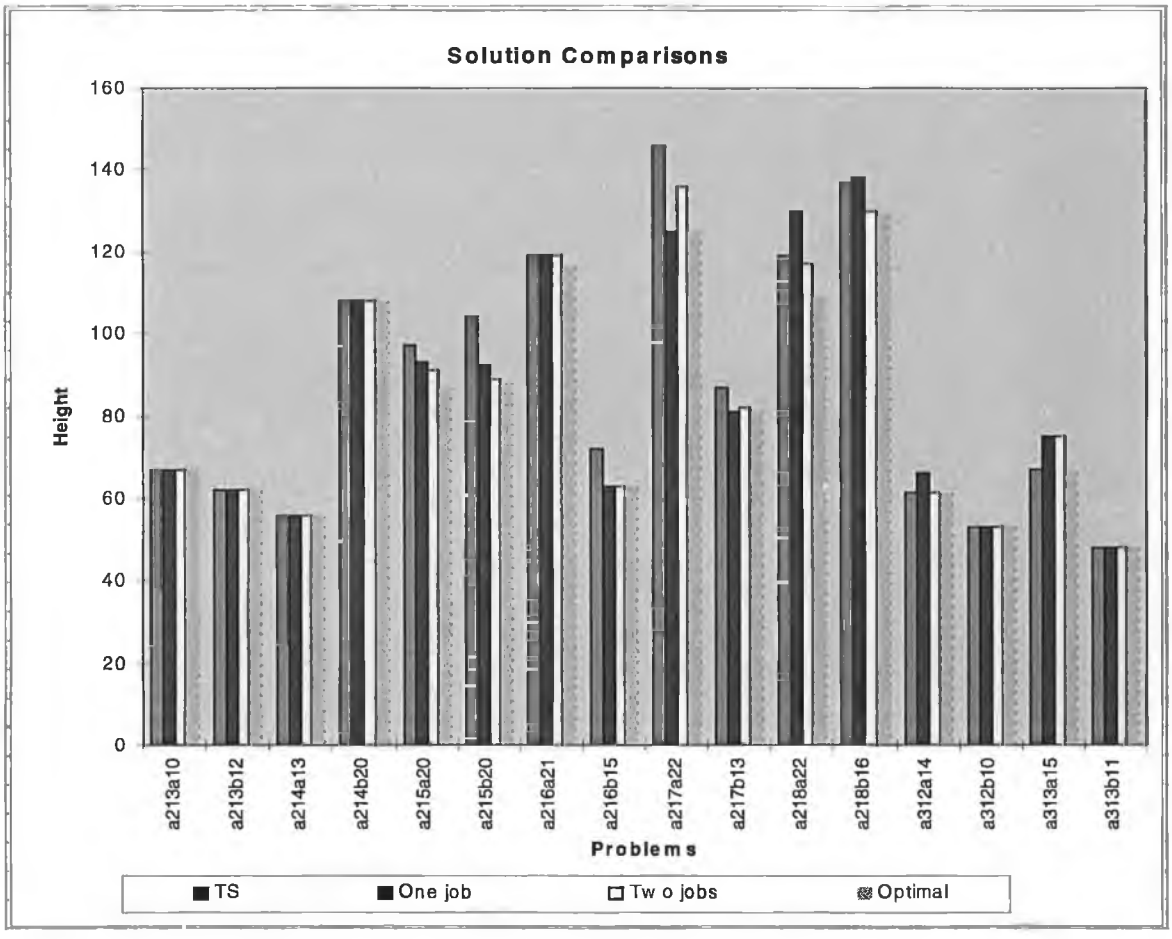


Figure 43: Solution comparisons on TS, 1 and 2 - job stochastic algorithms.

Both 1-job and 2-job perturbations are obviously better in terms of solution improvement than the general TS algorithm. The increase in CPU time of the ST case over the TS is clearly justified by the effect of the improvement in the solution values.

The question that still remains to be answered is "How many jobs should be perturbed for a given problem so as to yield the best result? ". Intuitively, this should be proportional to the number of jobs (n) and the length of the planning horizon (T). The sparser the resource profile, the more jobs are expected to be perturbed to change the resource profile.

Since we have fixed the number of steps, the best number of perturbations should now correspond to the case that gives the best result earlier than the rest. Using data for problem a414b22 (see appendix C), at each step we record the best solution so far for all job perturbations (1 to 10 job perturbations). Figure 44 summarises the results of this experiment for each job-perturbation.

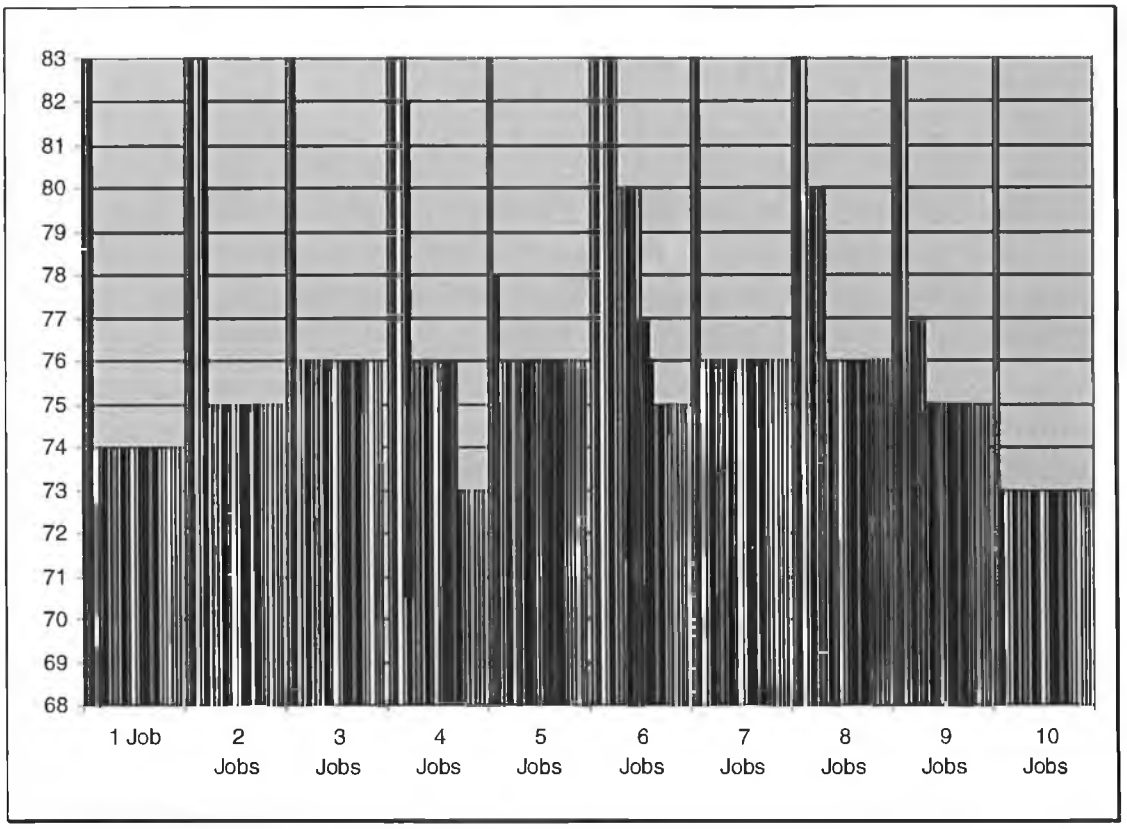


Figure 44: Summary of results for problem a414b22. Steps at each perturbation.

Each block in Figure 44 represent all the steps involved in a specified perturbation case. The 10-jobs perturbations were the best. This is so because the best result (73) was found earlier than the rest of the cases as shown by the shape of its corresponding block.

The experiment above gave us the following set of data for problem a414a22;

- Number of jobs, n = 22
- Planning horizon, T = 35
- The best number of jobs to be perturbed, p = 10

Since we expect the value of p to decrease with the concentration of jobs on the resource profile, we estimated the value of p as follows;

Let $^{35}/_{22} = q$, Then a constant q corresponds to the value of $p = 10$.
We can generally say that the ratio $^{T}/_{n}$ corresponds to the value of p perturbations.
Therefore, $p = (10x^{T}/_{n})/_{q} = (350^{T}/22_{n})$ or,

$$\boxed{p = {}^{15.9T}/_{n}}$$

After experimentation with different problems, the constant value of 15.9 was found to be a good approximation in calculating the value of p as shown in Table 20.
We present the summary of results for all 37 test problems for up to 8 job-perturbations (Table 20).

Problem	T	Initial H	TS	ST1	ST2	ST3	ST4	ST5	ST6	ST7	ST8	Opt.	p
a213a10	11	71	67	67	67	67	67	67	67	67	67	67	7
a213b12	15	78	56	52	54	56	54	52	56	54	54	52	8
a214a13	20	81	56	56	56	56	56	56	56	56	56	56	10
a214b20	20	123	108	108	108	108	108	108	108	108	108	108	6
a215a20	21	150	97	92	89	91	91	91	91	91	90	87	7
a215b20	16	147	104	91	93	93	93	91	90	89	89	88	5
a216a21	18	231	119	119	119	119	119	117	119	119	117	116	5
a216b15	14	112	72	63	63	63	63	63	63	63	63	63	6
a217a22	18	231	146	130	131	128	131	130	128	128	129	125	5
a217b13	16	130	87	81	81	81	81	81	81	81	81	81	8
a218a22	20	177	119	118	117	117	115	117	115	115	118	109	6
a218b16	19	204	137	136	137	137	132	137	137	137	132	129	7
a312a14	18	72	61	61	61	61	61	61	61	61	61	61	8
a312b10	17	59	53	53	53	53	53	53	53	53	53	53	11
a313a15	24	98	67	66	66	66	66	66	66	66	66	66	10
a313b11	28	94	48	48	48	48	48	48	48	48	48	48	16
a313b21	20	160	125	119	119	119	121	121	120	119	121	118	6
a314a22	27	165	118	113	113	113	112	113	113	113	113	111	8
a314b20	24	234	151	150	150	150	150	150	150	150	150	150	8
a315a21	21	151	63	61	59	61	59	60	59	61	60	57	6
a412a22	34	105	74	71	71	71	71	71	71	71	71	63	10
a412b20	26	113	88	84	84	84	80	84	84	84	84	77	8
a413a22	32	134	76	73	72	72	72	74	74	73	74	69	9
a413b20	29	114	91	83	82	79	82	84	82	84	84	77	9
a414a20	30	126	86	86	83	86	86	86	86	86	86	83	9
a414b22	35	133	83	73	73	76	72	73	76	73	72	71	10
a415a21	34	136	107	104	104	105	104	104	104	104	104	101	10
a415b20	26	126	94	82	80	86	82	80	81	80	80	80	8
a512a22	33	129	97	88	82	82	88	88	88	88	82	80	9
a512b22	35	137	81	74	74	66	74	68	66	74	74	65	10
a513a26	42	129	100	95	95	98	98	98	98	98	98	90	10
a513b22	36	136	123	123	123	123	123	123	123	123	123	123	10
a612a21	51	79	72	72	72	72	63	72	72	72	62	62	15
a612b16	34	97	49	49	49	49	49	49	49	49	49	49	13
a712a20	52	87	66	59	59	59	59	59	59	59	59	59	16
a712b20	44	76	55	54	54	54	54	54	54	54	54	54	14

KEY:
Optimal
STx — Stochastic case with x-job perturbations
Best H (Best Height)
TS — Tabu Search

Table 20: Summary of results for up to 8 jobs perturbations.

The last column shows the calculated values of the number of perturbations (p) for each problem. Apart from the cases which correspond to the value of $p > 8$, most of the other cases had their best solutions at or close to p perturbations.

The dark colour represents cases and the positions in which optimal solutions were found.

The table below shows the summary of results obtained by applying the formula for p.

Problem	TS	Opt.	p	STp	SA	CPU secs, STp
a213a10	*67*	*67*	7	67	67	2
a213b12	*56*	*52*	8	52	52	3
a214a13	*56*	*56*	10	56	56	4
a214b20	*108*	*108*	6	108	115	18
a215a20	97	*87*	7	91	94	21
a215b20	104	*88*	5	90	93	22
a216a21	119	*116*	5	117	120	25
a216b15	*72*	*63*	6	63	69	5
a217a22	146	*125*	5	128	149	34
a217b13	*87*	*81*	8	81	83	4
a218a22	119	*109*	6	115	122	27
a218b16	137	*129*	7	137	137	12
a312a14	*61*	*61*	8	61	61	4
a312b10	*53*	*53*	11	53	53	1
a313a15	*67*	*66*	10	66	66	8
a313b11	*48*	*48*	16	48	48	3
a313b21	125	*118*	6	120	136	18
a314a22	118	*111*	8	113	120	39
a314b20	*151*	*150*	8	150	150	22
a315a21	63	*57*	6	59	70	21
a412a22	74	*63*	10	71	80	43
a412b20	88	*77*	8	84	113	26
a413a22	76	*69*	9	70	134	38
a413b20	91	*77*	9	80	114	26
a414a20	86	*83*	9	83	91	26
a414b22	83	*71*	10	72	85	57
a415a21	107	*101*	10	103	113	38
a415b20	94	*80*	8	80	80	19
a512a22	97	*80*	9	82	129	45
a512b22	81	*65*	10	66	76	40
a513a26	100	*90*	10	90	129	75
a513b22	*123*	*123*	10	123	123	77
a612a21	72	*62*	15	72	74	99
a612b16	*49*	*49*	13	49	49	12
a712a20	*66*	*59*	16	59	87	52
a712b20	*55*	*54*	14	54	76	23

KEY

Optimal

STp Stochastic case with p perturbations

TS Tabu Search SA : Simulated Annealing

Figure 45: Summary of results on stochastic case with variable p.

There is clearly a considerable improvement in the solution value from the Tabu search case. Most of the problems took less than 1 minute of CPU time. We also presented the results as solved by the Simulated Annealing algorithm, clearly ST shows a considerable improvement of the other two approaches.

Conclusion and future work

- Tabu search generates reasonable solutions for the RLP. Careful selection of the parameters is essential in determining a good solution.
- Improved Tabu search has shown a good improvement over the general Tabu Search heuristic. Thus a perturbation process is a good strategy for improving the quality of the TS solutions.
- Stochastic algorithms show a good improvement over the TS deterministic case. Although the time spent on the stochastic case was higher than the TS algorithm, it was still reasonable as most of the problems were solved below 1 minute of CPU time.
- Real life problems normally do not involve a large planning horizon (e.g. 30 days of a month). However, they normally involve a large number of jobs (e.g. orders in a factory). A test for larger problem sets and specifically real life problems is therefore essential.

Chapter 6

HUMAN COMPUTER INTERACTION AND THE RLP

Introduction

In chapter 3 we saw how mathematical programming based exact approaches can be used to tackle the Resource Levelling Problem. However, this approach could only solve small sized problems. We have presented heuristic approaches in chapters 4 and 5, which can solve problems of larger size and provide good approximate solutions. However, real life problems can be so large that even heuristic procedures may have to spend a large amount of CPU time. In addition, the output may not concur with the social needs of the organisation concerned.

In manufacturing, the output of this process would be fed into a sophisticated editor to enable human schedulers to;

- Modify the work program to allow for last minute changes,
- Incorporate social constraints into the work profile.

In this chapter, we present a brief overview of how ideas in the theory of Human Computer Interaction (HCI) can be used to develop an interface for the RLP. We firstly present a general survey on the cognitive aspects of HCI that supports the idea of incorporating human capabilities into the problem-solving process. We then present our suggestions of a suitable interface for the RLP.

An overview of the literature

Various studies on the cognitive aspects of Human Computer Interaction have shown that human beings employ numerous problem-solving intuitive strategies that can sometimes outperform computers in terms of the quality of solution [Y89]. For instance, studies have shown that human beings employ, among others, a strategy known as 'Problem space reduction', a general technique of eliminating potential solutions so as to reduce the number of alternatives [G83]. As such it is usually regarded as a part of skilled human problem solving behaviour, and an aid to effective problem solution.

We are particularly interested in the human problem solving behaviour on problems that have been represented in a diagram form. Diagrams are important visual aids to reasoning because they provide readily interpretable information about problems to be solved. HCI Literature [BW97] suggests that a system to be used in support of problem solving process should provide such readily interpretable representations of problems.

Graphical displays

In applications ranging from statistical graphics to scientific visualisation, computers are used increasingly as tools for creating graphical displays of information. Most people have experienced the situation in which a good diagram can explain more than the prose one would use to describe it. A general observation is that perceptual codes in diagrams convey less information than symbolic ones. Symbolic representations elicit conscious cognitive processing, whereas analogic, pictorial ones are perceived more automatically [MJ90]. A map for example, provides specific locational information using grid co-ordinates, which for some people is more useful than analogic spatial arrangements.

Many studies have been done on the advantages of the use of graphical displays in mathematical problems. Douglas et al [D89] discuss the use of Graphical User Interfaces (GUI) on statistical graphs.

They introduce an empirical investigation of the basic features of a data graph using multivariate statistical techniques. William Cole [WC89] shows how the understanding of Bayesian reasoning can be enhanced via graphical displays. He concludes that graphical displays provide the 'mental model' that was missing in understanding Bayesian reasoning as pointed out by cognitive Psychologists.

One of the challenges in creating good object-oriented direct manipulation interfaces is in creating effective and appropriate graphics. William Verplank [WV88] introduces six challenges or principles of GUI design that emerged during the design project of the Xerox Star user interface

1. *Appropriate affect.* Try to capture all the subjective and emotional impact that different graphics can convey
2. *A match with the medium.* Create consistent quality graphics that is appropriate to the product and its market and make the most of the given medium
3. *Consistent graphic vocabulary*
4. *Visual order and user focus on the screen.* Contrast and animation when appropriately applied can draw users' attention to the most important features of the display
5. *The illusion of manipulable objects* and
6. *Revealed structure*

Principles 5 and 6 are considered to be new challenges because computers have now given us images that can be directly manipulated while they represent complex relationships, behaviours and structures that can be alternatively obscure or apparent. They are problems of dynamic, manipulable graphics with complex behaviours.

Halasz & Moran [HM82] and Rosenberg & Moran [RM84] have discussed the difficulties of creating appropriate user interface metaphors; the limit of the underlying metaphor should be obvious to the user. For example, the use of real names of the jobs such as 'frames' are more obvious to the user than names like 'job 1'. However for large problems, real names may be avoided in order to save space. Abbreviations should be used in this case; a consistent scheme should be adopted for developing them and a dictionary of abbreviations made available to the users [TT88].

In creating an illusion of manipulable objects, care must be taken. It should be clear that they can be selected and how to select them. It should be obvious when they are selected and that they will be the objects of the next action.

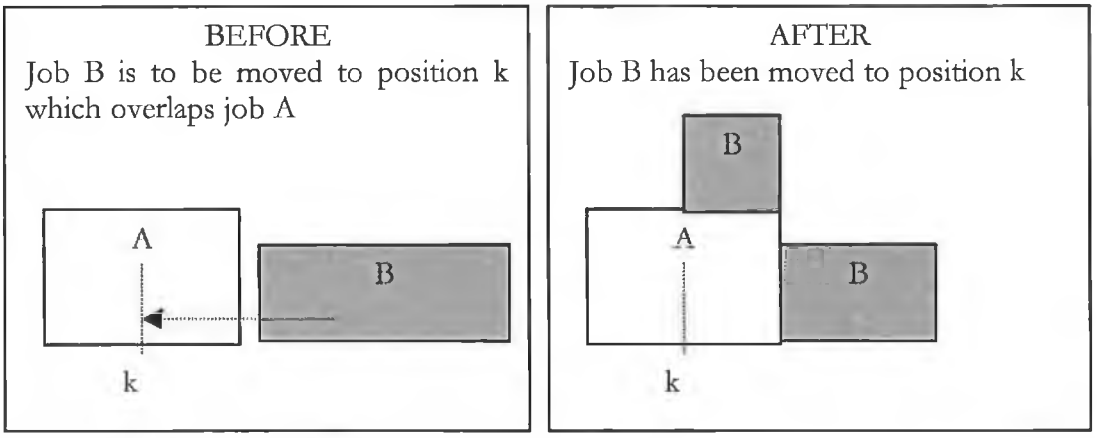
Since the RLP solution can be represented in a diagrammatic form (Resource Profile), we can employ the cognitive advantages of human beings and different display strategies to improve the performance of our heuristic solutions.

Suggestions on the design of the RLP interface

As we have seen on the literature review, many suggestions are available on how to design good user interfaces. Most of the literature is based on the authors' own experience on the interface design projects. We present our suggestion on how the interface for the RLP could be developed.

- Each job is represented by a set of rectangles, scaled according to its resource level (Height) and duration (Width).

- The shape of a job's diagram changes with the position in which it is placed i.e. fill 'holes' or climb 'hills' depending on the shape of the new position. See the example below for the movement of job A over B;



- Jobs from the same customer order are painted with the same colour, different from those of other orders. This will help to make the precedence relations visible, since precedence relations exist for jobs that belong to the same order. Any colour is influenced by its location, its placement and size and shape of the area it fills. Salomon [GS91] discusses on the effective use of colours in visualising large data sets. He assets that, the effectiveness of this use of colours relies more on human pattern recognition than on the ability to recognise specific colours.

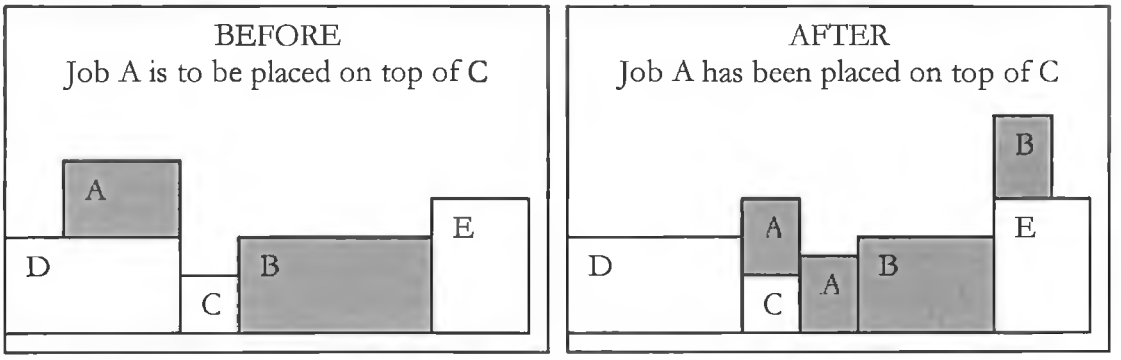
A certain degree of complexity is involved in the selection of colours to aid visualisation. In addition to the automatic selection of colours for each job, we also suggest the development of a customisation routine where users would be able to change colours to their own convenient patterns.

- Vertical scale (ruler) is important in visualising the height of the resource profile corresponding to any solution. Similarly a horizontal ruler should be maintained so as to make sure that no job is scheduled beyond the fixed planning horizon.

- We suggest the use of the mouse in moving objects (jobs) since it is much easier and natural for the user to be able to drag and drop an object. However, the mouse can move a job into a fractional position, in this case the new position should be selected to be the nearest integer. The use of keyboard cursors or function keys however can be programmed in such as way that a single move is always an integer. Both strategies may be used so as to give the user more flexibility and tools to work with.

- The RLP can be very large and it may not be possible to display all jobs into one single screen. Breaking the jobs into different windows may not be a very good option as visualisation of the whole resource profile is necessary for better understanding of the structure of the problem. The use of scroll bars is therefore a better suggestion where the user can be able to scroll through the whole resource profile in one piece. Zooming techniques are also important, the user should be able to view different portions of the profile as part of the problem solving strategy.

- Movement of a job involves a recursive movement of predecessor or successor jobs and may result into a major change in the resource profile. An example of jobs A, B, C, D and E illustrates the point;



Since jobs A and B belong to the same order, the precedence relations must be maintained. Thus if job A is moved in order to fill the 'hole' on top of C, then job B must be moved forward in order to maintain precedence relations. The resulting profile therefore shows worse height compared to the previous profile.

We suggest the use of the 'undo' command so as to be able to reverse actions that does not provide a good solution (if the user wants to do so).

- If a movement does not result in a feasible solution, two strategies may be allowed

    1. The job moves back to the original position and displays the error message on the status bar. This helps to make sure that the profile is always feasible during the problem solving process.

    2. Allow infeasible moves but keep a log file, which shows all jobs that are infeasible, and the reason for their infeasibilities. This will provide more freedom of movement and therefore enhance creativity on the part of the user.

We suggest the use of both strategies and give the user a choice to decide between the two.

- If the problem is very large, there is a possibility that the user may end up with a worse solution after a large number of trials and may need to recover the original profile. Instead of running the heuristic algorithm afresh, we may be able to preserve the original profile until the user is satisfied with the new changes. We suggest the use of two modes; Display and Edit. The initial profile is shown on the display mode where editing is not allowed. In order to be able to play around with the profile, the user should enter the edit mode and a new copy (new window) be created for this. The original window is therefore preserved on the display mode with the original profile.

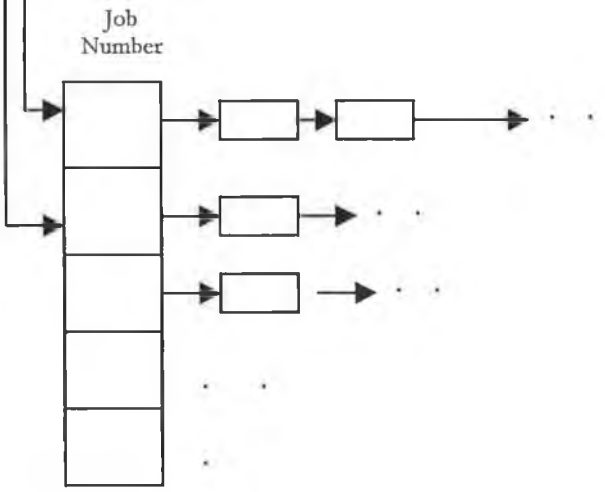
Data structures

As every move involves recursive movements of successors or predecessors we need to be able to keep all necessary data in an efficient way. We need to keep the successors and predecessors of each job, the current starting time, the resource level and the slack times of each job. A list of the data set is kept, where each list element contains all the necessary information about the job and a pointer to the adjacency list of successors. The predecessors of each job can be searched directly from the successor list.

Data List for each order

Job Number	Job Name	Resource Level	Duration	Start time	Float time	Free Float	Start time
.	.	.					

Successor List



In addition to these structures, we keep a list of resource levels at each point in the resource profile. Any move of a job must be checked for feasibility with these structures. If the move is feasible, the data structure has to be updated to reflect the change. All these structures were used in the previous algorithms and can be adapted to help in the graphics displays.

Concluding remark

We have suggested the use of visual display as an aid to improving the RLP output. Much has been said in the literature to support 'visual thinking' to creative problem solving. Although we have focused our attention on the objects' graphical displays, Tulli [TT88] suggest the use of what he calls 'ambidextrous thinking'. That is, using a variety of representations and transformations (graphical and linguistic, concrete and abstract, analogic and symbolic) and moving flexibly from one to the other. Given different representations describing the same thing, very different insights may be gained, different errors committed, different things remembered or forgotten.

Other representations of the RLP, such as a precedence network of jobs and tables of resource levels, duration and start time, are therefore necessary. The important goal is to provide not just a balance, but an appropriate span of alternative modalities.

Chapter 7

SUMMARY AND CONCLUSIONS

Work done so far

Exact methods

Four models were developed for the RLP and tested on small size problems that were randomly generated. We compared the performances of the developed models and present a summary of results.

- Model 1 consisted of both binary and continuos integer variables. Start times were presented by continuos integer variables. The resource level at each point on the planning horizon was represented by time-indexed binary variables. The model consisted of $n(T+1)+1$ variables and $T(2n+1)+2n+|H|$ constraints, where n is the number of jobs, T is the length of the planning horizon and |H| is the cardinality of the set of arcs forming the precedence network H.

- Model 2 is a purely binary time-indexed formulation. The resulting model has $n(T+1)$ variables and $n+|H|+T$ constraints, which is far less in size than model 1. The lower bound in the LP-relaxation in this case was not as good as the previous case.

- Model 3 was a result of the reformulation of model 2 by the replacement of the precedence constraints by a new set of inequalities, which is totally unimodular. The resulting model was larger than model 2; consisting of $n(T+1)$ variables and $T(|H|+1)+n$ constraints. However, the lower bound was better than rest of the developed models.

- Model 4 was developed without the inclusion of the time-indexed variables. It consists of continuos integer variables and binary variables which are not time-indexed. This resulted into a model with $3n^2+n+1$ variables and $3n^2+n+|H|$ constraints. The lower bound was very poor and the performance was consequently the worst.

Model 3 was found to be the best among the models developed. In general we found that time-indexed formulation approach provided better results. This is in concurrence with the findings of J, Sousa and L, Wolsey [SW92] on the single machine scheduling problems. Using this exact approach, we managed to solve to optimality problems of up to 30 jobs.

Heuristic methods

Three algorithms were developed.

Simulated annealing (SA):

We presented our approach to SA with a range of general and specific decisions on the set of parameters required by the algorithm. We classified our parameters into 9 cases and present a summary of results. Cases GVI (Geometric reduction function, Variable number of iterations, Increasing neighbourhood structure) and L1I (Lundy & Mees reduction function, 1 iteration at each temperature, Increasing neighbourhood structure) were found to provide better results than the rest of the tested cases. In addition to the selection of parameters, the success of the SA procedure was also found to depend on the structure of the problem. Density of jobs on the planning horizon (number of jobs per unit time of the planning horizon) and the amount of slack time in each job affected the performance. The amount of slack time for instance, determines the size of the searching space in which the SA algorithm needs to work on. With this approach, we managed to solve to optimality, problems of up 426 jobs.

Tabu Search (TS):

From the experience of the SA algorithm, it was clear that using modern heuristic techniques, we could solve problems with large size. However, it appears that this is the only study in this approach (we could not find any other work) and we therefore had no other work to compare the effectiveness of our algorithm. Thus, our interest in this case was more inclined towards the quality of the solution rather than on the size of the problem to be solved. We therefore generated 44 random small size problems (10 - 26 jobs) and solve to optimality by the MIP methods. We used the optimal solutions as a benchmark of the performance of our TS methods. As in the SA case, a major challenge was in the selection of parameters, which could provide us with good and effective solutions. The summary of results and analysis shows that TS is good approaches to RLP as our results were close the optimal values.

Perturbation algorithm (ST):

This approach was motivated by a study by G, Lesson and M, O'hEigeartaigh [GM93]. They successfully applied the perturbation algorithm to the crossing number problem. The algorithm works by performing random perturbations on the structure of the problem that is solved by a deterministic algorithm. We applied the TS procedure as a deterministic part of the ST. Although the algorithm took longer than TS, the improvement in performance was good. Half of the problems tested were solved to optimality and the remaining solutions were close to optimum values (see Fig 46). We also presented an analysis of the results and proposed a set of parameters to be used.

The perturbation approach showed a considerable amount of improvement over the rest of the models (see Fig 46 for comparisons). This study shows that, both SA, TS and ST are feasible and good approaches for the RLP.

Human Computer Interaction (HCI)

We presented a review on the literature to support the use of computer graphics to the RLP. This has been triggered by two major reasons;

1. In manufacturing, RLP can be so large that even powerful heuristic algorithms may require a very large amount of CPU time, which may not be acceptable by management. In this case, human beings can interrupt the process and finish up the remaining task. This has been supported by various studies on the cognitive aspects of HCI on the problem solving process (see Chapter 7). It is suggested that, given a well-designed interface, human beings can perform very well using their natural problem solving strategies.

2. In some cases, optimal or near optimal solutions presented by the computer algorithm may not concur with the social needs of the industry. In this case, human changes on the computer output are necessary.

We briefly presented a suggestion on how one could go about designing and implementing an interface for the RLP. We also suggested the possibility of using other representations apart from graphical profiles, which can enhance problem-solving process on the part of the user.

Future work

Exact approach

- Preprocessing and probing techniques as reviewed in chapter 2 can help to increase the size of problems that can be solved by the MIP models presented. Martin Savelsberg [MS93] provides a good framework for describing preprocessing and probing techniques for MIP and surveys some of the well-known basic techniques. He also extend his study to include some of the more recently developed techniques that are currently employed by the state-of-art general purpose mixed integer optimisers. George, Nemhauser, Martin Savelsberg and Gabriele Sigismondi presents one of these optimisers known as MINTO (Mixed Integer Optimiser, see [NSS94] and [SN96]). These techniques have been successfully applied to various MIP such as Knapsack, Lot Sizing, Node Packing, Linear Ordering and Cutting Stock problems (see [SN95]).

- Description of the RLP polytope by finding the associated facets is another approach that needs to be investigated. This approach has been applied to a number of combinatorial problems including Linear Ordering [R85], Travelling Salesman problems (see for instance [GP79a], [GP79b], [G80], [F91] and [CP80]). A branch and cut algorithm has been suggested as an efficient algorithmic approach to NP-Hard problems once the facets of the problem are known. M. Junger et al [JRT94] gives an introduction to cutting planes algorithms (branch and cut) with a list of successful practical applications in the literature.

- Development of other models or reformulation of the developed MIP models may provide a better lower bound and hence better performance.

Heuristic approach

- As we have observed in our experimentation, careful selection of parameters is very important in the effectiveness of both heuristics. More fine tuning of the presented parameters are bound to provide better results.

- It is well understood that, real life problems provide characteristics that are special to the problem and cannot be incorporated from the random generation of the problems. We decided to generate random problems because we could not get real life data during the time of the project. We therefore suggest that, more testing of the algorithms should be done using real life data. This might give us more insight into the behaviour of the RLP problem and hence better strategies for tackling the problem.

- Other modern heuristic techniques such as Genetic Algorithm and Neural Networks are worth investigating on their applicability to the RLP.

Human Computer Interaction

We have suggested a way in which an interface for the RLP could be developed. The design and development of this interface is another area of research.

```
C     *1st FORMULATION OF THE RESOURCE LEVELLING PROBLEM
C
OPTIONS NOREPORT
C
NOTATION
C
SUFFICES
C
 I     N     100
C
OPTION INTEGER
C
 J     M     100
C
OPTION INTEGER
C
 T     S     100
C
OPTION INTEGER
C
EXTERNAL VALUES
C
 R(I)     [1X,F5.1]
 P(I)     [1X,F5.1]
 H(I,J)   [1X,I3]
C
VARIABLES
C
 W      '*'
 X(I)   '*II'
 Y(I,T) '*II TT'
C
BOUND  BV
C
PROBLEM
MINIMISE
C
* OBJ   '***'
C
 W
C
SUBJECT TO
C
* RES   '**TT'
 SUM(I) R(I)*Y(I,T) - W .LE. 0.0
FOR ALL T
C
* BIN   '**II'
 SUM(T) Y(I,T) .EQ. P(I)
FOR ALL I
C
* ABV   '**II TT'
 T*Y(I,T) - X(I) .LE. (P(I)-1)
FOR ALL I,T
C
* BELW  '**II TT'
 X(I) + S*Y(I,T) .LE. (T+S)
FOR ALL I,T
C
* PRD  '**II JJ' NOT IF(H(I,J)  LE. SMALL)
  X(I1) - X(I) .GE. P(I)
FOR I1 .ST. (I1 .EQ J)
FOR ALL I,J
C
* BND    '**II'
  X(I) .LE. S - P(I)
FOR ALL I
C
ENDATA
```

Figure 46: MGG file for the first formulation (model 1) [S91]

```
TITLE  THE RESOURCE LEVELLING -DATA INPUT
*
*
MAXIMA  N = 10  M = 10  S = 42
*
EXTERNAL VALUES
*
R          0.0   1.0     7.0    1.0     7.0     4.0     11.0    4.0     14.0    0.0
*
P          0.0   11.0    7.0    16.0    12.0    11.0    13.0    15.0    10.0    0.0
*
*              1   2   3   4   5   6   7   8   9   10
H    1             1   1   1   1       1
H    2                                 1
H    3                                 1
H    4                                         1
H    5                                         1
H    6                                             1
H    7                                             1
H    8                                             1
H    9                                                 1
H    10
*
ENDATA
```

Figure 47: MG Input Data file for problem C302b.

```
C     *2nd FORMULATION OF THE RESOURCE LEVELLING PROBLEM
C
OPTIONS NOREPORT
C
C
NOTATION
C
SUFFICES
C
 I     N     100
C
OPTION INTEGER
C
 J     M     100
C
OPTION INTEGER
C
 T     S     100
C
OPTION INTEGER
C
C
EXTERNAL VALUES
C
 R(I)    [1X,F5.1]
 D(I)    [1X,F5.1]
 H(I,J)  [1X,I3]
C
C
C
VARIABLES
C
  W      '*'
C
  Y(I,T)  '*II TT'
C
BOUND BV
C
C
PROBLEM
MINIMISE
C
* OBJ   '***'
C
 W
C
C
SUBJECT TO
C
* BIN   '***II'
  SUM(T) Y(I,T) .EQ. 1.0
FOR ALL I
C
C
* PRSD  '**II JJ' NOT IF(H(I,J) .LE. SMALL)
  SUM(T) T*Y(I1,T)
  -SUM(T) T*Y(I,T) .GE. D(I)
FOR I1.ST. (I1.EQ.J)
FOR ALL I,J
C
C
* RESR  '***TT'
  SUM(I,T1) R(I)*Y(I,T1) - W .LE. 0.0
FOR T1.ST.(T1.GE.(T-D(I)+1) .AND. T1.LE.T)
FOR ALL T
C
C
C
ENDATA
```

Figure 48: MGG formulation for model 2

```
C     *3rd FORMULATION OF THE RESOURCE LEVELLING PROBLEM
C
OPTIONS NOREPORT
C
C
NOTATION
C
SUFFICES
C
 I     N     100
C
OPTION INTEGER
C
 J     M     100
C
OPTION INTEGER
C
 T     S     100
C
OPTION INTEGER
C
C
EXTERNAL VALUES
C
 R(I)    [1X,F5.1]
 D(I)    [1X,F5.1]
 H(I,J)  [1X,I3]
C
C
C
VARIABLES
C
  W     'w'
C
  Y(I,T)  '*II TT'
C
BOUND BV
C
C
PROBLEM
MINIMISE
C
* OBJ   '***'
C
  W
C
C
SUBJECT TO
C
* BIN   '***II'
  SUM(T) Y(I,T) .EQ. 1.0
FOR ALL I
C
* RESR  '**TT'
   SUM(I,T1) R(I)*Y(I,T1) - W .LE. 0.0
FOR T1.ST.((T1.GE.(T-D(I)+1)) .AND. (T1.LE.T))
C FOR ALL T
C
* CUT  '*IIJJTT' NOT IF(H(I,J).LE.SMALL)
  SUM(T1) Y(I,T1)
  - SUM(T2) Y(I1,T2) .GE. 0.0
FOR I1 .ST. (I1 .EQ. J)
FOR T1 .ST. (T1 .LE. (T-D(I)))
FOR T2 .ST. (T2 .LE. T)
C
C
C
ENDATA
```

Figure 49: MGG formulation for model 3

```
C     *4th FORMULATION OF THE RESOURCE LEVELLING PROBLEM
C
OPTIONS NOREPORT
NOTATION
C
SUFFICES
 I     N     100
C
OPTION INTEGER
C
 J     M     100
C
OPTION INTEGER
 T     S     100
C
OPTION INTEGER
C
EXTERNAL VALUES
C
 R(J)    [1X,F5.1]
 D(I)    [1X,F5.1]
 H(I,J)  [1X,I3]
C
VARIABLES
 W      '*'
 X(I)   '*II'
 Y(I,J)  '*II JJ'
C
BOUND  BV
C
 U(I,J)  '*II JJ'
C
BOUND  BV
C
 V(I,J)  '*II JJ'
C
BOUND  BV
C
PROBLEM
MINIMISE
* OBJ   '==>'
 W
C
SUBJECT TO
* PRD  '**II JJ' NOT IF(H(I,J)  LE. SMALL)
  X(I1) - X(I)  GE. D(I)
FOR I1 .ST. (I1 .EQ. J)
FOR ALL I,J
C
* RES    '**II'
  SUM(J) R(J)*Y(I,J) - W .LE. 0.0
FOR ALL I
C
* BELW   '**II JJ'
  X(I) - X(I1) - S*U(I,J) .LE. -1.0
FOR I1  ST. (I1 .EQ. I)
FOR ALL I,J
C
* ABV    '**II JJ'
  X(I1) - X(I) - S*V(I,J) .LE. -AB1
WHERE AB1=DUR(J)
FOR I1 .ST. (I1 .EQ. J)
FOR ALL I,J
C
* BIN  '**II JJ'
  U(I,J) + V(I,J) - Y(I,J) .LE. 1.0
FOR ALL I,J
C
FUNCTIONS
     FUNCTION DUR(K)
      DUR = D(K)
     RETURN
     END
ENDATA
```

Figure 50: MGG formulation for model 4.

Geometric Reduction (GVF) nrep += 0.05, *a* Value : 0.95, Fixed NbrHood: Max{10, m/100}								
Problem	Nodes per tree	# Orders	Total #Nodes	Profile Length	Starting Cost	Final Cost	Improvement in cost	Time (Sec.)
c3_02A	12	3	36	37	1159702	1094758	6%	3.00
c3_02B	9	3	27	45	80586	69538	14%	6.00
c4_02A	12	2	24	52	65010	62498	4%	6.00
c4_02B	15	2	30	66	154464	147746	4%	8.00
c5_02A	16	2	32	57	420012	375292	11%	9.00
c5_02B	27	3	81	97	3133218	3066546	2%	19.00
c6_02A	15	1	15	79	245799	243783	1%	9.00
c6_02B	12	3	36	77	189912	185656	2%	8.00
c7_02A	22	3	66	82	12569818	10896152	13%	19.00
c7_02B	26	3	78	70	3789161	2997507	21%	19.00
c3_03A	19	2	38	37	275556	265490	4%	6.00
c3_03B	19	3	57	40	682507	604823	11%	10.00
c4_03A	16	2	32	68	1305763	1191903	9%	13.00
c4_03B	21	1	21	54	124030	120990	2%	8.00
c5_03A	11	3	33	77	96345	96345	0%	8.00
c5_03B	36	2	72	82	10415471	9882711	5%	20.00
c6_03A	50	3	150	89	29102440	28162900	3%	30.00
c6_03B	43	3	129	93	16307697	13989709	14%	31.00
c7_03A	26	3	78	102	6963516	6486772	7%	15.00
c7_03B	64	2	128	105	11579453	8497259	27%	33.00
c2_04A	16	2	32	31	633273	540213	15%	9.00
c2_04B	11	2	22	44	84264	75576	10%	1.00
c3_04A	12	3	36	43	1779023	1679373	6%	8.00
c3_04B	18	3	54	45	4239064	3523748	17%	11.00
c4_04A	40	2	80	68	19465500	19465500	0%	292.00
c4_04B	77	2	144	66	81642620	59957131	27%	42.00
c5_04A	67	2	134	89	6998204	6865116	2%	36.00
c5_04B	76	1	76	85	1268567	1270773	0%	18.00
c6_04A	15	2	30	59	72040	72386	0%	6.00
c6_04B	178	2	356	100	62202010	38377430	38%	52.00
c7_04A	213	2	426	114	1.26E+08	1.26E+08	0%	108.00
c7_04B	190	2	380	124	39058380	36566440	6%	76.00
c3_05A	29	3	87	44	3508433	2850861	19%	12.00
c3_05B	37	3	111	58	4845554	4368358	10%	18.00
c4_05A	24	2	48	66	5999009	5163137	14%	8.00
c4_05B	35	3	105	81	20374440	17906042	12%	25.00
c5_05A	105	2	210	68	54630330	48388010	11%	45.00
c5_05B	96	1	96	88	35330710	31735210	10%	35.00
c3_06A	67	3	201	47	1.06E+08	95525760	10%	56.00
c3_06B	69	1	69	60	8557608	8450968	1%	25.00
c4_06A	68	1	68	84	20322800	18783480	8%	21.00
c4_06B	91	2	182	59	60404140	48842880	19%	43.00
c5_06A	34	1	34	90	441483	379329	14%	12.00
c5_06B	95	1	95	102	34268280	32469330	5%	30

Table 21: Summary of results for the GVF simulated annealing

Geometric Reduction (GVI)
nrep + = 0.05,
a Value : 0.95, Increasing NbrHood: Max{2.0, NbrHood += 0.15} <= m

Problem	Nodes per tree	# Orders	Total #Nodes	Profile Length	Starting Cost	Final Cost	Improvement in cost	Time (Sec.)
c3_02A	12	3	36	37	1159702	744884	36%	14.00
c3_02B	9	3	27	45	80586	67086	17%	9.00
c4_02A	12	2	24	52	65010	55142	15%	8.00
c4_02B	15	2	30	66	154464	152464	1%	11.00
c5_02A	16	2	32	57	420012	315340	25%	14.00
c5_02B	27	3	81	97	3133218	2402086	23%	30.00
c6_02A	15	1	15	79	245799	232659	5%	11.00
c6_02B	12	3	36	77	189912	161304	15%	12.00
c7_02A	22	3	66	82	12569818	9409486	25%	34.00
c7_02B	26	3	78	70	3789161	3034721	20%	25.00
c3_03A	19	2	38	37	275556	259038	6%	10.00
c3_03B	19	3	57	40	682507	599751	12%	19.00
c4_03A	16	2	32	68	1305763	1109231	15%	17.00
c4_03B	21	1	21	54	124030	105072	15%	9.00
c5_03A	11	3	33	77	96345	91445	5%	10.00
c5_03B	36	2	72	82	10415471	8728131	16%	31.00
c6_03A	50	3	150	89	29102440	21469670	26%	63.00
c6_03B	43	3	129	93	16307697	10869813	33%	49.00
c7_03A	26	3	78	102	6963516	5049284	27%	33.00
c7_03B	64	2	128	105	11579453	7494625	35%	54.00
c2_04A	16	2	32	31	633273	493847	22%	13.00
c2_04B	11	2	22	44	84264	64354	24%	8.00
c3_04A	12	3	36	43	1779023	1100355	38%	15.00
c3_04B	18	3	54	45	4239064	3445912	19%	23.00
c4_04A	40	2	80	68	19465500	14189841	27%	35.00
c4_04B	77	2	144	66	81642620	53114380	35%	68.00
c5_04A	67	2	134	89	6998204	4509264	36%	56.00
c5_04B	76	1	76	85	1268567	1030293	19%	26.00
c6_04A	15	2	30	59	72040	69112	4%	9.00
c6_04B	178	2	356	100	62202010	42339650	32%	148.00
c7_04A	213	2	426	114	1.26E+08	79355940	37%	200.00
c7_04B	190	2	380	124	39058380	23931840	39%	144.00
c3_05A	29	3	87	44	3508433	2588473	26%	27.00
c3_05B	37	3	111	58	4845554	4366546	10%	37.00
c4_05A	24	2	48	66	5999009	4116753	31%	22.00
c4_05B	35	3	105	81	20374440	15166405	26%	45.00
c5_05A	105	2	210	68	54630330	40148070	27%	89.00
c5_05B	96	1	96	88	35330710	19903120	44%	48.00
c3_06A	67	3	201	47	1.06E+08	78708560	26%	89.00
c3_06B	69	1	69	60	8557608	6941120	19%	34.00
c4_06A	68	1	68	84	20322800	12968694	36%	35.00
c4_06B	91	2	182	59	60404140	43537740	28%	79.00
c5_06A	34	1	34	90	441483	325577	26%	15.00
c5_06B	95	1	95	102	34268280	19240730	44%	49

Table 23: Summary of results for the GVI simulated annealing

Geometric Reduction: GFF
Fixed accepted moves:
a Value : 0.95, Fixed NbrHood: Max{10, m/100}

Problem	Nodes per tree	# Orders	Total #Nodes	Profile Length	Starting Cost	Final Cost	Improvement in cost	Time (Sec.)
c3_02A	12	3	36	37	1159702	943542	19%	2.00
c3_02B	9	3	27	45	80586	68880	15%	1.00
c4_02A	12	2	24	52	65010	64270	1%	44.00
c4_02B	15	2	30	66	154464	154464	0%	3.00
c5_02A	16	2	32	57	420012	372024	11%	5.00
c5_02B	27	3	81	97	3133218	3066546	2%	69.00
c6_02A	15	1	15	79	245799	232515	5%	13.00
c6_02B	12	3	36	77	189912	185656	2%	22.00
c7_02A	22	3	66	82	12569818	11305854	10%	3.00
c7_02B	26	3	78	70	3789161	3626143	4%	1.00
c3_03A	19	2	38	37	275556	265580	4%	24.00
c3_03B	19	3	57	40	682507	620453	9%	4.00
c4_03A	16	2	32	68	1305763	1191943	9%	2.00
c4_03B	21	1	21	54	124030	120990	2%	42.00
c5_03A	11	3	33	77	96345	96345	0%	5.00
c5_03B	36	2	72	82	10415471	9659899	7%	51.00
c6_03A	50	3	150	89	29102440	27717360	5%	157.00
c6_03B	43	3	129	93	16307697	14539705	11%	2.00
c7_03A	26	3	78	102	6963516	6424532	8%	12.00
c7_03B	64	2	128	105	11579453	10079257	13%	3.00
c2_04A	16	2	32	31	633273	595809	6%	8.00
c2_04B	11	2	22	44	84264	75838	10%	1.00
c3_04A	12	3	36	43	1779023	1458375	18%	52.00
c3_04B	18	3	54	45	4239064	3673474	13%	25.00
c4_04A	40	2	80	68	19465500	18572860	5%	88.00
c4_04B	77	2	144	66	81642620	65487670	20%	5.00
c5_04A	67	2	134	89	6998204	6865116	2%	114.00
c5_04B	76	1	76	85	1268567	1270773	0%	12.00
c6_04A	15	2	30	59	72040	64962	10%	3.00
c6_04B	178	2	356	100	62202010	60712100	2%	58.00
c7_04A	213	2	426	114	1.26E+08	1.24E+08	2%	5.00
c7_04B	190	2	380	124	39058380	38182420	2%	3.00
c3_05A	29	3	87	44	3508433	2929609	16%	3.00
c3_05B	37	3	111	58	4845554	4443734	8%	1.00
c4_05A	24	2	48	66	5999009	5103161	15%	1.00
c4_05B	35	3	105	81	20374440	18783510	8%	3.00
c5_05A	105	2	210	68	54630330	48410100	11%	2.00
c5_05B	96	1	96	88	35330710	30707150	13%	2.00
c3_06A	67	3	201	47	1.06E+08	98592070	7%	3.00
c3_06B	69	1	69	60	8557608	8450968	1%	29.00
c4_06A	68	1	68	84	20322800	18130660	11%	49.00
c4_06B	91	2	182	59	60404140	53120880	12%	2.00
c5_06A	34	1	34	90	441483	393387	11%	22.00
c5_06B	95	1	95	102	34268280	32821950	4%	2

Table 24: Summary of results for the GFF simulated annealing

Geometric Reduction: GFD
Fixed accepted moves:
a Value : 0.95, Decreasing NbrHood: Min{m, NbrHood -=0.15} >= 2.0

Problem	Nodes per tree	# Orders	Total #Nodes	Profile Length	Starting Cost	Final Cost	Improvement in cost	Time (Sec.)
c3_02A	12	3	36	37	1159702	909106	22%	1.00
c3_02B	9	3	27	45	80586	69964	13%	2.00
c4_02A	12	2	24	52	65010	61876	5%	5.00
c4_02B	15	2	30	66	154464	148006	4%	5.00
c5_02A	16	2	32	57	420012	359526	14%	4.00
c5_02B	27	3	81	97	3133218	2536646	19%	8.00
c6_02A	15	1	15	79	245799	223449	9%	77.00
c6_02B	12	3	36	77	189912	157624	17%	33.00
c7_02A	22	3	66	82	12569818	9408152	25%	52.00
c7_02B	26	3	78	70	3789161	3025143	20%	64.00
c3_03A	19	2	38	37	275556	262326	5%	4.00
c3_03B	19	3	57	40	682507	599391	12%	10.00
c4_03A	16	2	32	68	1305763	1109071	15%	14.00
c4_03B	21	1	21	54	124030	105622	15%	16.00
c5_03A	11	3	33	77	96345	91445	5%	10.00
c5_03B	36	2	72	82	10415471	8713163	16%	57.00
c6_03A	50	3	150	89	29102440	23490730	19%	10.00
c6_03B	43	3	129	93	16307697	11672203	28%	5.00
c7_03A	26	3	78	102	6963516	5181644	26%	14.00
c7_03B	64	2	128	105	11579453	8400679	27%	9.00
c2_04A	16	2	32	31	633273	493989	22%	8.00
c2_04B	11	2	22	44	84264	66140	22%	4.00
c3_04A	12	3	36	43	1779023	1086643	39%	10.00
c3_04B	18	3	54	45	2961511	2549043	14%	23.00
c4_04A	40	2	80	68	19465500	14416533	26%	6.00
c4_04B	77	2	144	66	81642620	53387200	35%	34.00
c5_04A	67	2	134	89	6998204	4890402	30%	7.00
c5_04B	76	1	76	85	1268567	1027715	19%	40.00
c6_04A	15	2	30	59	72040	62844	13%	2.00
c6_04B	178	2	356	100	62202010	42690350	31%	35.00
c7_04A	213	2	426	114	1.26E+08	99232740	21%	28.00
c7_04B	190	2	380	124	39058380	30173610	23%	22.00
c3_05A	29	3	87	44	3508433	2588149	26%	15.00
c3_05B	37	3	111	58	4845554	4371012	10%	17.00
c4_05A	24	2	48	66	5999009	4106439	32%	48.00
c4_05B	35	3	105	81	20374440	16151189	21%	23.00
c5_05A	105	2	210	68	54630330	39754310	27%	361.00
c5_05B	96	1	96	88	35330710	20340850	42%	17.00
c3_06A	67	3	201	47	1.06E+08	78718660	26%	45.00
c3_06B	69	1	69	60	8557608	7033248	18%	3.00
c4_06A	68	1	68	84	20322800	13067316	36%	8.00
c4_06B	91	2	182	59	60404140	44288320	27%	22.00
c5_06A	34	1	34	90	441483	343283	22%	4.00
c5_06B	95	1	95	102	34268280	21267630	38%	8

Table 25: Summary of results for the GFD simulated annealing

Geometric Reduction: GFI
Fixed accepted moves:
a Value : 0.95, Increasing NbrHood: Max{2.0, NbrHood += 0.15} <= m

Problem	Nodes per tree	# Orders	Total #Nodes	Profile Length	Starting Cost	Final Cost	Improvement in cost	Time (Sec.)
c3_02A	12	3	36	37	1159702	775244	33%	5.00
c3_02B	9	3	27	45	80586	67484	16%	1.00
c4_02A	12	2	24	52	65010	61662	5%	6.00
c4_02B	15	2	30	66	154464	147030	5%	6.00
c5_02A	16	2	32	57	420012	349802	17%	11.00
c5_02B	27	3	81	97	3133218	2913180	7%	2.00
c6_02A	15	1	15	79	245799	232623	5%	7.00
c6_02B	12	3	36	77	189912	169076	11%	14.00
c7_02A	22	3	66	82	12569818	10384544	17%	6.00
c7_02B	26	3	78	70	3789161	3170047	16%	14.00
c3_03A	19	2	38	37	275556	254514	8%	3.00
c3_03B	19	3	57	40	682507	599819	12%	7.00
c4_03A	16	2	32	68	1305763	1191903	9%	15.00
c4_03B	21	1	21	54	124030	107446	13%	8.00
c5_03A	11	3	33	77	96345	96345	0%	4.00
c5_03B	36	2	72	82	10415471	9041915	13%	9.00
c6_03A	50	3	150	89	29102440	26598610	9%	8.00
c6_03B	43	3	129	93	16307697	13341417	18%	3.00
c7_03A	26	3	78	102	6963516	6054988	13%	38.00
c7_03B	64	2	128	105	11579453	9102361	21%	5.00
c2_04A	16	2	32	31	633273	496781	22%	5.00
c2_04B	11	2	22	44	84264	75436	10%	1.00
c3_04A	12	3	36	43	1779023	1303989	27%	12.00
c3_04B	18	3	54	45	2961511	2549043	14%	11.00
c4_04A	40	2	80	68	19465500	16544907	15%	3.00
c4_04B	77	2	144	66	81642620	55531580	32%	7.00
c5_04A	67	2	134	89	6998204	5799554	17%	4.00
c5_04B	76	1	76	85	1268567	1153777	9%	4.00
c6_04A	15	2	30	59	72040	72040	0%	2.00
c6_04B	178	2	356	100	62202010	50027870	20%	17.00
c7_04A	213	2	426	114	1.26E+08	1.11E+08	12%	20.00
c7_04B	190	2	380	124	39058380	35760720	8%	16.00
c3_05A	29	3	87	44	3508433	2598869	26%	3.00
c3_05B	37	3	111	58	4845554	4366926	10%	11.00
c4_05A	24	2	48	66	5999009	4264601	29%	2.00
c4_05B	35	3	105	81	20374440	16742123	18%	63.00
c5_05A	105	2	210	68	54630330	40741560	25%	17.00
c5_05B	96	1	96	88	35330710	34998370	1%	15.00
c3_06A	67	3	201	47	1.06E+08	83030900	21%	6.00
c3_06B	69	1	69	60	8557608	7152192	16%	3.00
c4_06A	68	1	68	84	20322800	14693850	28%	3.00
c4_06B	91	2	182	59	60404140	47243700	22%	18.00
c5_06A	34	1	34	90	441483	368887	16%	3.00
c5_06B	95	1	95	102	34268280	26934260	21%	6

Table 26: Summary of results for the GFI simulated annealing

Lundy & Mees: L1F
One move:
a Value : 0.95, Fixed NbrHood: Max{10, m/100}<= m

Problem	Nodes per tree	# Orders	Total #Nodes	Profile Length	Starting Cost	Final Cost	Improvement in cost	Time (Sec.)
c3_02A	12	3	36	37	1159702	971726	16%	5.00
c3_02B	9	3	27	45	80586	70224	13%	4.00
c4_02A	12	2	24	52	65010	62328	4%	5.00
c4_02B	15	2	30	66	154464	147552	4%	7.00
c5_02A	16	2	32	57	420012	364382	13%	9.00
c5_02B	27	3	81	97	3133218	3066546	2%	14.00
c6_02A	15	1	15	79	245799	240291	2%	8.00
c6_02B	12	3	36	77	189912	185656	2%	8.00
c7_02A	22	3	66	82	12569818	11032565	12%	11.00
c7_02B	26	3	78	70	3789161	3306939	13%	9.00
c3_03A	19	2	38	37	275556	275490	0%	5.00
c3_03B	19	3	57	40	682507	624709	8%	6.00
c4_03A	16	2	32	68	1305763	1191903	9%	11.00
c4_03B	21	1	21	54	124030	120990	2%	7.00
c5_03A	11	3	33	77	96345	96345	0%	7.00
c5_03B	36	2	72	82	10415471	9519409	9%	15.00
c6_03A	50	3	150	89	29102440	27685360	5%	21.00
c6_03B	43	3	129	93	16307697	14574261	11%	24.00
c7_03A	26	3	78	102	6963516	6380140	8%	10.00
c7_03B	64	2	128	105	11579453	8489937	27%	20.00
c2_04A	16	2	32	31	633273	540233	15%	8.00
c2_04B	11	2	22	44	84264	76510	9%	4.00
c3_04A	12	3	36	43	1779023	1521671	14%	6.00
c3_04B	18	3	54	45	2961511	2549043	14%	6.00
c4_04A	40	2	80	68	19465500	19080120	2%	17.00
c4_04B	77	2	144	66	81642620	53140160	35%	24.00
c5_04A	67	2	134	89	6998204	6852414	2%	27.00
c5_04B	76	1	76	85	1268567	1248573	2%	15.00
c6_04A	15	2	30	59	72040	66980	7%	6.00
c6_04B	178	2	356	100	62202010	56807590	9%	40.00
c7_04A	213	2	426	114	1.26E+08	1.21E+08	4%	61.00
c7_04B	190	2	380	124	39058380	37680810	4%	49.00
c3_05A	29	3	87	44	3508433	3037031	13%	6.00
c3_05B	37	3	111	58	4845554	4610508	5%	12.00
c4_05A	24	2	48	66	5999009	4857961	19%	8.00
c4_05B	35	3	105	81	20374440	18704850	8%	16.00
c5_05A	105	2	210	68	54630330	47698830	13%	32.00
c5_05B	96	1	96	88	35330710	31472390	11%	23.00
c3_06A	67	3	201	47	1.06E+08	94355460	11%	37.00
c3_06B	69	1	69	60	8557608	8450968	1%	19.00
c4_06A	68	1	68	84	20322800	17815520	12%	15.00
c4_06B	91	2	182	59	60404140	49101380	19%	31.00
c5_06A	34	1	34	90	441483	390587	12%	10.00
c5_06B	95	1	95	102	34268280	32415980	5%	20.00

Table 27: Summary of results for the L1F simulated annealing

Lundy & Mees: L1D
One move:
a Value : 0.95, Decreasing NbrHood: Min{m, NbrHood -=0.15} >= 2.0

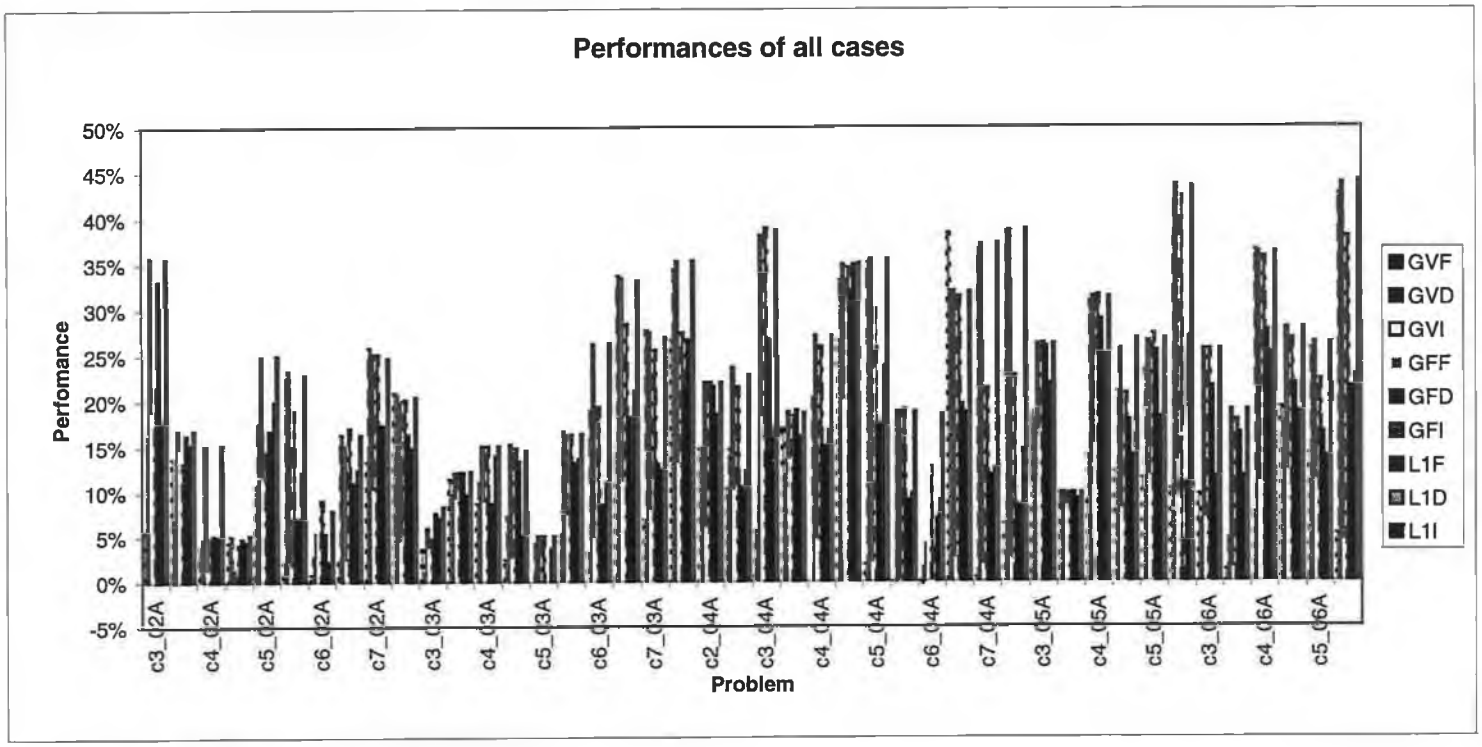
Problem	Nodes per tree	# Orders	Total #Nodes	Profile Length	Starting Cost	Final Cost	Improvement in cost	Time (Sec.)
c3_02A	12	3	36	37	1159702	955890	18%	5.00
c3_02B	9	3	27	45	80586	68286	15%	5.00
c4_02A	12	2	24	52	65010	61724	5%	5.00
c4_02B	15	2	30	66	154464	152464	1%	7.00
c5_02A	16	2	32	57	420012	336306	20%	6.00
c5_02B	27	3	81	97	3133218	2753664	12%	12.00
c6_02A	15	1	15	79	245799	240669	2%	7.00
c6_02B	12	3	36	77	189912	166292	12%	7.00
c7_02A	22	3	66	82	12569818	10394870	17%	9.00
c7_02B	26	3	78	70	3789161	3226415	15%	8.00
c3_03A	19	2	38	37	275556	255878	7%	5.00
c3_03B	19	3	57	40	682507	616353	10%	4.00
c4_03A	16	2	32	68	1305763	1123025	14%	9.00
c4_03B	21	1	21	54	124030	117704	5%	6.00
c5_03A	11	3	33	77	96345	92817	4%	8.00
c5_03B	36	2	72	82	10415471	8995555	14%	10.00
c6_03A	50	3	150	89	29102440	25875380	11%	16.00
c6_03B	43	3	129	93	16307697	12870617	21%	16.00
c7_03A	26	3	78	102	6963516	6099548	12%	11.00
c7_03B	64	2	128	105	11579453	8971289	23%	17.00
c2_04A	16	2	32	31	633273	515965	19%	4.00
c2_04B	11	2	22	44	84264	73900	12%	5.00
c3_04A	12	3	36	43	1779023	1498337	16%	5.00
c3_04B	18	3	54	45	2961511	2549043	14%	7.00
c4_04A	40	2	80	68	19465500	16216983	17%	11.00
c4_04B	77	2	144	66	81642620	59414020	27%	12.00
c5_04A	67	2	134	89	6998204	5331760	24%	16.00
c5_04B	76	1	76	85	1268567	1145025	10%	11.00
c6_04A	15	2	30	59	72040	65488	9%	6.00
c6_04B	178	2	356	100	62202010	50600110	19%	29.00
c7_04A	213	2	426	114	1.26E+08	1.1E+08	12%	45.00
c7_04B	190	2	380	124	39058380	33358700	15%	34.00
c3_05A	29	3	87	44	3508433	2742163	22%	5.00
c3_05B	37	3	111	58	4845554	4398038	9%	9.00
c4_05A	24	2	48	66	5999009	4500491	25%	9.00
c4_05B	35	3	105	81	20374440	17521380	14%	11.00
c5_05A	105	2	210	68	54630330	44712960	18%	16.00
c5_05B	96	1	96	88	35330710	25789700	27%	14.00
c3_06A	67	3	201	47	1.06E+08	93521920	12%	18.00
c3_06B	69	1	69	60	8557608	7557120	12%	10.00
c4_06A	68	1	68	84	20322800	15215562	25%	11.00
c4_06B	91	2	182	59	60404140	51465840	15%	15.00
c5_06A	34	1	34	90	441483	380517	14%	9.00
c5_06B	95	1	95	102	34268280	26490540	23%	14.00

Table 28: Summary of results for the L1D simulated annealing

Lundy & Mees: L1I
Fixed accepted moves:
a Value : 0.95, Increasing NbrHood: Max{2.0, NbrHood += 0.15} <= m

Problem	Nodes per tree	# Orders	Total #Nodes	Profile Length	Starting Cost	Final Cost	Improvement in cost	Time (Sec.)
c3_02A	12	3	36	37	1159702	746453	36%	13.00
c3_02B	9	3	27	45	80586	67042	17%	8.00
c4_02A	12	2	24	52	65010	55098	15%	9.00
c4_02B	15	2	30	66	154464	146342	5%	11.00
c5_02A	16	2	32	57	420012	315046	25%	13.00
c5_02B	27	3	81	97	3133218	2417778	23%	27.00
c6_02A	15	1	15	79	245799	226359	8%	10.00
c6_02B	12	3	36	77	189912	158878	16%	12.00
c7_02A	22	3	66	82	12569818	9464762	25%	27.00
c7_02B	26	3	78	70	3789161	3015663	20%	21.00
c3_03A	19	2	38	37	275556	252552	8%	9.00
c3_03B	19	3	57	40	682507	598595	12%	17.00
c4_03A	16	2	32	68	1305763	1109035	15%	14.00
c4_03B	21	1	21	54	124030	105884	15%	9.00
c5_03A	11	3	33	77	96345	91445	5%	11.00
c5_03B	36	2	72	82	10415471	8708953	16%	26.00
c6_03A	50	3	150	89	29102440	21451580	26%	48.00
c6_03B	43	3	129	93	16307697	10902619	33%	39.00
c7_03A	26	3	78	102	6963516	5086676	27%	29.00
c7_03B	64	2	128	105	11579453	7487063	35%	44.00
c2_04A	16	2	32	31	633273	493735	22%	11.00
c2_04B	11	2	22	44	84264	65034	23%	8.00
c3_04A	12	3	36	43	1779023	1091855	39%	13.00
c3_04B	18	3	54	45	2961511	2549043	14%	19.00
c4_04A	40	2	80	68	19465500	14189877	27%	27.00
c4_04B	77	2	144	66	81642620	53030510	35%	47.00
c5_04A	67	2	134	89	6998204	4513248	36%	45.00
c5_04B	76	1	76	85	1268567	1030663	19%	22.00
c6_04A	15	2	30	59	72040	58752	18%	9.00
c6_04B	178	2	356	100	62202010	42377230	32%	107.00
c7_04A	213	2	426	114	1.26E+08	79077420	37%	138.00
c7_04B	190	2	380	124	39058380	23889850	39%	107.00
c3_05A	29	3	87	44	3508433	2589119	26%	23.00
c3_05B	37	3	111	58	4845554	4367028	10%	31.00
c4_05A	24	2	48	66	5999009	4121237	31%	17.00
c4_05B	35	3	105	81	20374440	14940109	27%	34.00
c5_05A	105	2	210	68	54630330	40028580	27%	65.00
c5_05B	96	1	96	88	35330710	19971790	43%	35.00
c3_06A	67	3	201	47	1.06E+08	78713870	26%	61.00
c3_06B	69	1	69	60	8557608	6940216	19%	25.00
c4_06A	68	1	68	84	20322800	12942018	36%	26.00
c4_06B	91	2	182	59	60404140	43535940	28%	55.00
c5_06A	34	1	34	90	441483	325481	26%	14.00
c5_06B	95	1	95	102	34268280	19189140	44%	36.00

Table 29: Summary of results for the L1I simulated annealing


Performances of all cases
50%
45%
40%
35%
30%
25%
20%
15%
10%
5%
0%
-5%
Perfomance
c3_02A
c4_02A
c5_02A
c6_02A
c7_02A
c3_03A
c4_03A
c5_03A
c6_03A
c7_03A
c2_04A
c3_04A
c4_04A
c5_04A
c6_04A
c7_04A
c3_05A
c4_05A
c5_05A
c3_06A
c4_06A
c5_06A
Problem
GVF
GVD
GVI
GFF
GFD
GFI
L1F
L1D
L1I

Summary of results for Problem a414b22. The steps involved in each perturbation case.

Problem a414b22 Two jobs Pertubation
Planning Horizon =35 Mean = 85.61905 Stdev = 8.517489

step	H
Initial (0)	83
1	94
2	94
3	94
4	94
5	75
6	76
7	89
8	76
9	76
10	102
11	77
12	90
13	77
14	77
15	87
16	85
17	77
18	90
19	98
20	87

Optimal = 71
Best H = 75

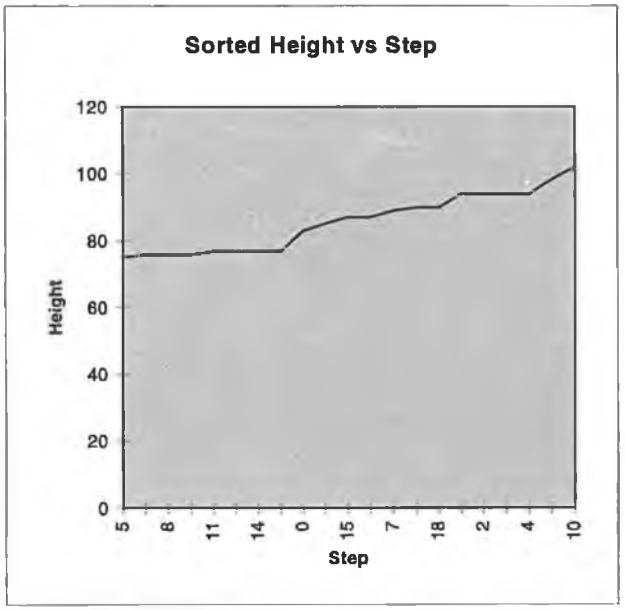


Figure 51: Two-job perturbation.

Three jobs Pertubation

Mean = 80.38095 stdev = 5.817871

step	
Initial (0)	83
1	83
2	76
3	92
4	86
5	96
6	83
7	77
8	77
9	77
10	77
11	76
12	76
13	87
14	77
15	77
16	83
17	77
18	76
19	76
20	76

Optimal = 71
Best H = 76

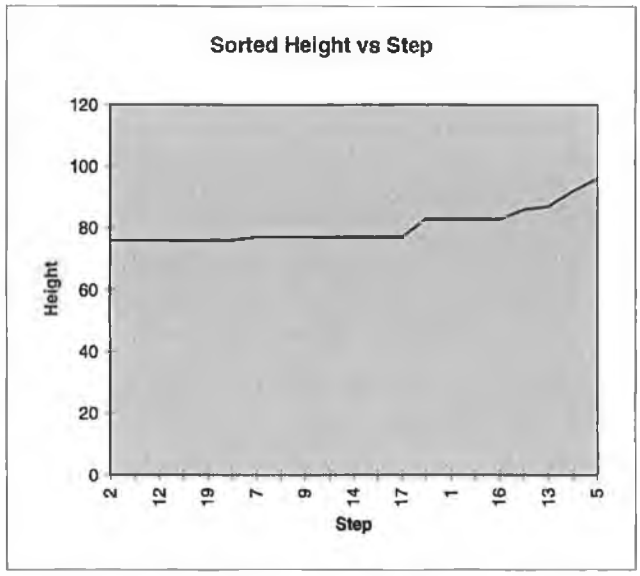


Figure 52: Three-job perturbation.

Five jobs Pertubation

Mean = 82.80952 stdev = 7.054212

Step	H	% Improv.
Initial (0)	83	
1	78	
2	76	
3	96	
4	82	
5	77	
6	76	
7	83	
8	85	
9	77	
10	83	
11	77	
12	85	
13	90	
14	76	
15	80	
16	94	
17	90	
18	77	
19	98	
20	76	
Best Height		76
Optimal Height		71

Step	Sorted H

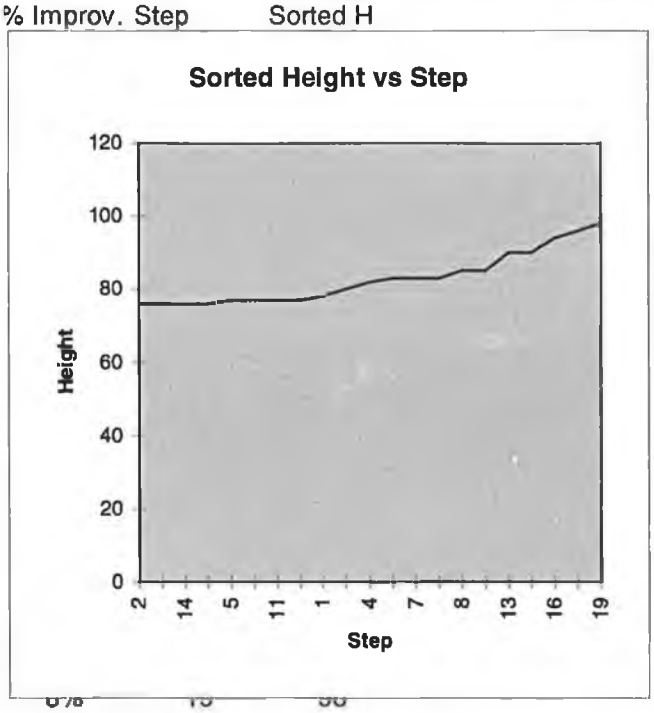


Figure 53: Five-job perturbations

Six jobs Pertubation		Mean =	87.47619	stdev =	11.64311
step	H	% Improv.	Step	Sorted H	
Initial (0)	83				
1	94				
2	94				
3	94				
4	94				
5	83				
6	80				
7	84				
8	126				
9	101				
10	89				
11	77				
12	77				
13	76				
14	75				
15	90				
16	83				
17	87				
18	80				
19	94				
20	76				
Best Height		75			
Optimal Height		71			

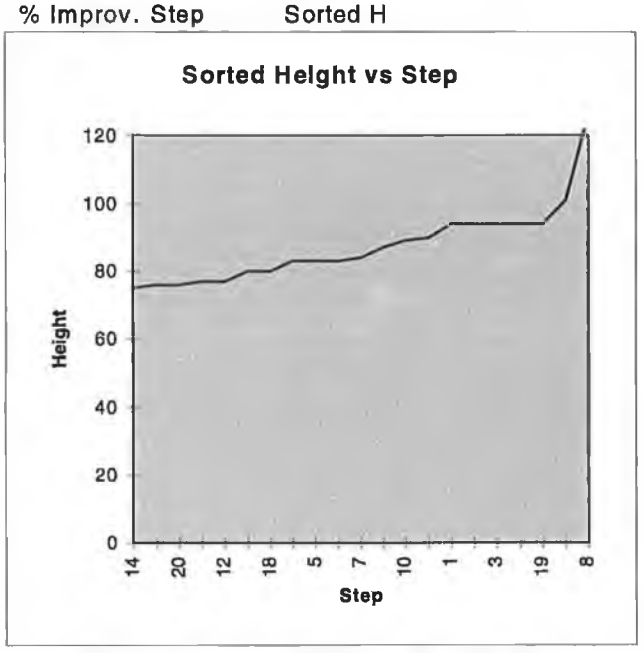


Figure 54: Six-job perturbations.

Seven jobs Pertubation		Mean =	79.90476	stdev =	4.265029
Step	H	% Improv.	Step	Sorted H	
Initial (0)	83				
1	83				
2	76				
3	77				
4	77				
5	77				
6	79				
7	77				
8	77				
9	83				
10	83				
11	94				
12	80				
13	84				
14	77				
15	77				
16	83				
17	80				
18	77				
19	77				
20	77				
Best Height		76			
Optimal Height		71			

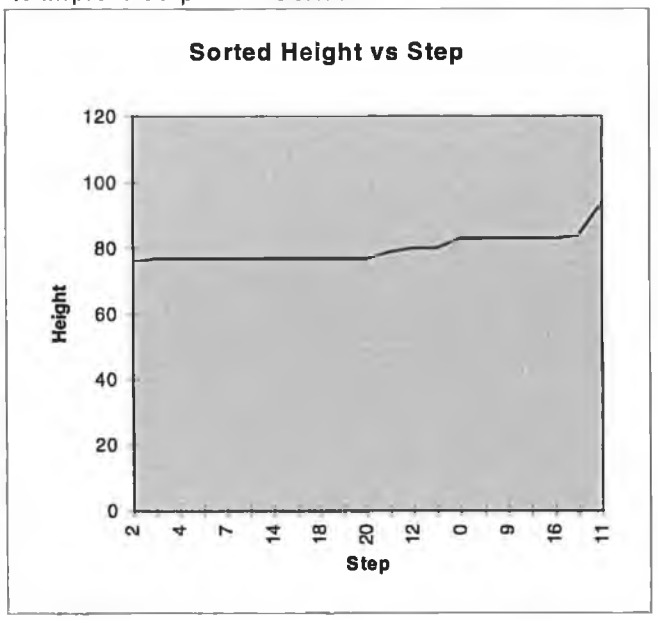


Figure 55: Seven-job perturbations

Eight jobs Pertubation Mean = 84.71429 stdev = 6.827466

Iterations	H	% Improv.	Steps	Sorted H
Initial (0)	83			
1	94			
2	90			
3	80			
4	88			
5	88			
6	94			
7	76			
8	77			
9	79			
10	92			
11	79			
12	77			
13	88			
14	77			
15	82			
16	98			
17	88			
18	90			
19	83			
20	76			
Best Height		76		
Optimal Height		71		

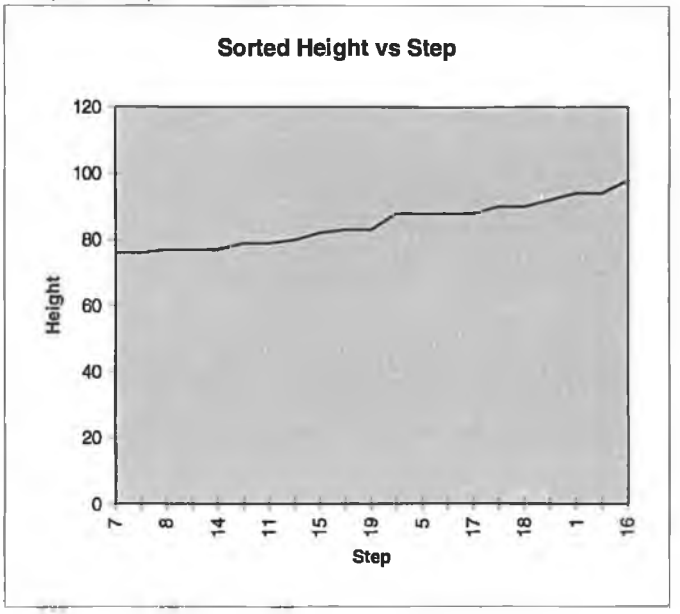


Figure 56: Eight-job perturbations

Nine jobs Pertubation Mean = 85.19048 stdev = 8.133997

Iterations	H	% Improv.	Step	Sorted H
Initial (0)	83			
1	94			
2	94			
3	77			
4	94			
5	83			
6	94			
7	75			
8	98			
9	77			
10	77			
11	88			
12	79			
13	79			
14	87			
15	77			
16	94			
17	87			
18	98			
19	77			
20	77			
Best Height		75		
Optimal Height		71		

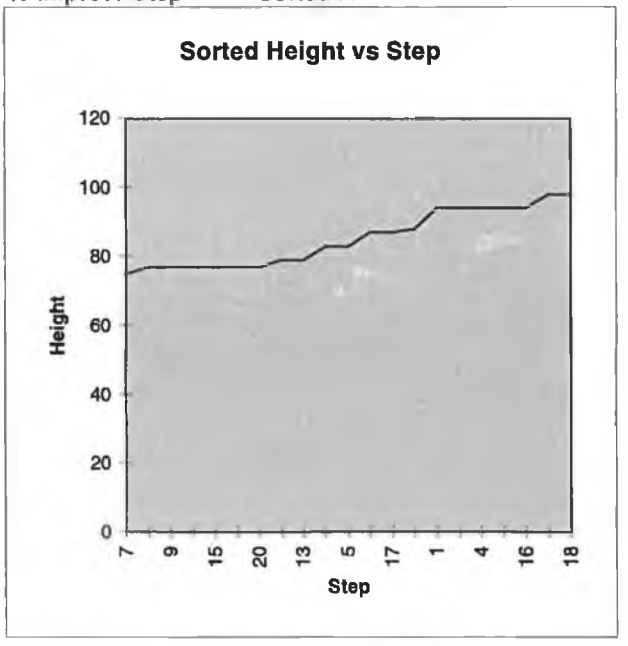


Figure 57: Nine-job perturbations.

Ten jobs Pertubation			Mean =	83.57143 stdev =	7.820303
Iterations	H	% Improv.	Step	Sorted H	
Initial (0)	83				
1	73				
2	77				
3	94				
4	89				
5	82				
6	76				
7	90				
8	85				
9	83				
10	76				
11	90				
12	97				
13	94				
14	77				
15	98				
16	76				
17	77				
18	77				
19	84				
20	77				
Best Height			73		
Optimal Height			71		

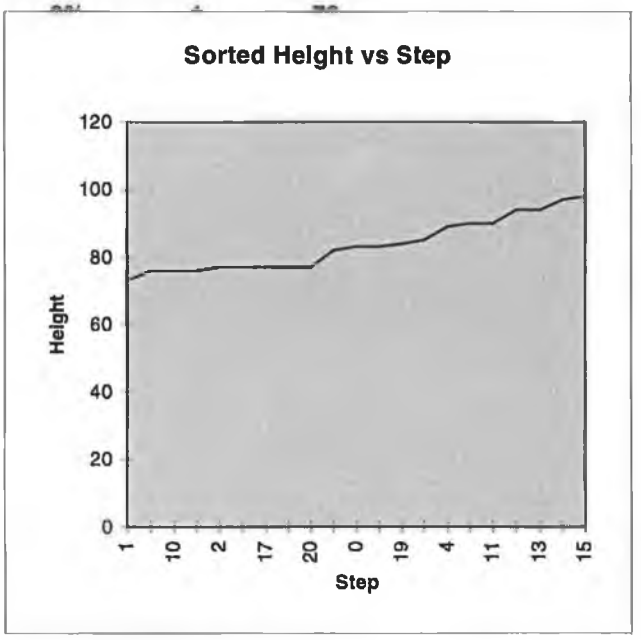


Figure 58: Ten-job perturbations.

[R93] Roger, G. S (1993): "*Operations Management, Decision Making in the operations function*", McGraw - Hill, Inc. New York.

[J89] John, H. B (1989) : "*Capacity Management*", South - Western Publishing Co., Cincinnati, West Chicago.

[W64] L. de Witte (1964): *"Manpower Levelling of PERT Networks"*, Data Processing for Science/Engineering, March/April, 29-38.

[D66] E. Davis (1966): *"Resource Allocation in Project Network Models - A Survey"*, Journal of Industrial Engineering, vol 17, 177-188.

[D73] E, Davis (1973*): "Project Scheduling under Resource Constraints - Historical Review and Categorisation of Procedures"*, AIIE Transactions, Vol. 5, 297-313.

[P89] Pulleyblank, W.R (1989) : *"Polyhedral Combinatorics"*, in Nemhauser,G.L, Rinooy,K and Todd,M.J (ed): Handbooks in Operations Research and Management Science 1 : *"Optimisation"*, North-Holland Publishing Co. Amsterdam - New York - Oxford - Tokyo, 371 - 440 .

[R85] Reinelt, G (1985) : *"The linear Ordering Problem : Algorithms and Applications"*, Herman Verlag Berlin.

[S86] Schrijver, A (1986) : *"Theory of Linear and Integer Programming"*, John Wiley & Sons, Chisester - New York - Brisbane - Toronto - Singapore.

[BG82] Bachem, A and Grötschel, M (1982): *"New aspects of Polyhedral theory"* in Korte, B (ed): Modern Applied Mathematics *"Optimisation and Operations Research"*, North-Holland Publishing Co. Amsterdam - New York - Oxford, 1982, 51 - 106.

[T75] Taha, H. A (1975): "*Integer Programming: Theory, Applications and Computations*", Academic Press, New-York.

[T89] Taha, A.H (1989) : *"Operations Research, an Introduction"*, MacMillan Publishing Co. New York.

[C73] Chvatal, V (1973) : *"Edmond's Polytope and a hierarchy of combinatorial problems"*, Discrete Mathematics 4 (1973), 305 - 337.

[N84] Nemhauser,G.L and Wolsey,L.A (1984) *: "A recursive procedure for generating all mixed integer cuts"*, CORE, Cornel University, New York.

[S80] Schrijver, A (1980): *"On Cutting Planes "*, Annals of Discrete Mathematics 9 (1980), 291 - 296. North - Holland Publishing Co.

[C82] Crowder,H, Johnson,E.L and Padberg, M (1982): *"Solving Large-Scale Zero-One Linear Programming Problems"*, Operations Research 31, No. 5 (1983), 803 - 834.

[H90] Hoffman, K.L and Padberg,M (1990): "*Improving LP-Representations of zero-one Linear Programs for Branch-and-Cut*", ORSA Journal on Computing 3, no.2 (1991), 121 - 134.

[W78] Williams, H.P (1978) : *"Model building in Mathematical Programming"*, John Wiley & Sons, Chisester - New York - Brisbane - Toronto - Singapore

[AM96] A. Mushi & M. O'hEigeartaigh (1996): "*A Simulated Annealing Procedure for the Resource Levelling Problem*". Dublin City University, School of Computer Applications. Working Paper: CA-0496.

[S91] SCICON Ltd (1991): *"MGG User Guide. Version 3.1"*, London.

[S93] SCICON Ltd (1993): "*SCICONIC User Guide. Version 2.30*", London.

[CA87] Christofides et al, (1987):"*Project Scheduling with Resource Constraints: A Branch and Bound approach*". European Journal of Operations Research 29, 262-273. North-Holland.

[W85] Williams, H. P. (1985): "*Model building in Mathematical Programming*", John Wiley & Sons, New York.

[I93] Ingber, A L. (1993): *"Simulated Annealing: Practice versus Theory"*, Journal of Mathl. Comput. Modelling, V18, No. 11, 29-57

[HB91] Hanan L.& Bruce, M. (1991) : "*Composite Stock Cutting through Simulated Annealing*", University of Missouri at Rolla, Research Report Numbers CSC 91-09 & ISC 91-04.

[C93] Colin R. Reeves(Ed.) (1993) : "*Modern Heuristic Techniques for Combinatorial Problems*", Blackwell Scientific Publications, Oxford.

[LM86] M. Lundy & A. Mees (1986): "*Convergence of an Annealing Algorithm*". Math. Prog. , 34, 111-124.

[H92] Hamdy, Taha. A (1992) : "*Operations Research, An Introduction*", MacMillan Publishing Company, New York.

[BK62] Burgess, A & Killebrew, J (1962) : "*Variation in Activity Level on a Cyclical Arrow Diagram*", The Journal of Industrial Engineering, vol. xiii, No. 2.

[KC95] K. Aardal & C. van Hoesel (1995):*"Polyhedral Techniques in Combinatorial Optimisation",* Memorandum COSOR 95-15, Eindhoven University of Technology, The Netherlands.

[AM96b] Mushi, A & O'Heigeartaigh, M (1996): *"The resource Levelling Problem (RLP). Mixed Integer Programming Formulations"*, School of Computer Application, Dublin City University, Working Paper : CA-2296.

[D94] Dash Associates (1994): "*XPRESS-MP user guide*", Dash Associates Limited. Release 8.

[GM93] Gary Leeson & Micheal O'hEigeartaigh (1993): *"The Crossing Number Problem: A Comparative Study of Simulated Annealing, TABU Search and MYOPIC Algorithms"*. School of Computer Applications, Dublin City University, Working Paper: CA-3193.

[Y89] Yvonne Waern (1989): *"Cognitive Aspects of Computer Supported Tasks"*, John Wiley & Sons Ltd, New-York.

[G83] T.Green, S. Payne & G. Van der Veer (1983): *"The Psychology of Computer use"* in B. R. Gaines (Ed.) *"Computer and People Series"*, Academic Press Inc. London.

[BW97] Bruce, W & Gersh, J (1997): *"User Modelling and Information Presentation for Naval Tactical Picture Agents"*, The John Hopkins University, Woking paper in Applied Physics Laboratory, USA

[MJ90] Mountford, S, J (1991): *"Tools and techniques for creative design"*, in Laurel, B (Ed.) (1991): *"The Art of Human Computer Interface Design"*, Addison-Wesley Publishing Company, Inc. Massachusetts.

[D89] D. Gillan, R. Lewis & M. Rudisil (1989): *"Modes of User Interactions with graphical interfaces. I. Statistical graphs"*, CHI'89 Conference proceedings on Human Factors in Computing Systems. ACM Press, Texas.

[WC89] W. Cole (1989*): "Understanding Bayesian Reasoning via Graphical Displays"*, CHI'89 Proceedings of the conference on Human Factors in Computing Systems. ACM Press, Texas.

[WV88] W. Verplank (1988): *"Graphic Challenges in designing object-oriented user interfaces"* in M, Helander (Ed.) (1988): *"Handbook of Human Computer Interaction"*, Elsevier Science Publishers, B.V (North-Holland).

[HM82] Halasz, F & Moran T. (1982): *"Analogy considered harmful"*. In the Proceedings of the Human Factors in Computer Systems Conference. Gaithersburg: ACM.

[RM84] Rosenberg, J. K. & Moran T. (1984): *"Generic Commands"*. Proceedings of the first International Conference on Human-Computer Interaction, INTERACT-84. London: IFIP.

[TT88] Tulli, S Thomas (1988): *"Screen Design"*, in M, Helander (Ed.) (1988): *"Handbook of Human Computer Interaction"*, Elsevier Science Publishers, B. V (North-Holland).

[GS91] Gitta Salomon (1991): *"New Uses for Colour"*, in B. Laurel (Ed.) (1991): *"The art of Human Computer Interface Design"*, Addison-Wesley Publishing Company, Inc. Massachusetts.

[SW92] Sousa Jorge & Wolsey, Lawrence (1992*): "A Time-Indexed Formulation of non-preemptive single machine scheduling problems"*, Mathematical Programming, 54 (1992) 353-367, North-Holland.

[MS93] M. Savelsbergh (1994): *"Preprocessing and Probing Techniques for Mixed Integer Programming Problems"*, ORSA Journal on Computing, Vol. 6, No. 4, 445-454, Fall 1994.

[NSS94] G. Nemhauser, M. Savelsbergh & G. Sigismondi (1994): *"MINTO, a Mixed INTeger Optimizer"*, Operations Research Letters, 15, 47-58.

[SN96] M. Savelsbergh & G. Nemhauser (1996): *"Functional description of MINTO, a Mixed INTeger Optimizer. Version 2.3"*, Report COC-91-03D, Georgia Institute of Technology.

[SN95] M.W.P. Savelsbergh, G.L. Nemhauser (1995): *"A MINTO short course"*, Report COC-95-xx, Georgia Institute of Technology.

[GP79a] Grotschel, M & Padberg, M (1979): *"On the symmetric Travelling Salesman Problem I: Inequalities"*, Mathematical Programming 16, 265-280, North-Holland.

[GP79b] Grotschel, M & Padberg, M (1979): *"On the symmetric Travelling Salesman Problem II: Lifting theorems and facets"*, mathematical Programming 16, 281-302, North-Holland.

[G80] Grotschel, M (1980): *"On the symmetric Travelling Salesman Problem: Solution of a 120-city problem"*, Mathematical Programming Study 12, 61-77, North-Holland.

[F91] Matteo Fischetti (1991*): "Facets of the symmetric Travelling Salesman Polytope"*, Mathematics of Operations Research, Vol 16, No. 1,

[CP80] Crowder, H & Padberg, M (1980): *"Solving Large Scale symmetric Travelling Salesman Problems to optimality"*, Management Science, vol. 26, No. 5.

[JRT94] Jünger, M, Reinelt, G & Thienel, S (1994): "Practical Problem-Solving with Cutting Plane Algorithms in Combinatorial Optimisation", Interdisplinäres Zentrum für Wissenschaftliches Rechnen, Universität Heidelberg, Preprint 94-24, Netherlands.

October, 1995 / Amushi@compapp.dcu.ie