Cell batching optimisation for the New Zealand Aluminium Smelter

Thorsten Sven Piehl
Department of Engineering Science
University of Auckland
New Zealand
tpie003@esu1.auckland.ac.nz

Abstract

The New Zealand Aluminium Smelter (NZAS) at Tiwai Point produces aluminium in tapping bays of 48 to 51 electrolysis cells. Three cells are tapped into one batch. A tapping bay consists of up to 17 batches. The purity of the aluminium produced differs from cell to cell. The premium of a batch increases more than linear with increasing purity. The objective of the cell batching process is it to maximise the overall premium of a tapping bay by choosing the optimal combination of batches.

The cell batching process is modelled as a Set Partitioning Model. This model is solved using the Revised Simplex Method and Branch-and-Bound (B&B) with a constraint branching approach. This approach has difficulties solving the smelter problem to optimality because the LP-IP gap and the B&B-tree, which has to be explored, are large. Improvements of the existing approach for dealing with the LP-IP gap and the tree size have been implemented. This paper outlines the improvement of the software currently used for the cell batching process of NZAS.

1. Introduction

In his year four project Tuck [4] approached the cell batching optimisation problem of the New Zealand Aluminium Smelter (NZAS) at Tiwai Point. NZAS produces aluminium by electrolysis in three regular and one short production line. A regular production line consists of four 150 meters long tapping bays. There are 51 electrolysis cells in each tapping bay. The short production line has 48 cells of a new type in a single 300 meters tapping bay. Each cell is tapped once per 24 hour time period during a day or a night shift into a crucible. A crucible is a "bucket", which can take the aluminium of up to three cells. These three cells have to be in one tapping bay. The purity of the aluminium differs from cell to cell. It depends mainly on the increasing age of the cell, the purity of the alumina used and the way the cell has been operated during its lifetime. The most common contaminants are iron, silicon, gallium and nickel. Samples of the aluminium are taken regularly during the production process to determine the purity of the aluminium. Further technical information about the production process can be found in Tuck's work.

High purity aluminium achieves a high premium on the metal markets and hence cells with high purity aluminium should be tapped into the same crucible. The objective of the resulting optimisation problem is to maximise the total premium of a tapping bay given the purity of aluminium produced in the cells of the tapping bay. Tuck described the

problem as a set partitioning optimisation model. Batches or triples of cells are grouped to maximise the total premium produced. Section 2 of this paper describes the mathematical model used. This model gave results increasing the former manual premium by over 10%. However, the solution process had unexplained difficulties in the Branch and Bound tree of the optimisation. Ryan [1] addressed these problems by implementing an integer allocation, which attempts to find an integer solution at each node of the tree. This solution is derived from the existing LP-solution by a greedy sequential heuristic allocation. However, Tuck's model was still unable to solve the problem to optimality inside the critical solution time of 2 minutes. The solution method and its problems are explained in Section 3. Section 4 to Section 6 discuss approaches to improve the solution by using a better integer allocation, by decreasing the LP-IP gap and by implementing an additional branching strategy based on the objective function of the LP-solution. Numerical results of the improvements are given in Section 7. These results show that the problem can now be solved to optimality inside the critical time with a further increase in premium of 3% to 4%.

2. Set Partitioning Model

The cell batching problem is modelled as a Generalised Set Partitioning Problem (SPP):

$$\begin{aligned} &\text{maximise} && z = p^T x \\ &\text{subject to} && Ax = e,\ s^T x \le D,\ x \in \{0,1\}^n, \end{aligned}$$

where A is a 0-1 matrix and $e^T = (1,\ldots,1)$. For each tapping bay the A-matrix consists of 51 rows (48 rows for production line 4). Each row of A is corresponding to a cell in the tapping bay and ensures that a cell appears in exactly one batch or crucible. The columns of A represent all valid triples of cells which can be tapped into the same batch. A variable or batch of three cells contributes exactly to 3 rows in the A-matrix. The row numbers are the same as the cell numbers in the batch. The A-matrix has unit values at these positions. The matrix has entries of zeros at all other positions. For example a batch made up of cells 1,2 and 3 is represented in the model by a column with zeros except for unit values in row 1, 2 and 3. The elements of A are defined as

$$\begin{aligned} a_{ij} &= 1 \text{ if cell i is included in batch j} \\ &= 0 \text{ otherwise.} \end{aligned}$$

The distance between cells, which can be tapped into the same crucible, has to be limited by NZAS because of the size of tapping bays and the semi-manual operation of the crucibles. The specified maximum distance between cells is called spread S. The spread is defined as the difference between the maximum and minimum cell number in a batch. The additional constraint $s^T x \le D$ reflects the fact that some cells are allowed to have a wider spread than others. The vector s has unit value in position j, if the batch j has a wider spread than a minimal spread S_1, and zeros otherwise. A wider spread than a maximal spread S_2 is not allowed in the optimisation. The right hand side D of the constraint is the maximum number of batches allowed with a spread between S_1 and S_2.

When cells are off-line due to rebuild or repair, they will not be tapped. That leads to one or more batches that do not have the full taping weight of three times 1280 kg. Off-line cells are treated in the model as cells that have zero tapping weight. They can appear in any batch during the enumeration process ignoring the spread limit. Despite the appearance of off-line cells, the approach always gives triples of cells in a batch.

The columns or variables of this model can be enumerated easily. The model has 52 constraints (51 cell constraints + 1 spread constraint) for tapping bays in production lines 1 to 3 and 49 constraints for production line 4. The number of variables depends on the spread values and the number of off-line cells in the tapping bay. It lies between 300 and 7000 generated variables. A solution for this SPP will be made up of exactly 17 variables at unit value (16 for production line 4). These variables represent the chosen batches. The other variables will have zero values.

The objective function of the optimisation model is a linear function. The premium p of each variable (batch) is the price that the metal content of the associated batch would achieve on the metal market. This price depends mainly on the purity of the aluminium and on a low iron contamination. The aluminium purity of a batch is the weighted average of the cell purities involved in the batch. The weight is the actual tapped metal weight of the cells. This weight is assumed to be a constant 1280 kg per cell. Batches that include off-line cells have a lower metal weight. Their premium depends on the metal purity and the metal weight. If a batch has one off-line cell, its premium is 2/3 of a batch with 3 cells and the same metal purity.

The premium increases more than linearly (almost exponentially) with increasing aluminium purity. Aluminium with purity less than 99.6% costs the company money because this aluminium has to be reworked; aluminium with purity up to 99.9% achieves a moderate price; and aluminium with purity better than 99.9% has an exceptionally high premium. The premium function is shown in figure 1. The premium does not only depend on the aluminium purity but also on the iron contamination. Metal with the same aluminium purity but with higher iron contamination has a lower premium. The graph below assumes low iron contamination. Having a varying iron contamination leads to several premiums for the same aluminium purity.

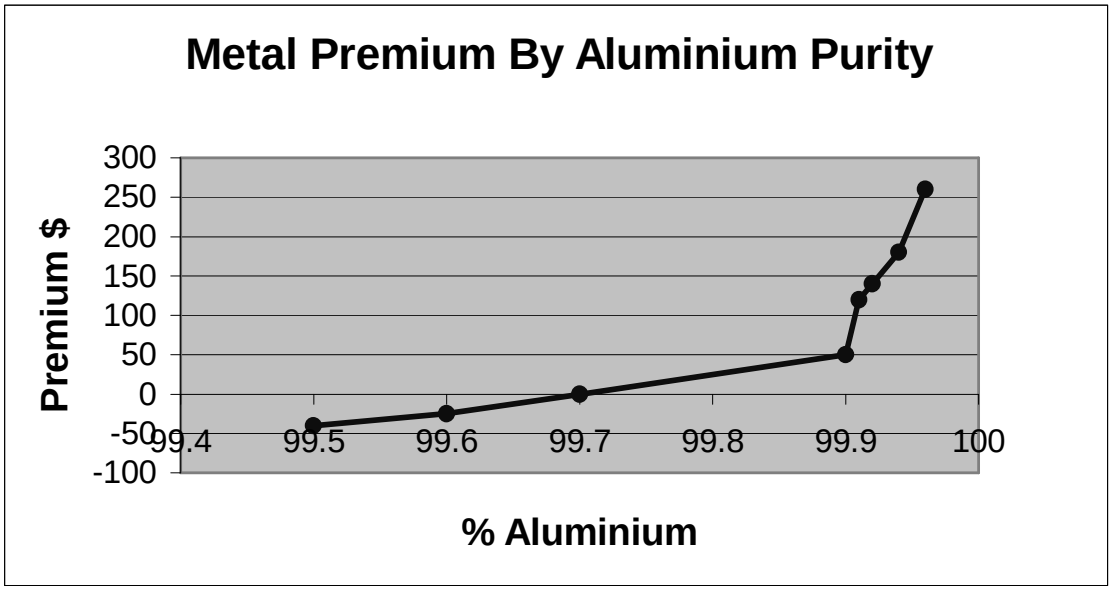

Figure 1: metal premium function by aluminium purity

3. Tuck's Solution Process and Problems

Solving the GSPP model involves two steps. First, the integer restriction is relaxed and the resulting linear program is solved. This leads to a mainly fractional solution with some integer structure. In the second step, the fractional variables have to be forced to integer values using a branch and bound algorithm.

The Branch&Bound(B&B)-strategy used is a constraint branching approach defined by Ryan and Foster [3]. Imposing a one-branch means that all variables including just one cell of the two cells (constraints) on which is branched, are banned from the descending LP. Imposing the zero-branch bans all variables that include both cells. Every branch removes variables from the LP and decreases the number of potential variables that can appear in the solution. The one-branch is the default branch because more variables can be banned using this strategy. Ryan and Falkner [2] showed that odd-order cycles are responsible for fractional solutions in SPP models and that constraint branching breaks these odd-order cycles. The branching strategy is effective and leads to an integer optimal solution. However the size of the tree is huge. Hence it takes a long time to reach a satisfying or even optimal solution. There are two reasons for this.

The first reason is that after imposing a sequence of branches, the LP feasible solution can become integer infeasible. This happens because the branching decisions can lead to isolated cells which can not occur in a feasible batch within the prescribed spread. This infeasibility can be manually predicted at an early stage in the B&B-tree but the B&B-algorithm does not terminate before it detects LP-infeasibility, which usually is a long sequence of nodes deeper in the tree. The unnecessary enumeration of such a system of integer infeasible branches consumes a lot of computational time. The second reason is that the initial LP-IP gap at the root node can be large. That leads to a long series of branching decisions in the B&B-tree that are concerned with decreasing the LP-objective value to bring it close to the integer solution. The decrease in the LP-objective is small if constraint branching is used. This is good when the gap is small. However if the gap is large, a large tree depth is necessary to close the gap. A side effect of the large LP-IP gap is that more LP-feasible, but IP-infeasible, branching combinations exist, which have to be explored. Most of these branches would not occur with a tighter LP-IP gap.

Therefore, the above GSPP model cannot be solved to optimality in reasonable time and optimal IP-solutions are hard to find. To achieve optimal IP-solutions and to cut down the computational time to solve the problem, three protection mechanisms are built into the algorithm. First, the algorithm terminates if the IP-solution is within 5% of the LP-solution. Second, Ryan implemented a robust integer allocation heuristic. This heuristic computes at every node of the B&B-tree an integer solution produced from the fractional LP-solution. The solutions are good estimators of the optimal solution and they are found early in algorithm. So the best-allocated IP-solution is used as lower bound in the tree. Third, the number of successful integer allocations is limited to 25. After that the algorithm terminates, the IP-solution is the best found allocated solution so far. This ensures that a solution is always found but the solution quality is not always optimal.

4. Integer allocation

4.1 Old Integer Allocation

The old integer allocation heuristic is a fast heuristic, which provides an integer solution at every node of the B&B-tree. It forces an integer solution from a fractional solution by making a sequence of greedy decisions. First, all variables (batches) with unit value in the fractional solution are fixed at that value. After that, the cell with the lowest cell number and fractional coverage is identified. The algorithm tries to find the batch with the highest premium which includes the first fractional cell. If there is no valid batch among the uncovered cells for the first cell, the cell with the next largest cell number, which is at unit value, is found. The batch including that cell is set to value zero and the algorithm starts again with the smallest cell number which is not yet covered.

This algorithm always finds an integer solution but it does not take into account the structure of the problem. Cells at the edges of the tapping bay have the least number of valid batches due to the spread constraint. It is sensible to allocate these cells first but the algorithm does it sequentially starting with the low cell numbers. Hence, cells with high cell numbers are allocated at the end of the algorithm. For most of the runs, no valid batches exist for these cells because of isolated cells. In these cases, variables which are integer in the LP-solution will be taken out of the solution. The underlying integer structure of the LP can be destroyed. Another weakness of the algorithm is the greedy sequential decision process. This is locally optimal but not necessarily globally optimal.

4.2 New integer Allocation

The new integer allocation process fixes all variables with unit value. It computes for all cells with fractional batches the number of possible valid batches that can be built from the unfixed cells. The algorithm allocates the cell with the least number of valid batches first. This overcomes the problem of the sequential approach.

The next step is to overcome the purely greedy sequential decision process. The algorithm creates a decision tree. Not only the batch with the highest premium for the least covered cell is investigated but also a maximum of W other valid batches for this cell. The batches with the highest premiums make up this neighbourhood with maximal W elements. The value of W determines the width of the decision tree. A number D of integer allocation decisions for each outcome of the different batches W, found in the first step, are made. These decisions are also based on the least cover criteria and investigate W different batches for each further decision. A feasibility check is performed at each allocation step. If the choice of a batch leads to an isolated cell, the choice of this batch will be rejected. The value of D determines the depth of the decision tree.

A decision tree of width W and depth D is created. The partly allocated solution with the best objective value found in the tree is chosen. This best batch combination serves as starting point for the next decision tree. This process is repeated until all cells are covered or isolated cells appear and the process can not produce an integer feasible batch combination. In this case, the old integer allocation is called to ensure that an integer solution can be found.

The decision after exploring the tree is a greedy decision, but it takes a sequence of different batching possibilities into account. The size of the tree is controlled by the choice of the parameters *W* and *D*. Values of 1 for both parameters lead to a greedy sequence of decisions based on the least cover criteria. A value of 17 for *D* and a high value for *W* lead to a total enumeration of all possible batch combinations. The chosen parameter combination for *W* and *D* are 3 and 5. This parameter combination performs best considering the trade-off between computational time and objective value of the solution.

5. Cell Cuts

The initial LP-IP gap of the problem can be large. Therefore it is sensible to apply methods to close that gap. One approach is to impose valid inequalities. This will strengthen the integer properties of the model and will lead to a better B&B performance.

A natural set of valid inequalities exists for the cell batching optimisation. We can maximally tap three cells into one batch in the cell batching process. If we have a fourth cell, we will need at least 2 batches to tap all 4 cells. If we consider 7 cells, we will need at least 3 batches to tap these cells, etc. More precisely, the number of batches necessary to tap a number of cells *c* is always greater than ceiling(*c*/3). The number of batches contributing to a cell is the same as the sum of the variable values in the LP-solution which include the cell. For example if the cell triple (1,2,3) is in the LP-solution with unit value, the sum of variables (batches) contributing to all three cells 1, 2 and 3 is 1. Consider that batches (1,2,3), (1,2,4), (1,3,4) and (2,3,4) are in the solution with a value of 1/3 each. Clearly, the cell constraints (SPP-constraint) are satisfied. For example, the sum of all variables contribute to cell 1 is 3 *1/3 = 1 (batches 1,2 and 3). The sum of all variables contributing to cell 1 and/or 2 is 4 * 1/3 (batches 1,2,3 and 4), to cell 1 and/or 2 and/or 3 is 4 * 1/3 and to cells 1 to 4 is 4* 1/3. But the number of batches, which contribute to four cells, has to be at least 2 for an integer solution. This is a violation of the intuitive constraints described above.

In more general terms, let *U* be a subset of cells in a tapping bay and *c* the number of cells in the set *U*. Valid inequalities for the cell batching problem have the form:

$$u^T x \geq \text{ceiling}(c/3)$$

where $u_j = 1$ if x_j contributes to at least one cell of the set *U*

$u_j = 0$ otherwise.

The number of all such valid inequalities is large. All possible combinations of subsets *U* have to be built to cover all available valid inequalities, for example $U_1 = \{1,2,3,4\}$, $U_2 = \{1,2,3,5\}$, $U_3 = \{1,2,3,4,5,6,7\}$, etc. Hence, not all inequalities can be implemented initially. The reason for that is that the time to solve the resulting LP would increase rapidly. If we consider only subsets of 4 cells and spread S_2 of 9, the number of all available valid inequalities is more than 8500 constraints. It is not even practical to implement this subset of constraints.

5.1 A-Priori Cell Cuts

Only a small subset of valid inequalities is implemented a-priori. The first implemented valid inequality is build from the subset $U_1 = \{1\}$, which consists only of the first cell.

Cell 2 is added to built the next subset $U_2 = \{1,2\}$ and the next constraint. This sequential process stops when U_{51}, consisting of all cells in the tapping bay (U_{48} for production line 4), is reached. In addition to these sequential forward cell cuts, the backward equivalent cell cuts are also imposed. The backward cell cuts start with the last cell. The second to last cell is added after that and the process is finished, when all cells are added to the subset. The number of a-priori cuts added to the initial GSPP-model are102 respectively 96 constraints. This leaves the system with 154 respectively 145 constraints in total.

5.2 Dynamic Cell Cuts

Dynamic cuts can be generated during runtime. The LP-solution is searched for all valid inequalities that are violated. These valid inequalities are added to the problem and the LP is solved again. This process is repeated until no valid inequality is violated.

Dynamic cuts can be generated at every node in the B&B-tree, but the generation process was only tested at the root node. Due to the large number of existing valid inequalities and the necessary enumeration process to find them, this process is too expensive to implement (NP-hard), especially since a large number of inequalities has to be found to decrease the LP-objective. Furthermore, the complete set of valid inequalities is not facet defining which was assumed beforehand. This is illustrated in the example shown in figure 2. None of the cells 4 to 9 violates any GSPP-constraint or any valid inequality but the solution is still fractionally. Hence the expensive generation process can not be justified because the achieved solution is not necessarily integer.

Batch Value	Cells		
0.5	4	5	8
0.5	4	6	7
0.5	5	6	9
0.5	7	8	9

Figure 2: counter example for the facet defining assumption

6. Objective Cuts

The LP-relaxation of the GSPP-model is not a strong relaxation. For example assume the highest premium in an LP-solution is 40. The relaxation permits that the sum of all variables with premium 40 can be 9.5. However, the sum of variables with the same objective has to be integer in an IP feasible solution because fractional tapping is not allowed. The LP takes advantage of the weak relaxation and finds the optimal solution using fractional batches. Obviously, it would have taken 10 batches with value 40, if this had maximised the objective. This gives the motivation to implement an additional branching scheme based on the sum of variables with the same objective. The objective cut idea has to be implemented as a branching approach because forcing the sum of variables down as well as forcing them up can lead to feasible LP-solution. The optimal solution can either lie on the down or on the up side of the cut. Hence, the objective cuts are not globally valid.

Only a small number of different premiums exist in a tapping bay. Obviously, the sum of all batches with the same premium has to be integer. Because of these two facts, cuts can be derived from the premium of variables. The values of variables with the same

objective are added to get t. This process starts with the highest premium h in the LP-solution. If the sum t of the variables is fractional, a branch will be imposed by adding a cut. If the sum is not fractional, the next highest premium h is tested and so on until no fractional sum can be found. If no fractional sum is found the constraint branching scheme will be used. This is implemented for all nodes in the B&B-tree. The one-branch limits the sum of variables with the premium h to no more than the largest integer number smaller than t. This is the default branch. It will decrease the LP-bound by at least the fractional part of t times the difference of the premium h and the next lower premium. The zero-branch limits the sum of variables with premium h to no less than the smallest integer number larger than t. This branch is predicted to be either infeasible or have an objective much lower than the objective of the one-branch. The cuts have the following form for the one-branch and zero- branch respectively:

$$o^T x \leq \lfloor t \rfloor \text{ or}$$
$$o^T x \geq \lceil t \rceil$$

where $o_j = 1$ if $p_j(x_j) = h$

$o_j = 0$ otherwise.

This branching scheme reduces the LP-objective significantly. It can be observed that if only one objective cut has to be imposed, the optimal integer solution will have the value of the LP-objective at the objective cut node. Objective cuts strengthen the LP-relaxation. The actual LP-IP gap should rather be measured after imposing cell cuts than before, especially because not implemented valid inequalities exist. The objective cuts have the most significant impact on the solution procedure used for the cell batching optimisation. Due to the large reduction in the LP-solution by imposing objective cuts, the cell batching optimisation can now be solved to optimality well inside the critical time.

7. Results and Conclusion

The old integer allocation was not well designed because it ignored the problem structure. The initial LP-IP gap was large and leaded to a deep B&B-tree. The optimisation problem could not be solved to optimality and even a good solution was not guaranteed. The improvements explained in the previous chapters address all these problems. Therefore, every single step of the improvement process has an impact on the solution but only implementing all of the changes guarantees the final optimal result. The following results are computed for one production day (13 tapping bays) at the smelter. Despite the small example set, conclusions can be drawn and the results will be validated in a series of tests at the smelter.

The new integer allocation improves the IP-solution by around 2%. The drawback of the new approach is the increase in computational time from 8.41 seconds total solution time to 14.48 seconds. This is a time increase of almost 100%. The new allocation performs better for 5 out of 13 tapping bays and equally for the others. At least the same objective value can be guaranteed by implementing new allocation. The average number of nodes created per tapping bay is a bit smaller as before, 13.2 compared to 13.62. The number of nodes is the same for bays with the same objective, the five improved bays show a significant decrease in nodes created. This is especially important for the further

improvements because a good solution can be achieved with less branching. Nevertheless, the new allocation process has the least impact on the solution process.

Introducing a-priori cell cuts decreases the LP-solution from a total premium of 15933.33 to 15500 which is a decrease of around 1%. However, the effect of the cuts varies depending on the problem. No changes in the LP-solutions can be observed for 4 tapping bays. These bays do not violate the implemented subset of valid inequalities in and hence no change in the LP-objective can be observed. The decrease for the remaining bays goes down to a maximum of 3%. Bringing down the LP-bound leads to a further improvement of the IP-solution of 1%. The important fact of a-priori-cuts is that the average number of nodes created drops from 13.2 to 4.85. That means that a solution inside the bound gap of 5% can be reached in most cases before the maximum number of allocations of 25 is performed. A solution inside the bound gap can not be found for 3 tapping bays compared to 6 before. The size of the B&B-tree shrinks significantly. A further interesting observation is the impact of the cuts on the integer allocation. The new allocation can only be completed successfully 25% of the time without the cell cuts due to isolated cells. After the cuts are implemented, the percentage rises to almost 100%. Hence, implementing the a-priori gets rid of most solutions which lead to isolated cells violating the feasibility check of the new allocation process. The drawback of a-priori cuts is the increase in time necessary to solve the LP due to the increase of constraints. The tripling of the constraints leads to a time increase for solving the LPs from 0.04 seconds average to 0.4 seconds average for every LP in the problem. This time increase and the more frequent use of the new allocation process lead to a drastically solution time increase of 350% to 36.24 seconds compared to Tuck's approach.

All computation so far used the bound gap value of 5% and the maximum number of allocation of 25. The second value was necessary to ensure that the algorithm terminates inside the critical time. This protection is loosened to a value of 50 allocations and the bound gap is set to 0.1% (effectively zero) for the last improvement step. It is possible to solve the cell batching optimisation to optimality after implementing the objective cut branching scheme. The algorithm achieves in the final state an overall increase of 4% in premium compared to Tuck's implementation. This solution is the proved optimal solution because all branches of the B&B-tree are enumerated. The number of nodes created is 3.46. This is not significantly less than the average for the earlier approach. However, this result includes one bay which contributes to more than half of the number of nodes of the overall number of nodes. The average is 1.375 created nodes without this bay. The reason for that outlier is the fact that the initial sum of variables for all objective values is integer. After imposing a few constraint branches, the algorithm switches to objective cuts and the optimal can directly be realised. It can be observed that if only one objective cut has to be imposed, the optimal integer solution will have the value of the LP-objective at the objective cut node. After imposing this objective cut branch, the algorithm finds the associated integer solution by imposing just a few constraint branches to remove odd-order cycles or by successfully completing an integer allocation with that objective value. The objective cut branching scheme has the most significant effect on the solution process because the LP-objective can be decreased fast. This tightens up the LP-IP gap and the effective constraint branching scheme can remove odd-order cycles to achieve integrality. This process is supported by a sophisticated integer allocation which helps reducing the size of the B&B-tree.

The final implementation of the cell batching optimisation software consisted of Tuck's GSPP-model combined with the implementation of the new integer allocation, of a-priori cell cuts and of objective cut branching. This approach solves the problem to optimality and hence, the research for the cell batching optimisation is completed. Results for the different solution steps are shown in figure 3.

	Tuck's approach	New allocation	Cell cuts	Final implementation
LP-Premium	15933.33	15933.33	**15500**	15500
IP-Premium	14670	14925	15045	**15260**
LP-IP Gap in %	8.6%	6.75%	3.02%	1%
Premium-Increase	0	255	375	590
Premium-% Increase	0%	2%	3%	**4%**
Total Solution Time	8.15	14.48	36.24	**28.13**
Time-% Increase	0%	78%	345%	245%
Average # Nodes per B&B	13.62	13.2	4.85	3.46

Figure 3: results of improvement process

The achieved improvement in premium will have a significantly impact on NZAS in the future. However, the cell batching optimisation is only a part of a larger production-scheduling problem. This larger problem involves decisions about the finished aluminium product and about the quality of aluminium necessary for the product. Time requirements for the production as well as for the delivering process have to be taken into account. Constraints on the production schedule for each furnace, on the capacity of the tapping crane and on the time sequence for the tapping of each cell exist as well. This larger problem has not yet been investigated in depth and gives opportunities for further research.

Acknowledgement

The work reported in this paper is the Master's project of the author towards a Master's of Operations Research. This work was done under the kind supervision of Prof. D. M. Ryan, Head of the department of Engineering Science, University of Auckland. I would like to thank Prof. Ryan for giving me the opportunity to study in New Zealand under his supervision.

References

[1] D. M. Ryan, Optimised Cell Batching for New Zealand Aluminium Smelters Ltd, Annual Conference of ORSNZ, (1998).

[2] D. M. Ryan, J.C. Falkner, On the integer properties of Scheduling Set Partitioning Models, European Journal of Operations Research, 35 (1988), pp 442-456.

[3] D. M. Ryan, B. A. Foster, An Integer Programming Approach to Scheduling, in A. Wren (ed) Computer Scheduling of Public Transport, North Holland, (1981), pp 269- 280.

[4] S. Tuck, Optimal Cell Batching for New Zealand Aluminium Smelters Tiwai Point Facility, Year Four Project, Dept of Engineering Science, University of Auckland, (1997).