Greedy heuristics for the multiple-choice, multidimensional knapsack problem difficulties for heuristics with a periodic delivery schedule problem

Tazlauanu, S.V.

Award date:
2010

Eindhoven, June 2010

Greedy heuristics for the multiple-choice, multidimensional knapsack problem: Difficulties for heuristics with a periodic delivery schedule problem

by
Tazlauanu Stefan Victor

Student identity number 0641376

in partial fulfilment of the requirements for the degree of

Master of Science
in Operations Management and Logistics

Supervisors:
Dr.ir. R.A.C.M Broekmeulen, TU/e, OPAC
Dr. T. Van Woensel, TU/e, OPAC

TUE. Department Technology Management.
Series Master Theses Operations Management and Logistics

Author (Stefan Tazlauanu) may be contacted at:
s.v.tazlauanu@student.tue.nl
stefanvictort@gmail.com

KEYWORDS: Knapsack, multiple-choice multidimensional, MMKP, greedy heuristics

June 2010

Abstract

Several types of heuristics are described that can find good, feasible solutions, with little amount of calculations for the Multiple-choice Multidimensional Knapsack Problem (MMKP). These heuristics are relevant to large MMKP problems that cannot be solved exactly and for which the other heuristics are computationally intensive. These heuristics perform well and are fast enough for applications that update a current solution continuously. The different heuristics are sensitive to the structure of the problem and can be used to gain information about a MMKP problem. The heuristics are a rapid optimization of an initial solution. Moser's algorithm has been used in many prior researches. It has moderate complexity for calculation time and has been shown to find good solutions. Moser's algorithm performance is compared to the proposed heuristics. Four experiments are used to test the performance of the heuristics: a test problem for which the bounds are swept through different levels, a set of generated MMKP problems with different structures, a test-bed of problems obtained from the literature that have been solved with other heuristics and a real problem from a previous research that can be considered as a MMKP problem.

1) Introduction

This research looks at how a mathematical problem, the multiple-choice multidimensional knapsack problem (MMKP), can be used to model large problems (involving thousands of decisions with dozens of alternatives involving dozens of shared resources). Several heuristics are presented for optimizing an initial starting selection of alternatives to a good feasible solution for large problems in a minimal number of computations. Finding rapidly a good solution to a large problem is important for applications that continuously review a solution. Minimal calculations also guarantee that, as the problems grow, the time to find a solution does not become too long.

In this research three types of heuristics are presented. The first heuristic type, named generic, is the fastest way of optimizing an initial solution to the MMKP problem. The second and third types of heuristics, named delta and directed, involve slightly more computations, however they remain fast. Several different ways of implementing these three types of heuristics are shown. The combinations of the heuristic type with the way the heuristic is implemented result in heuristics that find different solutions for the MMKP problems.

The main contribution of this report is to present the different approaches of building heuristics that can solve large MMKP problems rapidly and that give information on the structure of the MMKP problem.

The MMKP problem (equation 1) refers to situations such as: given a set of k decisions each of which can be approached with m alternatives: what is the best solution, selecting one and only one alternative, for every decision so that the value is maximized while n

constraints are respected. The alternatives must be completely independent in terms of value and resource consumptions even though they use the same set of constrained resources.

$$(\mathrm{P})\ \text{Max} \sum_{i=1}^{k}\sum_{j=1}^{m} p_{ij}x_{ij}$$

$$\text{S.T.} \quad \sum_{i=1}^{k}\sum_{j=1}^{m} n_{ijl}x_{ij} \le n_l \quad \forall \ \ l=1,n$$

$$\sum_{j=1}^{m} x_{ij} = 1 \quad \forall \ \ i=1,k$$

$$x_{ij} \in \{0,1\} \quad \forall \ \ i=1,k;\ j=1,m$$

Equation 1: The MMKP problem: (P)

To model a real situation with the MMKP problem, first we enumerate the available alternatives for the decisions to be taken. Special care must be taken to identify all resources used by any of the alternatives. Next, the resource consumption for each of the alternatives must be clarified. The resources that are constrained must be identified and the maximum usage for all alternatives of the problem must be found. Then, the value for each of the alternatives must be calculated. This can be done by careful inspection of the benefits of an alternative with respect to costs. The value must allow differentiating among alternatives in terms of how advantageous they are. The MMKP problem can then be solved to select an alternative and only one alternative for each of the decisions.

The structure of a MMKP problem is defined by its parameters. The structure of the MMKP problem is also defined by any patterns in its parameters. The parameters of the MMKP problem are the number of decisions, the number of alternatives, the number of constraints, the values for all the alternatives and the resource consumption for all the alternatives. A careful inspection of the structure of the MMKP problem can allow fine-tuning these heuristics. A good solution can be found in a reliable fashion by selecting the best solution from a set of several heuristics that find good solutions given the MMKP problem structure.

In the current research the structural elements of the MMKP problem will be considered. First, the effect of different problem resource constraints for the different constrained resources would be considered. Second, the effect of using different initial solutions for finding a good feasible solution will be addressed. Third, for each alternative the resource consumptions can be seen as the different dimensions of a vector. The similarity between the direction vectors for resource consumptions between the different alternatives is another structural feature of the MMKP problem. This can also be seen as overall correlation among resource consumptions used by different alternatives. Last, the correlation of value with constrained resources for the alternatives is studied.

Several experiments are attempted. A test problem with few elements is created to test that the heuristics behave as expected. The test problem shows that the heuristics find different solutions and that they are dependent on the problem resource constraints being changed. Next, large MMKP problems are generated and solved with the heuristics. The resulting solutions are used to prove and discuss how the heuristics are dependent on the MMKP problem structure. Next, a test-bed of problems from the literature is also solved using the MMKP heuristics in order to compare the resulting solutions with other resulting solutions available in the literature. Finally, difficulties in modeling a real situation as an MMKP problem are shown with the help of a case study. This particular MMKP problem instance renders most of the presented heuristics inefficient. It is discussed how the case study can still be modeled and solved as a MMKP problem.

This report is organized as it follows. Section two gives a literature review. Section three describes the heuristics used to solve the MMKP problem. Section four gives a brief account of how the heuristics were implemented in a software solution. Section five presents several experiments that were used to test the heuristics. In this section the structure of MMKP problems is discussed and difficulties with a real case are addressed. Section six presents managerial insights and section seven shows possible directions for future research. Finally section eight gives the conclusions of the research.

2) Literature Study

The MMKP problem is a result of the combination of two knapsack problems that have been addressed in the literature as: the knapsack problem involving multiple decisions with multiple alternatives given one constraint (MCKP) and the knapsack problem involving multiple constraints when selecting alternatives for one decision (MDKP). For a review of the different models that are part of the knapsack problem family and various methods in solving these problems the reader is referred to the books: (Martello and Toth, 1990) or the more recent (Kellerer et al., 2004).

There is ample literature addressing various approaches that can be taken to solve the MMKP problem all involving advantages and disadvantages. Many algorithms used to solve the MMKP problem are discussed in (Han et al., 2010). It is mentioned that the structure of the MMKP has not been intensely studied. In this research the structure of the MMKP problem will be addressed further. The paper presents a comprehensive set of results for the required computation time to find a solution given different correlations among constraint consumptions and values. The paper also looks at the effect of resource availability on computation time. It is shown that MMKP problems with average resource availability and high correlation between resource consumptions and values for an alternative take a longer time to find a solution. Heuristics that do not find an exact solution perform more poorly on such MMKP problems. The symmetry of the MMKP problem is said to be known to cause difficult problems. More heuristics that can solve the MMKP problem are also discussed in (Akbar et al., 2006). The heuristics presented in this report have faster computation time than other heuristics from the literature. They are intended to obtain a good feasible solution for large problems rapidly.

A heuristic based on Lagrange multipliers for solving the MMKP problem is introduced in (Moser et al., 1997). The heuristic finds a solution with a computational complexity of $O(m(n-g)^2+m*n)$ where there are m resources, n alternatives and g decisions. Moser heuristic has been repeatedly addressed in literature as a heuristic finding good solutions in a reasonable amount of time. Moser was implemented together with other heuristics in this research to offer a means for comparison.

Several researchers have used the MMKP problem in improving solutions for multimedia applications. In (Shabani et al., 2007) Moser's algorithm is used successfully in utilizing the network resources more efficiently for multiservice direct-sequence code division multiple-access (DS-CDMA) cellular networks. In (Shabani et al., 2007) it is mentioned that an exact branch and bound algorithm has a computational complexity of $O(2^{M^2N})$. Such a heuristic will take too long as the problem size increases. Similarly, (Khan, 1998) shows in his dissertation how MMKP can be used to insure quality of service in a multimedia application. He uses two algorithms to solve the MMKP: a branch and bound heuristic that branches based on the bound obtained through linear relaxation of the problem and a heuristic that is close to Moser's approach in computational complexity and also gives good results. The present research would like to address whether MMKP problem solutions can be found for even larger problems. Such problems are frequent in industrial engineering when looking at flows of production or flows of finished products.

(Hifi et al., 2004) present a heuristic in order to find a good initial solution and then perform local search to improve the solution of the problem. They link their research and results to a set of problems available for scientific purposes. (Hifi et al., 2006) shows how column generation can be used to extend local search. The problems used in (Hifi et al., 2004) were used for comparison with the performance of the heuristics presented in this research. The heuristics from this research will involve less calculation and will find a solution with lower total value. However, the heuristics can be used with much larger problems. It will be shown that the solutions found are in average 95% of the value of the solutions from (Hifi et al., 2004).

3) Heuristics for solving MMKP

The heuristics presented in this section serve several purposes. They find a feasible solution for the MMKP problem. They can handle very large problems. They are fast in finding a solution. They are dependent on the problem structure and will perform better for a certain structure and worse for another structure. Using several heuristics and choosing the best solution will be faster than applying a computationally intensive heuristic. A good solution can be found reliably, that is for all instances of a MMKP problem, by selecting the best solution from a set of several heuristics. The structure of the MMKP problem being solved can be identified by looking at which heuristics give a good solution and which perform poorly.

In this section, a general approach to build a greedy heuristic to solve the MMKP problem is first presented. Next, two equations are introduced: one for an aggregation function and one for an alignment penalty function. These functions will be used by the heuristics. The following three subsections introduce three heuristic types (generic, delta and directed) that can be used given the general approach from the first subsection and using the function presented in the second subsection. Equations are presented on how to calculate the decision factors for selecting an alternative for each of the heuristic types. For the delta and generic heuristic types one possible way of calculating alignment penalty with the function introduced in the second subsection is given. Next, the way Moser heuristic was implemented is discussed. For each of the heuristic types the calculation complexity is presented. Finally, the other aspects that are addressed in building different heuristics for each of the three types of heuristic are presented in the last subsection.

3.1) A greedy like approach to building heuristics for solving the MMKP problem

In order to solve the MMKP problem rapidly a greedy like algorithm approach is taken for all heuristics. Each alternative has a decision factor evaluated and then alternatives are iteratively selected. The following steps can summarize this approach:

1) Select initial solution.
2) Calculate decision factor of all alternatives and sort alternatives for each group.
3) Sort groups based on their best alternative.
4) Iterate until end criteria is reached:
- **a)** Select or discard best alternative.
- **b)** If the alternative is selected, discard any other alternatives in the group that would worsen the solution, and then move the group to the set of groups modified. Otherwise check if the next alternative in line is still best, if not move group to new position based on this new best alternative.
- **c)** Recalculate any decision factors if necessary
- **d)** Resort if necessary
- **e)** If no more alternatives that improve the solution are available, then bring back all groups that have been modified and allow further modifications.

Section 4 and appendix 1 diagrams 1-3 show how the heuristics were implemented in a dynamic link library software solution implemented in the c programming language.

The above shown greedy algorithm, proceeds by selecting the next best selection for an alternative, given a ranking scheme for all the decision in the problem, until the end criterion is reached. Once an alternative is selected for a decision it is assumed to remain the best alternative for the decision. Decisions cannot be further changed. If the end criterion is not reached after all decisions have been changed to their best alternative, the next best alternatives are considered for all decisions. It is assumed these next alternatives are less efficient than the previous selection however they still improve the results.

3.2) The aggregation and alignment equations used by the heuristics

For each alternative, the decision factor for selecting the alternative is calculated as a ratio of value to consumption of constrained resources. Value is taken as an input. The consumption of constrained resources is calculated as the aggregate of constrained resource consumptions using equation 2. Table 1 in appendix 1 shows the results of changing the value of p, the power, in equation 2. In the end p=2 is selected. Since all resource consumptions are greater or equal to 0, the aggregate is similar to the Euclidian length. However, different instances of MMKP problems will perform better for different functions used to calculate the aggregate resource consumption. This will be highlighted by the case study.

$$\text{Aggregate} = \operatorname{sgn}\left(\sum_n |x_n|^p \operatorname{sgn}(x_n)\right) * \left|\sum_n |x_n|^p \operatorname{sgn}(x_n)\right|^{1/p} \qquad \text{where p is an integer}$$

Equation 2: Calculating aggregate over n dimensions (either aggregate of resource consumption for an alternative or aggregate of differences in using different resources between two alternatives)

In order to respect special considerations, such as preferring alternatives that use resources that are more abundant, the decision factor of an alternative can be further scaled by a penalty factor. In this research alignment is considered as an approach of using a penalty factor. Alignment is defined by equation 3. Alignment is the cosine between the resource consumptions vector for an alternative and the penalty vector. Note that all resource consumptions are assumed to be null or greater in this research.

$$\text{Alignment Factor} = (V_c \bullet V_{dv}) / \sqrt{(V_c V_c^{'})}$$

where V_c is constraints vector, V_{dv} is penalty direction vector

Equation 3: Alignment

The alignment penalty is a number between 0 and 1. Since in many problems all alternatives of interest have an alignment greater then 0.707, that is an angle smaller than 45^o, the alignment can be raised to a power to accentuate its effects. The results of raising the alignment to a power are shown in Table 2 in appendix 1. For the rest of the paper alignment is taken as given in equation 3, without being raised to a power.

3.3) The "generic" heuristic type

The first and fastest heuristic type, named generic, is to calculate decision factor ranking for each alternative only once as shown in Equation 4.

$$\xi = V / C$$

Where:

- V is the value input.
- C is the aggregate obtained with equation 2 with p=2 and x_n=resource consumption input for nth constrained resource.
- V and C are always greater than 0.
- ξ is the decision variable or the ranking for the alternative.

Equation 4: Generic heuristic worth calculation

The motivation for this heuristic type is to find an optimization of the initial solution as fast as possible that is with minimal number of computation. To achieve this, decision factors for every alternative are calculated only once as shown in equation 4. These decision factors are then used to move towards a better solution as explained at the beginning of the section. The generic type heuristic is aimed at making best use of resources with respect to benefits obtained.

Once the problem is loaded, the time to find a solution depends on sorting all the decisions after a new alternative selection are made for every decision. This sort can be implemented in O(n Log n). It will happen as many times as there is a new choice made for all decisions. This happens less than the number of alternatives, since alternatives worsening the solution are dropped every time a new alternative is selected.

If alignment is used then the penalty direction vector, V_{dv} (equation 3), was the direction vector of the problem resource constraints. This means that alternatives with usage of resources similar to the problem resource constraints would be preferred. In this way, an alternative excessively using a resource that is more critical would be penalized. Normalizing the resource consumption causes all problem resource constraints to be one and the alignment would prefer alternatives with equal percentage usage of all constrained resources.

3.4) The "delta" heuristic type

The second heuristic type, named delta, involved calculating the decision factor rankings by looking at differences between alternatives and the currently selected alternative. Every time a new alternative was selected for a decision the decision factor rankings of the alternatives for this decision had to be recalculated. The delta heuristic type calculations are shown in Equation 5.

$$\xi = (V_{new} - V_{old}) / aggregate(C_{new} - C_{old}) \qquad (5a)$$

$$\xi = aggregate(C_{old} - C_{new}) / (V_{old} - V_{new}) \qquad (5b)$$

Equation 5a is used when initial solution involves alternatives with lowest overall resource usage. Alternatives with value lower than the current selection are discarded. Therefore $V_{new} - V_{old} >= 0$ *is always true*

Equation 5b is used when initial solution involves alternatives with highest value. Alternatives with overall resource consumption higher than the current selection are discarded. Therefore $aggregate(C_{old} - C_{new})) >= 0$ *is always true.*

For both Equation 5a and Equation 5b:

If denominator is 0 then:

$\xi = M/2 + (V_{new} - V_{old})$ *for 5a*

$\xi = M/2 + aggregate(C_{old} - C_{new})$ *for 5b*

If denominator is less than 0 and $\xi < 0$ *then*

$\xi = M + \xi$

Where:

- V_{new} is the value input of the alternative;
- V_{old} is the value input of the present selection;
- The 'aggregate' function is given by equation 2 (with p=2);
- C_{new} is the vector for resource consumption on all dimensions for alternative;
- C_{old} is the vector for resource consumption on all dimensions for present selection;
- M is a large number.
- ξ is the decision variable or the ranking for the alternative.

Equation 5: Delta heuristic worth calculation

The purpose of the delta heuristic type is to obtain better optimization of the initial solution compared to the generic heuristic type. This heuristic requires slightly more calculations as decision factors for all alternatives still relevant to improving a decision must be recalculated when the selection of an alternative for a decision is changed. The generic heuristic type is aimed at choosing alternatives that make best use of resources with respect to benefits. The delta heuristic type tries to maximize improvement in benefits given extra resource consumption all relative to the previous alternative selection for a decision.

In equation 5a, when the denominator is 0, improvement is possible with no extra consumption. This is always preferred to improvement that requires extra consumption. Finally if the denominator is negative then improvement is possible while reducing consumption. This is always considered the best situation and alternatives freeing more of the consumption while achieving same improvements of benefits are preferred.

To calculate the computational complexity let: n be the number of elements, g the number of decisions, l the number of constraints and m the number of alternatives. For the delta algorithm there is the additional calculation due to recalculating all alternative decision factors rankings when a selection changes a decision. The order of the

alternatives for a decision is sorted at the time of the calculation of the decision factors. (l+y)x operations are necessary to recalculate decision factor rankings for alternatives remaining for a decision, x<m represents the remaining alternatives, y<x/2 represents checks required to sort the alternative. Usually x and y get small due to discards and delta heuristic time performance is only slightly worse than generic heuristic. The maximum number of alternative selections is (n-g), if no alternatives are discarded in average there are m/2 alternatives left. The calculation complexity would in the worst case scenario be O(mn Log n + (n-g)(l+m/4)m/2)). However due to discards the solution is reached in much less than (n-g) iteration and the average number of alternatives remaining for a decision is small. Because of this the delta algorithm type is in practice only slightly more computationally intensive then the generic heuristic type as shown by timing calculations in appendix 4.

If alignment is used than the penalty direction vector, V_{dv} (equation 3), is the direction vector for the resources that are still free or missing depending on the initial solution for the algorithm. This means benefits are considered more valuable if more abundant resources are used or alternatively if more critical resources are freed. Note however that the criticality of resources would only be considered at the time when the selection for a decision changed. Even though the criticality of resources could be altered by further changes of alternatives for decisions, this would not be reflected in the calculation of decision factors for previous decisions. This is because the delta type heuristic priority is still deemed to be speed.

3.5) The "directed" heuristic type

The third heuristic type, named directed, involves calculating the decision factor rankings with respect to the most critical resource by looking at differences between alternatives and the currently selected alternative. Every time a new alternative is selected for a decision the decision factor rankings of the alternatives for this decision must be recalculated. The delta heuristic type calculations are shown in Equation 6.

$$\xi = (V_{new} - V_{old}) / (C'_{new} - C'_{old}) \qquad (6a)$$

$$\xi = (C'_{old} - C'_{new}) / (V_{old} - V_{new}) \qquad (6b)$$

Equation 6a is used when initial solution involves alternatives with lowest overall resource usage. Alternatives with value lower than the current selection are discarded. Therefore $V_{new} - V_{old} >= 0$ *is always true*

Equation 6b is used when initial solution involves alternatives with highest value. Alternatives with overall resource consumption higher than the current selection are discarded. Therefore $C'_{old} - C'_{new} >= 0$ *is always true.*

For both Equation 6a and Equation 6b:

If denominator is 0 then:

$\xi = M/2 + (V_{new} - V_{old})$ *for 6a*

$\xi = M/2 + agregate(C'_{old} - C'_{new})$ *for 6b*

If denominator is less than 0 and $\xi < 0$ *then*

$\xi = M + \xi$

Where:

- V_{new} is the value input of the alternative;
- V_{old} is the value input of the present selection;
- C'_{new} is the resource consumption along the critical dimension for alternative;
- C'_{old} is the resource consumption along the critical dimension for alternative;
- M is a large number.
- ξ is the decision variable or the ranking for the alternative.

Equation 6: Directed heuristic worth calculation

The purpose of directed type heuristics is to consider only the most critical resources. This may result in better solutions if one resource is especially scarce. In the situation were several resources are scarce or other heuristics always choose alternatives using the same resources a directed heuristic may be advantageous. Since all alternatives for all decisions must be reconsidered every time the critical resource changes, the heuristic requires more calculations.

In equation 6a, when the denominator is 0, improvement is possible with no change in the critical resource. This is always preferred to improvement that requires extra consumption of critical resource. Finally if the denominator is negative then improvement is possible while reducing consumption of critical resource. This is always considered the best situation and alternatives freeing more of the consumption of the critical resource while achieving same improvements of benefits are preferred.

To calculate the computational complexity let: n be the number of elements, g the number of decisions, l the number of constraints and m the number of alternatives. The directed algorithm must recalculate all decision factor rankings if the critical dimension changes. It is possible the critical dimension changes with every iteration and that there

are no discards of alternatives. This causes a computational complexity of g(n-g)(l+y)x with x<m and y<x/2. This can increase computational time significantly. In practice a heuristic only tries several alternatives for every decision. This means if T alternatives are selected for each decision computational complexity due to recalculating all decision factors given that critical resource change on every iteration is $Tg^2(l+y)x$ with x<m and y<x/2. Raising g to a power results in the heuristic time being considerably longer as it can be seen from the heuristic timing results in appendix 4.

3.6) Moser heuristic

Moser heuristic involves using Lagrange multipliers in deciding which alternative to select next. It requires recalculating change in Lagrange multipliers for all alternatives of all decisions at each iteration. It will take longer to solve then the directed heuristic presented in the previous section. Moser algorithm has been used many times in literature and it has been presented as one of the fastest ways to obtain a good solution. In this research it was implemented to see whether the fast heuristics proposed will perform as well in finding a solution. Moser algorithm is generally believed to obtain results that are above 95% of the best possible solution. Equation 7b presents how the decision factor is calculated based on the change in Lagrange multiplier for Moser algorithm equation 7a.

$$\delta_j = \{v_{j_k} - v_j - \mu_{i'} \sum_{i=1}^{m} (w_{ij_k} - w_{ij})\} / (w_{i'j_k} - w_{i'j}) \qquad (7a)$$

$$\xi = 0 \qquad \text{If } (w_{i'(jk)} - w_{i'j}) = 0,\ v_{(jk)} - v_j < 0,\ \delta_j < 0 \qquad (7b)$$

$$\xi = M - \delta_j \qquad \text{Otherwise} \qquad (7b)$$

Where:

- δ_j is the increase in Lagrange multiplier for critical dimension (Moser et al., 1997).
- ξ is the decision variable or the ranking for the alternative.
- $v_{(jk)}$ is the value of present selection;
- v_j is the value of the alternative;
- $\mu_{i'}$ is the Lagrange multiplier on the critical dimension;
- $w_{i(jk)}$ is the resource consumption on dimension i for the selection;
- w_{ij} is the resource consumption on dimension i for the alternative;
- $w_{i'(jk)}$ is the resource consumption on the critical dimension i' for selection;
- $w_{i'j}$ is the resource consumption on the critical dimension i' for alternative.

Equation 7: Moser's decision variable

3.7) Heuristics addressed in this research

For the three heuristic types (generic, delta and directed), there are additional considerations in how a heuristic is implemented. These considerations will be shown to

affect the solution found. Furthermore the effect will be different depending on the structure of the MMKP problem being solved.

For the generic and delta heuristic type, alignment penalty can be enabled or disabled to further advantage the selection of alternatives using a certain combination of resources.

For all three heuristic types the resource consumption can be normalized or not. Normalization is achieved by dividing resource consumptions with the problem resource constraints so that resource consumptions represent percentages of resources available.

The heuristics are evaluated from three different initial solutions that are used as a starting point. The initial solution ‘low resources’, involves selecting the alternatives with the least aggregate resource consumptions, equation 2. The solution discards all alternatives with a lesser value than the current selection. It proceeds by increasing the value by selecting alternatives with the greatest decision factor ranking while respecting constraints. This continues until all alternatives are exhausted.

The second initial solution, ‘equalized low’, minimizes and equalizes the usage of resources for the starting point. This is achieved by normalizing the constraints and selecting iteratively alternatives with minimal resource consumption on the most used constraint dimension. This method can select an alternative with high resource consumption except for a low on the critical dimension. Alternatives with less value then the initial choice are still discarded. This approach discards good alternatives and is less reliable than the ‘low resources’, however it performs better in certain specific contexts as it will be shown.

The third alternative, ‘high value’, selects as initial solution the highest values for each decision. It proceeded by maximizing the decrease in the resources consumed for a minimal loss in value. It discards alternatives with higher aggregate resource consumption than the current selections. For the ‘high value’ approach improvements can always be selected. Improvements decrease value and resource usage until the solution becomes feasible or all improvements are exhausted. Because improvements are selected discarding alternatives with higher aggregate resource consumptions, some time no solution is found. Because all constraints must be respected the final solution frees more than enough resources on many of the constraint dimensions.

To find a good solution, the solution found with a ‘high value’ heuristic has to be used as a ‘low resource’ initial solution. This way the extra freed resources are used. The heuristics following a ‘high value’ with a ‘low resource’ initial solution for one of the three heuristics are called ‘swap’.

4) Heuristics Implementation and testing of implementation

A dynamic link library (DLL) was created using the C programming language. The DLL implements the Moser heuristic and the heuristics discussed in the previous section. The DLL is self-contained and can be used from any application. The DLL is a C programming pointer solution. Because the computational complexity of the generic and delta algorithm was dependent on the sorting of the decisions (a list of alternatives), it was believed this could be implemented efficiently with C using pointers to limit overhead in manipulating the data. For the DLL each decision is a linked list of alternatives. The MMKP problem itself is loaded as a linked list of decision lists.

The DLL is handled using Matlab. This choice is made due to Matlab ability to handle large matrices, save data, read and write from xls tables and call DLL routines. The data for the problems as well as the results are recorded in xls files. Matlab linear solving module and a CPLEX interface were used for finding exact solutions for the test MMKP problem presented in section 5.1.

Seven tests for different problem resource constraints involving different heuristics were conducted and the solutions of the DLL were checked to be correct at each iteration of selecting or discarding the best improvement. The checks performed to test that the DLL is working properly on April 9, 2010, including the seven tests of finding a solution step by step, were recorded in a document that is not included in this report. This document is available if needed. Further modifications to the DLL after April 9 were documented in the header of the DLL files. The DLL was tested with the test MMKP problem described in section 5.1.

All the experiments described in this report were achieved from MATLAB using the methods described in Appendix 8 that made use of the DLL described above. The data and software used in this research are available if needed.

The reader can refer to Appendix 1 to see the interface to the DLL and how solving the heuristics was implemented in the DLL. The aim of this implementation was to achieve fast speed for solving the MMKP problem with the generic and delta heuristic types. The computational complexity for various steps in the DLL is shown in appendix 2. The timing results for solving MMKP problems with the DLL are given in the appendix 3.

5) Experiments

This section describes four experiments and their results. The first subsection introduces the test MMKP problem, a very small problem used to test that the DLL and the heuristics are working as expected. The second subsection presents an experiment were several MMKP problems were obtained that have been solved with other algorithms in the literature. These problems are treated as a test-bed used to evaluate the performance of the heuristics when compared with some of the other solutions presented in the literature. The third subsection presents a set of 15 large problems that were generated in this research to test the performance of the heuristics given different MMKP problem structures. Finally the fourth experiment involves a case study taken from a previous research conducted at the Technical University of Eindhoven. The selected problem can be modeled as a MMKP problem however several difficulties arise and only a few of the heuristics can be used. This is relevant to showing how one must be careful when modeling a real situation as a MMKP problem.

5.1) Test MMKP problem

The first experiment is intended to create a problem to test that the DLL is working as expected for all heuristics. The test MMKP problem is small involving only five decisions and eighteen alternatives. Two constrained resource consumptions are considered to allow bias towards either of the two constraints. Finally the problem resource constraints are swept through different levels to demonstrate how dependent the solutions of the heuristics are on the problem structure. Different heuristics will find different solutions for different problem resource constraints.

The MMKP test problem and the results for all heuristics with seven different problem resource constraints can be found in appendix 4. The results prove that the DLL works, that the found solutions are different for different heuristics, that the solutions do change if the problem resource constraints change and that the heuristics perform generally well, finding solutions that are better than the Moser heuristic solution.

5.2) Literature MMKP problems

Another experiment consists of evaluating the heuristic performance with regard to data presented in the literature. The heuristics were used to evaluate MMKP problems in two other scientific papers (Hifi et al., 2004) and (Khan et al., 2002). The results can be seen in appendix 6. The heuristics outperform Moser's algorithm. The performance of a group of 6 heuristics was selected and compared to the best known approximations of 20 problems from the literature (Appendix 6 table 10). These 6 heuristics were shown to be fastest to solve (appendix 4). The heuristics are shown to find solutions that are 94-97% of the best known approximation. Furthermore worst case is 92.8% and best 99.5%.

5.3) Generated MMKP problems experiment

Another experiment consists of generating MMKP problems with various characteristics. The motivation was to check how the heuristics solved large MMKP problems with different structures. The size was chosen to be 3000 items with 20 alternatives and 20 constraints. This size was deemed large enough and the problem fit on a excel sheet of a size of 60000 by 23 cells. The generated problems can be reviewed in appendix 5, which shows how the various parameters of the MMKP problems were generated and gives information on the reasoning behind each of the problems. Tables 6-8 in appendix 5 can be addressed to see results for the solutions found with each of the heuristics for each problem.

5.3.1) Interpretation of solutions for MMKP generated problems

The first three generated MMKP problems show that correlation among resource usage of different alternatives is necessary in order to create alternatives with different aggregate resource consumption. If resources are abundant, a high value initial solution performs better; as resources become scarce the low resource initial performs better. The delta algorithm is also shown to outperform the generic and directed algorithms for these three problems.

The generated problems 4, 5, 6, 7 and 8 scale the problem resource constraints and resource consumptions with the same factors for different resource availabilities. As before the scarcity of resources causes low resources initial solution to perform better. Furthermore, high scarcity causes an equalized low resource initial to perform very well in several problems. In the problems, 5, 6 and 8 the generic heuristic performs as well as the delta heuristic. During trials of generating problems, there were instances found, when an equalized low resource initial will find a feasible solution while a low resource or high value will fail to do so. Normalizing constraints improves the behavior of the directed algorithm for low resource in problems 4, 5, 6 and 7. For delta and generic normalizing is not having an evident effect. Alignment influences results; however it is not always benefic.

Generated problem 9 shows that equalized low resource can perform well even when resources are abundant if some resources are much more critical than others.

Generated problem 10, 11 show delta heuristic with high value initial to perform better given more resources available, even if value relates to resources differently.

The generated problem 12 shows that for random values not correlated to resources, the generic heuristic performs well.

The generated problem 15 shows that for clusters of correlated resource usages with different characteristics the directed heuristic performs best. Also the swap heuristic is

necessary for generic and delta heuristic as they free many more resources than needed while trying to find the best feasible solution.

The delta and swap algorithms always perform well.

The results of the generated problem confirm that the heuristic can be used to gain additional information about a MMKP problem and that for different MMKP problems with different structures, different heuristics will perform better.

5.3.2) MMKP structure

Three initial solutions were presented. One initial solution starts with alternatives with the highest values and then decreases resource consumption for minimal decrease in value. This approach performs best if resources are abundant and the best solution includes some of these highest value alternatives. Another initial solution starts with alternatives having the lowest aggregate resources consumption and then increases resource consumption while trying to maximize increase in value. This approach is best in situations where resources are highly restricted and the best solution might include some of the lowest resource aggregate alternatives. The last initial solution tries to minimize all dimensions equally by selecting choices with minimal resource consumption on the most used dimension. This initial may be necessary if one resource is very critical such that it should be given more attention than minimizing aggregate resource usage.

One structural element of the MMKP problem is the availability of resources. High value initial solution will perform better for high availability, while low resource for low availability. If a resource is critical and is used to heavily by low resource aggregate alternatives then an equalized low initial will do better and may be necessary to find a feasible initial solution.

Another structural element of the MMKP problem is the problem resource constraints. An aggregate measure of resource consumption that assumes units are equally important can be wrong. Constraint usage can be normalized, this changes resource usage to reflect percentage used with respect to bounds. This can make aggregate measurement more useful. This structural element is reflected by how critical resource consumptions are with respect to the problem resource constraints. If some resources are more critical than others in terms of usage normalizing the constraints should help improving the results. Normalization did not have any effect in the generated problems.

Another aspect is which combination of resources an alternative uses. Two alternatives with similar aggregate resource consumption may use different resources. To differentiate between these alternatives, an alignment penalty can be used. An alternative is penalized for using a different combination of resources than the preferred one. Some possibilities for penalty vectors are: the problem resource constraints, the resources that are left to be used in the case of a low resource initial, or the resources that are left

missing in the case of a high value initial. In the generated problems alignment was found to have random effects on the solution that was found.

As discussed in literature (Han et al., 2010) correlation among variables is another structural feature of the MMKP. There are two such correlations: correlation of value with aggregate constrained resources and correlation among resources used by different alternatives. Correlation of values with resource consumption makes finding a solution more difficult and requires a good aggregate calculation function or results will shift away from the best possible solution. Correlation of resource consumptions for different alternatives is the same as similarity of direction vectors for two alternatives.

The MMKP problem structural elements identified are:

- Availability of resources
    - Initial solution choice will impact heuristics solution.
    - Generic heuristics may perform better for low resource availability or whenever only one alternative is selected such as when an equal low resource initial performs well.
- The criticality of specific dimensions
    - Normalization may improve results
    - Directed heuristics may perform better
- The similarity in resource consumptions direction vector for alternatives
    - Not thoroughly tested
- The correlation of values with resource consumption
    - Delta heuristics may perform better when correlation is present
- The correlation of resource consumptions for different constraints
    - Directed heuristics may perform better

5.4) Case study involving real MMKP problem

5.4.1) Real case study introduction

In a previous research (Buel, 2009), a schedule for when to deliver products to retail stores is found. Ton van Buel has developed a simulation that evaluates all possible delivery schedules for the products at a given store. The simulation finds the number of deliveries that will occur for a product, the probability they will occur on a certain delivery schedule day and the number of replenishments occurring at a store on days when there is no delivery. Next, a heuristic is used to select the schedules that are best for delivering products to a store. This step is very similar to a MMKP problem and can be decomposed into several MMKP problems.

The model is described by section 4.5.1 and 4.5.2 in (Buel, 2009), these sections are included in appendix 9 as a reference.

In this real case the decisions are represented by the individual products and the alternatives are the schedules that are to be considered for each of the products. The

relevant constraints are the number of products that can be assigned at a warehouse and the space available for delivery on a given day.

Equation 16 in appendix 9 taken from (Buel, 2009) shows that the relevant cost was made of two components: the drop cost for truck deliveries and the number of idle replenishments (replenishments at the store when there is no truck delivery). Since alternatives for the MMKP problem must be independent the drop cost could not be included in the MMKP problem. The value of each alternative was calculated using equation 8 bellow, which is based on equation 15 in appendix 9, except for including concurrent replenishments also (replenishments that happen at the store during working hours). Replenishments were subtracted from 100 to obtain the value of an alternative. This is because the number of replenishments should be minimized. 100 was used so that all values are greater than 0. Inquiry of the data showed that one product had as many as 66 concurrent replenishments (replenishment at the store during working hours). Idle replenishment (replenishments at the store in the morning before the store opens on a day when there are no deliveries) could be at most 6.

$V_{ij} = 100 - H_{ij} + OL_{ij} - H_{conc}$
Where:
V_{ij} : value for alternative j of item i
H_{ij} : weekly number of planned replenishments
H_{conc} : weekly number of concurrent replenishments
OL_{ij} : average number of order lines
Equation 8: Value calculation for alternatives

5.4.2) Difficulties using the MMKP problem to model the case study

There are several difficulties in modeling this case study as a MMKP problem.

First, the aggregate for differences in resource consumptions between alternatives is approximately 0. This is because all alternative consume one spot in a warehouse and involve the same total deliveries (demand does not change), divided differently among delivery days. Because of this the delta algorithm should not be used since scaling by 0 is meaningless.

Furthermore, for the aggregate function (equation 2), using p=2 causes lowest aggregate resource consumption for alternatives with highest value. Both low resource and high value initial solutions would select the same alternatives and discard all other alternatives. This is due to total deliveries, or demand, being spread over more days: the square root of the sum of squares decreases. Instead, the aggregate resource consumption should be calculated with p=1 in equation 2. However if p=1 all aggregate resource consumption would be approximately equal except for small differences that are random due to rounding errors. Resource consumption aggregate was set to a constant value and aggregate method in the DLL was modified to simply return 1.

Another difficulty is that once all alternatives have an aggregate resource consumption of one, no feasible low resource initial solution can be found. Since all alternatives have the same aggregate resource consumption, the first alternative is selected every time and it uses the same resources. A low resource initial solution like the delta heuristic simply does not make sense.

Finally, the only difference between assigning an item regionally or nationally in terms of resource consumptions is having 1, 0 or 0, 1 for two of the constraints. This means that the two alternatives will be identical in terms of value and aggregate resource consumptions. Alignment could be used to differentiate between the two alternatives however the generic heuristic does not update the decision factors. One location will be preferred over the other always for the generic heuristic with alignment enabled. A low resource initial solution will deplete all space for the preferred location and than assign the last products that have more replenishments to the less preferred location. For a high value initial solution, preferring one location always will cause no solution being found. The delta heuristic cannot be applied because of the lack of differences in aggregate resource consumption. The directed algorithm will work. However, it is also slower in finding a solution and it will always prefer one alternative to the other when the critical resource is not one of the location constraints.

It is also interesting to notice that for this problem the constraints can be grouped in two clusters: 6 constraints related to the delivery days, 2 constraints related to the location. While the directed algorithm will consider the most critical constraint, either a delivery day or a storage location, it will not consider the criticality of the second cluster of constraints. A different directed heuristic could be implemented to allow the definition of cluster of constraints. Such a heuristic could order the clusters in terms of criticality and then if several alternatives are best within the first cluster a decision can be taken with respect to the second cluster constraints.

Based on the above observations, the following heuristics can be applied: the swap heuristic for directed and generic heuristic type, the generic heuristic with equalized low resource initial, and the generic heuristic with low resource initial set to null alternatives (value and all resource consumptions set to 0). All heuristics used to solve the MMKP were set to enabled normalization and disabled alignment penalty. Given the highly structured MMKP problem considered these heuristics end up in going through a precise selection of alternatives this is discussed in more detail bellow.

5.4.3) Solving the Dutch retailer MMKP problem

Looking at the case study, two stages can be identified for solving the MMKP problem:

- STAGE 1: For a single shop given the delivery capacity for every day and the results of (Buel, 2009) store operation model (see appendix 9) The MMKP problem can be solved to select the schedules to which each item should be assigned. This is done assuming one warehouse with unlimited space.
- STAGE 2: Same as above however two warehouses are considered one that is regional and one that is national, two more constraint dimensions are added for the maximum number of products that can be assigned to each warehouse.

STAGE 1

To find which schedule to choose at a store the case is modeled with the MMKP problem (equation 1) as stated bellow:

- x_{ij}: represents whether alternative j is selected for item i
- p_{ij}: is the value of an alternative calculated with equation 8
- n_{ijl}: is the resource consumption for constraint l for alternative j of item i
- l: represents the different constraints there are 6 for the capacities on the 6 days
- n_l: represents the bounds of the problem that is maximum deliveries for each day
- n_l for each day is set to 32.4 m^3 times the number of truck deliveries on the given day.
- The alternatives j are the schedules that are considered for each items, all schedules that can be considered given the selection of the days are considered

STAGE 2

To consider whether to store items at the regional or national warehouse two more constraints are introduced for the maximum storage at the two warehouses. MMKP problem can be solved again with different schedules being considered to restrict number of deliveries from national warehouse to regional warehouse.

5.4.4) Heuristics used to solve MMKP stage 1 and stage 2

5.4.4.1 Equalized low resource initial solution generic algorithm

For stage 1 equalized low initial solution will try to pick highest value for minimal consumption on the most critical resource as an initial solution. Therefore the heuristic will start by picking schedules that use all available days except for the day that is the critical dimension. This will result in an initial solution that is a combination of schedules using maximum frequency less a day. Since generic algorithm is then used, the heuristic will continue by trying to improve the value of a decision that is selecting any all day schedule or a better maximum frequency less a day.

For stage 2 the initial solution would alternate location between the two warehouses while selecting maximum frequency schedule until the deliveries on any of the week day become critical. Then schedules not using the particular delivery day are selected at the regional warehouse. When improving the solution alternatives at the regional warehouse will be attempted first.

5.4.4.2 Low initial solution, null entries, generic algorithm.

For stage 1, an alternative using no resources and having no value is added to the groups. These null alternatives are selected as a starting point by the low resource initial solution. The generic algorithm would than proceed by maximizing value through selection of the schedule with a maximum frequency. When resources are not sufficient other schedules get to be considered.

For stage 2, the same is true except that first items are assigned regionally and at the end items get assigned to less frequent schedules that do not use the busiest days and eventually get stored nationally.

5.4.4.3 Swap directed algorithm

For stage 1, the high value solution will select for all items the schedule with maximum frequency. The directed algorithm will then select an alternative that does not use the most busy delivery day. Instead the schedule with one day less frequency that has the least replenishments is selected.

For stage 2, the highest frequency at the national location will be selected if necessary.

Note that since all aggregates are set equal, a high value initial solution results in no discards of alternatives when selection changes.

5.4.5) Solution for case study

The solutions were obtained starting from an access database created with the store operation model (Buel, 2009). The operational behavior of any of the schedules at three of the stores in terms of deliveries and number of replenishments were simulated . The tables of the database were exported to excel. Using these exported excel tables Matlab was used to read the data and parse the MMKP problems, than solve these MMKP problems with the heuristics discussed in the previous subsection. Matlab was also used to export the results and create the excel tables with the solutions that can be seen in appendix 10.

The results show that given the structure of the problem the directed heuristic performs better than a generic heuristic. From the results it can also be seen that for given demand patterns, it is best to choose certain delivery days to minimize replenishments. The best

set of delivery days are highlighted for each frequency of the number of deliveries in a week (Table 11, appendix 10).

For every shop the total cost of the best set of delivery days for all possible frequencies of deliveries per week were calculated (Table 12, appendix 10). It is shown that the three stores are better served by different frequencies due to the differences in the demand they experience.

The solutions were also calculated for sets of delivery days that use more than one truck per delivery day. As expected, for the same number of trucks, replenishments are minimized if there is only one truck per delivery day (Table 13, appendix 10). .

Solutions for different deliveries per week frequencies were calculated also for the case when concurrent replenishments are not considered (Table 13, appendix 10). This allows comparison with (Buel, 2009) results. It can be seen that for frequencies of 1, 5 or 6 days (Buel, 2009) had better results, while for 2, 3 or 4 frequencies the results are slightly better than those achieved by (Buel, 2009). The difference for the 1 day frequency seems to indicate a slight variation of the data that was used. The slightly better MMKP results for the 2, 3 and 4 day frequencies can be attributed to the improvement of the directed heuristic over a generic heuristic. (Buel, 2009) heuristic is similar to a generic heuristic that only allows a fix amount of schedules to be used.

Solutions were also found for the cases when both a regional and a national warehouse are considered. The effects of using different sets of delivery days for the national warehouse were found to be different for the different stores. (Table 13, appendix 10).

5.4.6) Selecting location for items when all stores are considered

If all stores are considered, the solutions of the individual stores must be made to match for the location of products. Different stores will differ in the selection of products they would like to store locally. One way to tackle this difficulty is to assign faster moving product (products facing larger demand) to the local warehouse until the warehouse is full. Note that only the demand from solutions assigning an item locally should be considered when deciding which items to assign locally.

The problem involving all the stores cannot be modeled as the MMKP due to the interdependence between different shops that must select related alternatives. The equations describing the problem when looking at all the stores as well as a heuristic that can be used to find a feasible solution is shown in appendix 11.

6) Management Insights

It is important to note that the strategy presented in the introduction can only be applied in situations where alternatives for decisions use the same resources and are completely independent. This means that choosing an alternative should not affect in any way the benefits or resource usage of another alternative. In the case study this assumption of independence was not entirely true since the decision of a truck delivery on a certain date would affect many alternatives. This is why the decision for the delivery days was to be taken independently of solving the MMKP problem. Truck deliveries were considered as a problem resource constraint, constraining how many deliveries can be made in one day.

To model a situation with the MMKP problem the following steps are necessary:

- Enumeration of alternatives for all decisions and identification of resources used
- Clarification of resource consumption for the alternatives and identification of constrained resource limitations
- Identification of a function (taxonomy) to evaluate scalar benefit for each alternative
- Solving MMKP problem

Three types of greedy heuristic can be used to improve an initial solution for the MMKP problems:

- One time calculation of all alternatives ranking.
- Calculation of alternatives ranking with respect to current decision selection every time a different alternative is selected for meeting a decision.
- Calculation of alternatives ranking with respect to the most critical resource constraint every time the critical resource constraint changes.

The first two methods were shown to be very fast in finding a solution. The first method time to find a solution was shown to depend on the time to sort all the decisions based on their next best alternative.

It was also shown that the results of these different heuristics could be used to gain additional information on the structure of the MMKP problem and therefore the type of situation one is dealing with.

The three heuristics were shown to be adjustable to improve the obtained solutions by considering: normalizing resource consumption, using different initial solutions and using a penalty vector to generate a penalty factor for alternatives based on their alignment.

Most important, alternatives were ranked given a ratio of the scalar benefit to an aggregate measure of resource consumption. Understanding how to properly obtain the benefit scalar through consideration of all aspects relevant to differentiating between the alternatives is critical. So is using a proper function for getting the aggregate of resource consumption. Such a function was presented in equation 2 and the effect of changing p the power factor was shown to influence results and be critical when looking at a case study.

7) Future research

One topic not addressed in this paper was that of negative resource consumption, that is of an alternative that creates resources on some dimensions. Whether the aggregate function from equation 2 will still perform well in such a situation could be investigated. Furthermore, a situation where for some decisions the alternatives have both negative benefits and negative aggregates could also be investigated with the heuristics mentioned in this report.

Alignment could be further investigated as it was not clearly identified when its influence is negative and when it is benefic. Furthermore a heuristic based on alignment could be developed. For example, if there are l constraints, than 2^l vectors could be identified as all the binary combinations with l digits. All alternative could be assigned to a set matching one of this direction vectors. An algorithm could proceed to find a solution based on these sets of similarly aligned alternatives.

In the case study it was shown that the constraints can be clustered in groups of several constraints related to one type of resource (such as choice of delivery day and choice of location). A directed type heuristic can be considered that selects a set of best alternatives with respect to the most critical cluster and follows by ranking these alternatives based on consideration of the next most critical cluster of constraints.

Another aspect that can be further expanded is how heuristics performance can be used to identify MMKP problem structure.

The presented heuristics could be used for a software solution that continuously monitors the best selection of the current alternatives present in a database.

8) Conclusion

In this report it was shown how the MMKP problem can be solved with a fast greedy heuristic. This is important for solving large problems rapidly. Large problem can often be found in industrial engineering relating to flows of production or flows of finished products.

Three types of heuristics were presented for implementing the greedy heuristic for large MMKP problems. The three types differed in how they calculated the decision factors used to select the next best alternative for improving a solution. These heuristic types were shown to find different solutions at different speeds.

In order to calculate the decision factors used to rank the alternative two functions were introduced. A function was introduced that allowed calculating the aggregate of resource consumptions for an alternative or the aggregate of differences in resource consumptions between two alternatives. A penalty factor function was introduced as the cosine between an alternative resource consumptions and a direction vector in a space equal to the number of constrained resources.

Normalization of resource consumption and the type of initial solution used were shown to further alter the behavior of heuristics. Four initial solutions were considered during the research: selecting the alternatives with the highest value, selecting the alternatives with the lowest aggregate resource consumption, selecting dummy alternative with value and all resource consumptions set to 0, selecting alternatives such that all resource consumption is equalized while being kept minimal.

The presented heuristics were shown to solve large MMKP problems rapidly. The presented heuristics were shown to give results as good as Moser's heuristic. The presented heuristics were used to solve a test bed of problems that were previously addressed in the literature and were shown to find solution with a total value of around 95% of the best known approximations presented in the literature.

The heuristics were shown to be dependent on the structure of the MMKP problem and differences in the solutions given the different heuristics were shown to offer information about the structure of the MMKP problem being solved. The structural elements of the MMKP problem were discussed and experiments showing the interaction between this structure and the given heuristics were presented.

A real case study for a large problem was solved using the heuristics. Given the structure of the problem, the difficulties with the heuristics were discussed and it was shown how the heuristic could be used to still handle the problem.

Bibliography

T. van Buel, *A periodic delivery scheduling challenge: "The design of a model to optimize the delivery schedules for the products of the stores of PLUS retail"*, 2009, Master of science in operation management and logistics, Technical University Eindhoven

Bing Han, Jimmy Leblet, Gwendal Simon, "Hard multidimensional multiple choice knapsack problems, an empirical study"*, Computers & Operations Research*, 2010, 37, 172-181.

M. Hifi, M. Michrafy, A. Sbihi, "Heuristic Algorithms for the Multiple-Choice Multidimensional Knapsack Problem", *The journal of the operational research society*, 2004, Vol.55, No. 12, pp. 1323-1332

M. Hifi, M. Michrafy, A. Sbihi, "A Reactive Local Search-Based Algorithm for the Multiple-Choice Multi-Dimensional Knapsack Problem", *Computational Optimizations and Applications,* 2006, 33, 271-285

Hans Kellerer, Ulrich Pferschy, David Pisinger, *Knapsack problems*, 2004, Springer

Silvano Martello, Paulo Toth, *Knapsack Problems: algorithms and computer implementation*, 1990, Wiley

Martin Moser, Dusan P. Jokanovic, Norio Shiratori, "An algorithm for the Multidimensional Multiple-Choice Knapsack Problem"*, IEICE Trans. Fundamentals*, 1997, Vol. E80-A, No.3

Md. Mostofa Akbar, M. Sohel Rahman, M. Kaykobad, E.G. Manning, G.C. Shoja, "Solving the Multidimensional Multiple-Choice Knapsack Problem by constructing convex hulls", *Computers & Operations Research,* 2006, 33, 1259-1273

Mahdi Shabani, Keivan Navae, and Elvino S. Sousa, "A Utility-Based Downlink Radio Resource Allocation for Multiservice Cellular DS-CDMA Networks" *EURASIP Journal on Wireless Communications and Networking Algorithmic aspects of wireless networks*, 2007, pp9-19

Md. Shahadatullah Khan, "*Quality Adaptation in a Multisession Multimedia System: Model, Algorithm and Architecture"*, 1998, University of Victoria, Dissertation

Appendix 1: Dynamic link library implementation of heuristics

Diagram 1 shows the interface to the dynamic link library, which is the function through which the dll is accessed and the order in which this happens.

Diagram 2 and 3 show how the heuristics are implemented. The steps through which the dll goes after solving the problem is invoked through the interface are shown.

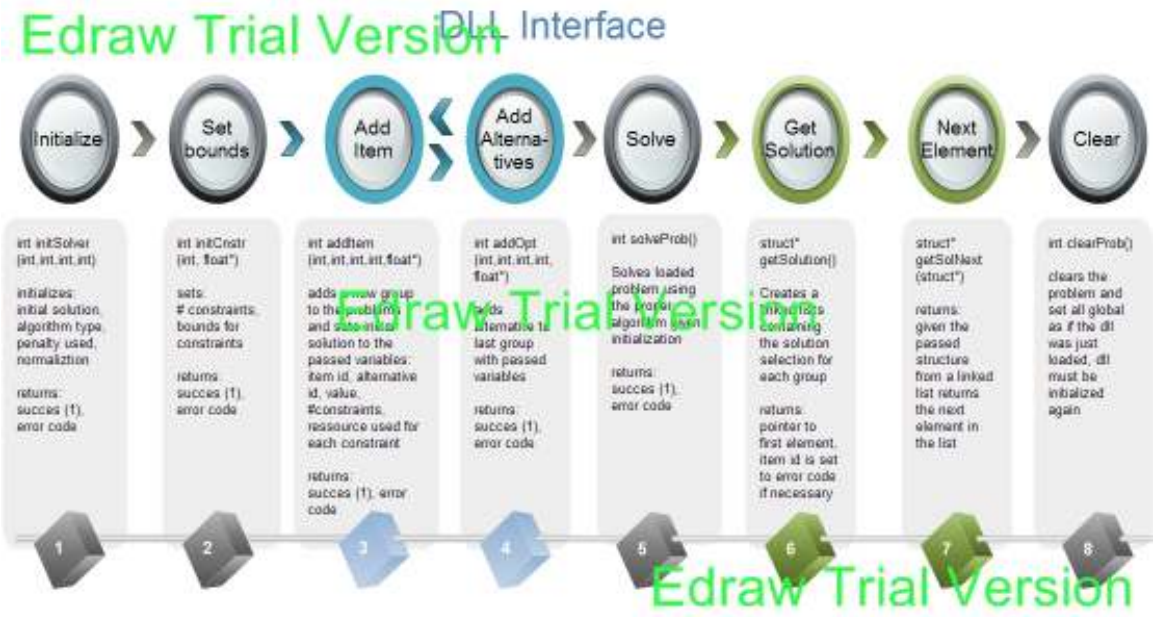

Figure 1: Dll interface

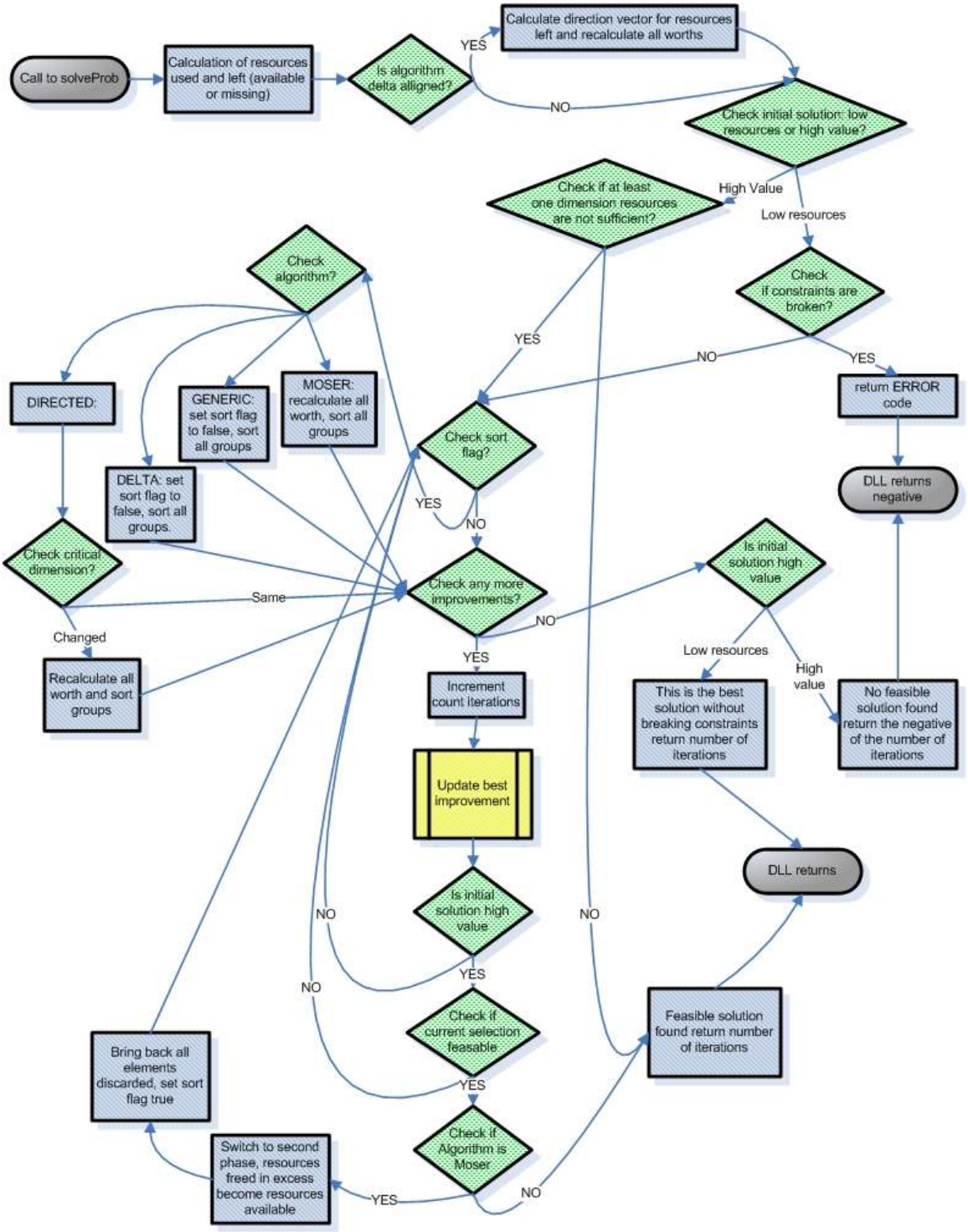

Figure 2: DLL implementation of solving problem for the algorithms (see figure 3 for implementation of updating the best improvement)

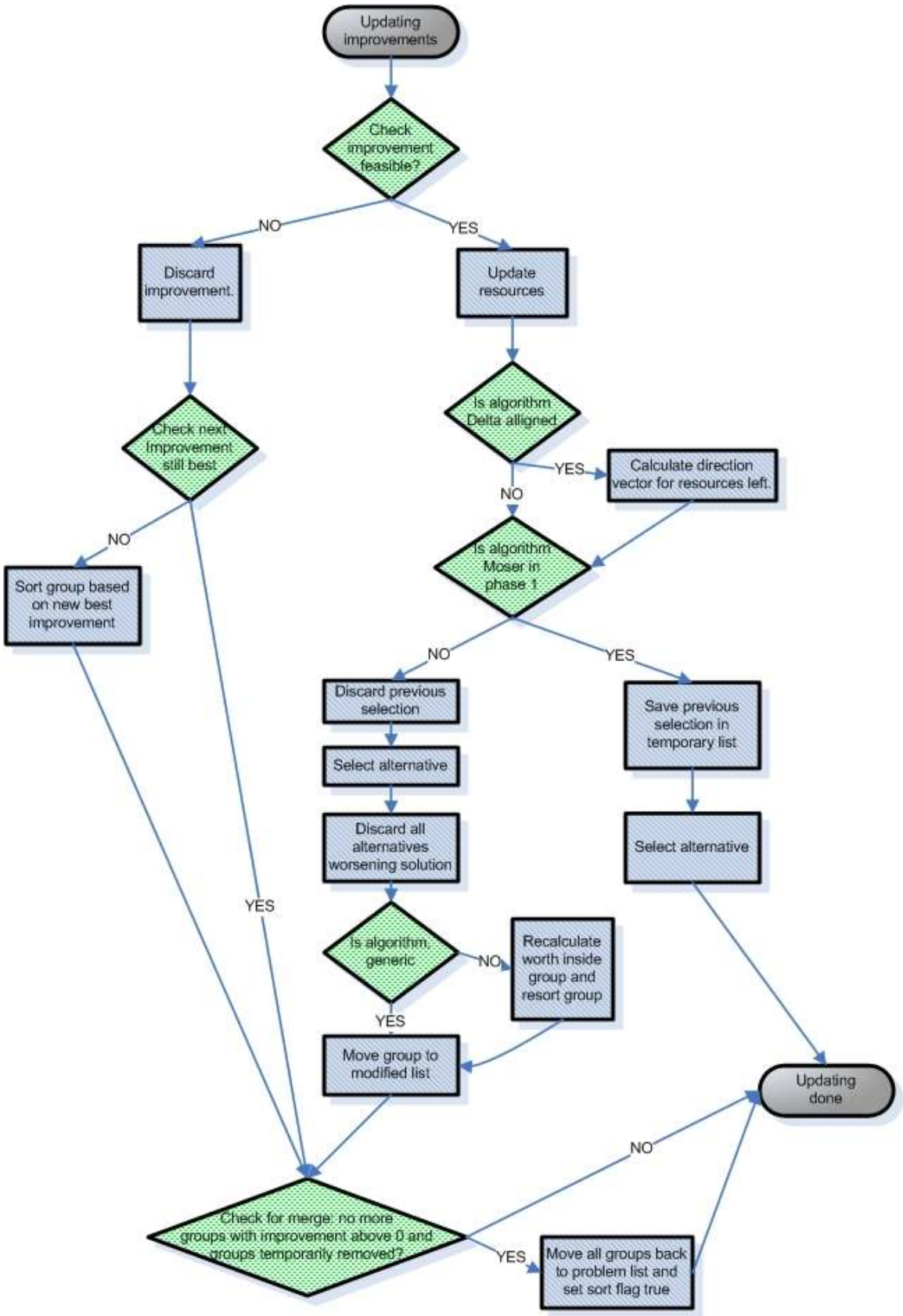

Figure 3: Implementation of updating improvement in figure 2

Appendix 2: Various calculation complexities for the dll

To calculate the computational complexity let: n be the number of elements, g the number of groups, m the number of options and l the number of constraints.

Calculating worth for an element is O(l)
Checking feasibility is O(l)
Updating resources is O(l)
Calculating direction vector for resources left is O(l)
Discarding alternatives worsening solution after new selection is O(x) where $x<m$ depending on how many alternatives are left.
Recalculating improvements in a group and sorting the group is $O(x(l+x/2))$ where $x<m$ depending on how many alternatives not discarded.
Sorting a group, if necessary, after its best improvement is rejected is O(g)
Sorting all groups once a selection made for each group $(g^2/2)$
Recalculating all worth $O(gx(l+x/2))$ where $x<m$ depending on how many alternatives not discarded.

The computational complexity of loading the problem into the dll is O(gm(l+m/2))

If all alternatives are discarded one by one as unfeasible, then complexity is: O((n-g)g)
If all alternatives are selected one by one with no discards and no worth calculations than complexity is $O((n-g)+mg^2/2)$

Therefore the worst case for generic algorithm is: $Max(O((n-g)g), O((n-g)+mg^2/2)$
The worst case for delta algorithm is: $Max(O((n-g)g), O((.5(n-g)m(l+m/4)))+(mg^2/2))$
The worst case for directed algorithm is: $O((.5g(n-g)m(l+m/4)) + (mg^2/2))$

The worst case will never occur in practice. That is because alternatives are discarded each time an alternative is selected therefore m could be replaced in the equations above with a small integer. Also when an improvement is not feasible it is possible that the next improvement in line is still best. Also sorts were assumed to take their worst amount of checks.

Clearly for the generic algorithm the sort of the groups $O(g^2/2)$ is an important number as every time a choice has been made for all groups this sort will occur.

For the delta algorithm there is the additional (n-g)(l+y)x operations necessary to recalculate worth in a group when a selection changes. $x<m$ represents the alternatives remaining in the group. $y<x/2$ represents checks required to find an alternative new place in the group.

The directed algorithm must recalculate all worth if the critical dimension changes. It is possible this critical dimension changes with every iteration. This causes a computational complexity of g(n-g)(l+y)x with $x<m$ and $y<x/2$. This can increase computational time significantly because of the g(n-g), which is equal to $(m-1)g^2$.

Appendix 3: Changing power for aggregation function

Pr. Test (45 30) is the test problem, see appendix 3, with constraints set to 45 30, this was selected because it was the only instance where the heuristics were found to perform poorly.
Pr. Lit. I8 is one of the problems taken from literature see appendix 4.
Pr. 15 is one of the problems that were generated see appendix 5.

Table 1	power p	Table 2	power p
M0	-1	A0	no alignment
M1	1	A1	1/alignment square
M2	2	A2	1/alignment
M3	3	A3	Alignment
M4	Max	A4	Alignment square
M5	Min	A5	Alignment cube

Generic	M0	M1	M2	M3	M4	M5
Pr. Test (45 30)	44	44	44	44	44	44
Pr. Lit. I8	35136	35177	35384	35258	35249	35102
Pr. 15	6673500	6711300	6720500	6722100	6801300	6633100

Delta	M0	M1	M2	M3	M4	M5
Pr. Test (45 30)	44	44	45	45	45	44
Pr. Lit. I8	35174	34698	35274	35285	35379	31242
Pr. 15	6050600	6722800	6703100	6692600	6819300	4257200

Table 1: Changing power for aggregate function (Equation 2)

Generic	M0	M1	M2	M3	M4	M5
Pr. Test (45 30)	44	44	44	44	44	44
Pr. Lit. I8	35384	35383	35177	35219	35403	35167
Pr. 15	6720500	6581900	6671600	6764600	6746500	6740400

Delta	M0	M1	M2	M3	M4	M5
Pr. Test (45 30)	45	44	44	45	45	45
Pr. Lit. I8	35274	34918	35121	35183	35133	35123
Pr. 15	6703100	6392900	6614900	6904100	6757300	6858400

Table 2: Raising alignment to a power to increase penalty

Appendix 4: Timing measurements

In the table below the numbers between brackets represent the number of iterations to reach a solution. F signifies a solution could not be reached. The numbers after '->' are the time in seconds to reach the solution.

MMKP_Pr5	***Low Res (NA-A)***	***High Val (NA-A)***	***Low Res (NA-A)***	***High Val (NA-A)***	***Eq Low res***	***Swap***
Generic	F->0	F->(.55-.55)	F->0	F->(.55-.56)	(29889)->.89	F->0.547
Delta	F->0	F->(.59-1)	F->0	F->(.58-.73)	(30826)->1.11	F->0.75
Directed	F->0	(5437)->129	F->0	(5304)->145	(28935)->1.03	(31604)->146
MMKP_Pr9	Low Res (NA-A)	High Val (NA-A)	Low Res (NA-A)	High Val (NA-A)	Eq Low res	Swap
Generic	(17242-17240)->.44-.44	(2711-2711)->.08-.06	(17232-17244)->.44-.45	(2711-2711)->.06-.06	(17841)->.45	(15448)->.59
Delta	(15109-14908)->.48-.48	(3222-4540)->.14-.19	(15004-14813)->.56-.5	(3223-3844)->.16-.20	(15014)->.56	(19328)->.77
Directed	(16333)->.64	(3116)->.16	(15849)->.56	(3117)->.16	(15941)->.56	(16981)->.69
MMKP_Pr3_C	Low Res (NA-A)	High Val (NA-A)	Low Res (NA-A)	High Val (NA-A)	Eq Low res	Swap
Generic	(23764-23894)->.55-.64	(3478-3491)->.12-.14	(23778-23795)->.56-.55	(3478-3491)->.12-.12	(23163)->.51	(21204)->.78
Delta	(19208-20234)->.53-.62	(3966-3842)->.14-.19	(19199-20300)->.53-.62	(3966-3842)->.16-.19	(20385)->.62	(17211)->.86
Directed	(20516)->.64	(4299)->129	(20564)->.73	(4317)->129	(20911)->.56	(22662)->130
	MMKP_Pr5	***MMKP_Pr9***	***MMKP_Pr3_C***	***MMKP_Pr7***	***MMKP_Pr8***	
Moser	F->5.7minutes	F->5.2minutes	F->65.7minutes	1340->2.3minutes	5284->8.4min	

Table 3: Timing measurements

From the table we can approximate for a problem with 60000 elements 60000 values and 1200000 constraints that: one Moser iteration takes about .1 second, one directed iteration can take an average of .03 seconds and delta and generic take about the same time which is in the range of microseconds per iteration. This also means that for the delta algorithm the time is more dependent on the sorts that are also part of generic algorithm than on actual calculation of new worth for a group.

For a description of the problems see appendix 5.

Appendix 5: Test Problem

The problem used to test the algorithm is given in the two tables bellow.

The first table summarizes the items with the given alternatives values and resource consumption. The second table gives different scenarios for various resource availability as well as different balance of resources available between the two dimensions defined by the constraints.

Option/Item	*item 1*	*item 2*	*item 3*	*item 4*	*item 5*
alternative 1	**20**	**15**	**12**	**25**	**16**
constraints	15,15	20,20	5,12	20,20	15,5
alternative 2	**2**	**5**	**6**	**4**	**8**
constraints	5,1	6,3	2,5	2,2	2,8
alternative 3	**15**	**10**	**8**		**10**
constraints	18,6	6,20	7,4		9,8
alternative 4	**8**		**10**		**11**
constraints	7,8		7,7		9,10
alternative 5	**9**				
constraints	2,10				

SCENARIOS	*scen 1*	*scen 2*	*scen 3*	*scen 4*	*scen 5*	*scen 6*	*scen 7*
constraint 1	60	40	26	65	45	30	24
constraint 2	60	40	26	40	30	50	30

Table 4: Test problem

Table 5 gives the results for the test problem. The highlighted row shows the bounds for the problem, the exact solution, and Moser's algorithm solution value. The three heuristics are presented on the three rows bellow. The first two columns are solutions if constraints are not normalized for initial 'low resource' or 'high values'. (NA – A) refers to whether alignment penalty was enabled or not. The next four columns are solutions for the case when resources are normalized. Again two columns present the 'low resource' and 'high value' initial solutions. This is followed by results for equalizing the consumption for low resources. The swap heuristic shows the solution obtained by using the 'high value' solution as an initial starting point for the 'low resource' algorithm.

Bounds/Exact/Moser	60 60	73	72			
Test: 60 60	***Low Res (NA-A)***	***High Val (NA-A)***	***Low Res (NA-A)***	***High Val (NA-A)***	***Eq Low res***	***Swap***
Generic	67 - 67	67 – 67	67 – 67	67 - 67	67	67
Delta	***73 - 73***	47 – 51	***73 - 73***	47 - 51	67	67
Directed	67	47	67	47	62	67
Bounds/Exact/Moser	40 40	51	50			
Test: 40 40	***Low Res (NA-A)***	***High Val (NA-A)***	***Low Res (NA-A)***	***High Val (NA-A)***	***Eq Low res***	***Swap***
Generic	***46 - 51***	32 – 31	***46 - 50***	32 - 31	46	***50***
Delta	***51 - 51***	42 – 46	***50 - 50***	42 - 46	46	46
Directed	46	42	46	42	46	47
Bounds/Exact/Moser	26 26	33	33			
Test: 26 26	***Low Res (NA-A)***	***High Val (NA-A)***	***Low Res (NA-A)***	***High Val (NA-A)***	***Eq Low res***	***Swap***
Generic	***31 - 31***	25 – 31	***29 - 29***	25 - 25	0	***29***
Delta	***33 - 31***	25 – 25	***29 - 29***	25 - 25	0	***29***
Directed	***33***	27	***29***	27	0	***33***
Bounds/Exact/Moser	65 40	67	60			
Test: 65 40	***Low Res (NA-A)***	***High Val (NA-A)***	***Low Res (NA-A)***	***High Val (NA-A)***	***Eq Low res***	***Swap***
Generic	***57 - 57***	32 – 52	***58 - 58***	***58 - 58***	***57***	***58***
Delta	***57 - 60***	42 – 44	***57 - 55***	50 - 44	***57***	***55***
Directed	***57***	42	***57***	50	***57***	***57***
Bounds/Exact/Moser	45 30	53	0			
Test: 45 30	***Low Res (NA-A)***	***High Val (NA-A)***	***Low Res (NA-A)***	***High Val (NA-A)***	***Eq Low res***	***Swap***
Generic	***44 - 45***	32 – 44	***44 - 44***	42 - 42	***44***	***45***
Delta	***45 - 45***	42 – 37	***45 - 45***	42 - 44	***44***	***45***
Directed	***44***	44	***44***	42	***44***	***45***
Bounds/Exact/Moser	30 50	51	49			
Test: 30 50	***Low Res (NA-A)***	***High Val (NA-A)***	***Low Res (NA-A)***	***High Val (NA-A)***	***Eq Low res***	***Swap***
Generic	***45 - 44***	42 – 42	***51 - 46***	42 - 42	***51***	42
Delta	***49 - 49***	36 – 37	***51 - 49***	41 - 39	***51***	44
Directed	44	41	44	41	***51***	44
Bounds/Exact/Moser	24 30	36	0			
Test: 24 30	***Low Res (NA-A)***	***High Val (NA-A)***	***Low Res (NA-A)***	***High Val (NA-A)***	***Eq Low res***	***Swap***
Generic	***36 - 36***	32 – 32	***36 - 36***	32 - 32	***36***	34
Delta	***36 - 36***	36 – 32	***36 - 36***	32 - 32	***36***	34
Directed	***36***	***36***	***36***	34	***36***	***36***

Table 5: Results for test problem (see comments above the table)

Appendix 6: Generated problems and their solutions

The problems generated to test the heuristics had all a size of 3000 groups of 20 alternatives with 20 constraints. They are described below. Uniform(x,y) chooses a value between x and y with uniform distribution. Normal(x,y) is a random value following normal distribution with mean x and standard deviation y. The italic comments summarize hypothesis related to each problem:

PROBLEMS 1-3: *SHOW EFFECT OF RESOURCE AVAILABILITY AND CORRELATION BETWEEN DIMENSIONS FOR RESOURCE CONSUMPTION*

MMKP_Pr1_NC: Baseline without correlation
all bounds 105%: C_i=1.05*50*3000 $\forall$ i=1…20
all weights: W_{ji}=50+uniform(-20,20) $\forall$ i=1…20 j=1…60000
all values: $V_j = 223+2*\sqrt{\sum_{i=1}^{20} W_{ji}^2} * |normal(0,0.25)| \forall$ j=1…60000

Due to the law of large number simply selecting the highest values will be feasible since all constraints follow the same distribution. If the bound are above 100% this remains the case, if the bounds are below 100% there will be no feasible solution.

MMKP_Pr1_C: Baseline
all bounds 105%: C_i=1.05*50*3000 $\forall$ i=1…20
all weights: R_j = uniform(-20,20) $\forall$ j=1…60000
W_{ji}=50+uniform(.8,1.2)*R_j $\forall$ i=1…20 j=1…60000
all values: $V_j = 223+2*\sqrt{\sum_{i=1}^{20} W_{ji}^2} * |normal(0,0.25)| \forall$ j=1…60000

Introducing correlation creates differences among alternatives as their resource consumption is clustered around different random values within the same distribution. Since there are ample resources a high value initial is expected to perform better.

MMKP_Pr2_NC: Baseline with bounds set to 90%, no correlation
all bounds 90%: C_i=0.9*50*3000 $\forall$ i=1…20
all weights: W_{ji}=50+uniform(.8,1.2)*R_j $\forall$ i=1…20 j=1…60000
all values: $V_j = 223+2*\sqrt{\sum_{i=1}^{20} W_{ji}^2} * |normal(0,0.25)| \forall$ j=1…60000

No solution due to all alternatives being somewhat identical

MMKP_Pr2_C: Baseline with bound set to 90%
all bounds 90%: C_i=0.9*50*3000 $\forall$ i=1…20
all weights: R_j = uniform(-20,20) $\forall$ j=1…60000
W_{ji}=50+uniform(.8,1.2)*R_j $\forall$ i=1…20 j=1…60000
all values: $V_j = 223+2*\sqrt{\sum_{i=1}^{20} W_{ji}^2} * |normal(0,0.25)| \forall$ j=1…60000

Decrease in bounds forces more choices towards lower value alternatives. Low resource initial expected to perform better.

MMKP_Pr3_NC: Baseline with bounds set to 70%, no correlation
all bounds 70%: C_i=0.7*50*3000 $\forall$ i=1…20
all weights: W_{ji}=50+uniform(.8,1.2)*R_j $\forall$ i=1…20 j=1…60000
all values: $V_j = 223+2*\sqrt{\sum_{i=1}^{20} W_{ji}^2} * |normal(0,0.25)| \forall$ j=1…60000

No solution due to all alternatives being somewhat identical

MMKP_Pr3_C: Baseline with bounds set to 70%
all bounds 70%: C_i=0.7*50*3000 $\forall$ i=1…20
all weights: R_j = uniform(-20,20) $\forall$ j=1…60000
W_{ji}=50+uniform(.8,1.2)*R_j $\forall$ i=1…20 j=1…60000
all values: $V_j = 223+2*\sqrt{\sum_{i=1}^{20} W_{ji}^2} * |normal(0,0.25)| \forall$ j=1…60000

Very low bounds force most choices around lowest values; a low resource initial is expected to perform better.

PROBLEMS 4-8: *SHOW EFFECT OF SCALING PROBLEM BOUNDS AND RESOURCE CONSUMPTION*

MMKP_Pr4: Baseline with bounds and weights scaled keeping proportions
scaling factors: S_i = 0.48 +.05*i $\forall$ i=1…20
all bounds 105%: C_i=1.05*50*3000*S_i $\forall$ i=1…20
all weights: R_j = uniform(-20,20) $\forall$ j=1…60000
W_{ji}=S_i*{50+uniform(.8,1.2)*R_j} $\forall$ i=1…20 j=1…60000
all values: $V_j = 223+2*\sqrt{\sum_{i=1}^{20} W_{ji}^2} * |normal(0,0.25)| \forall$ j=1…60000

Normalization should result in same results as before. If not normalized alignment could play a bigger role in finding a better solution.

MMKP_Pr5: Baseline with reduced bounds and weights scaled keeping proportions
scaling factors: S_i = 0.48 +.05*i $\forall$ i=1…20
all bounds 80%: C_i=0.8*50*3000*S_i $\forall$ i=1…20
all weights: R_j = uniform(-20,20) $\forall$ j=1…60000
W_{ji}=S_i*{50+uniform(.4,1.6)*R_j} $\forall$ i=1…20 j=1…60000
all values: $V_j = 223+2*\sqrt{\sum_{i=1}^{20} W_{ji}^2} * |normal(0,0.25)| \forall$ j=1…60000

This problem allows for lower correlation among dimension with low resource availability creating a problem with high scarcity of resources.

MMKP_Pr6: Baseline with reduced bounds and weights scaled keeping proportions
scaling factors: S_i = 0.69 +.03*i $\forall$ i=1…20
all bounds 80%: C_i=0.8*50*3000*S_i $\forall$ i=1…20
all weights: R_j = uniform(-20,20) $\forall$ j=1…60000
W_{ji}=S_i*{50+uniform(.8,1.2)*R_j} $\forall$ i=1…20 j=1…60000
all values: V_j = 223+2*$\sqrt{\sum_{i=1}^{20} W_{ji}^2}$ * $\left|normal(0,0.25)\right|$ $\forall$ j=1…60000

This problem is similar to problem 5, with different scales and correlation, again alignment and low initial solution are hypothesized to perform better.

MMKP_Pr7: Baseline with first and last constraints scaled
all bounds 105%: C_i=1.05*50*3000 $\forall$ i=2…19
C_1=0.5*1.05*50*3000, C_{20}=1.50*1.05*50*3000
all weights: R_j = uniform(-20,20) $\forall$ j=1…60000
W_{ji}=50+uniform(.8,1.2)*R_j $\forall$ i=2…19 j=1…60000
W_{j1}=0.5*{50+uniform(.8,1.2)*R_j} $\forall$ j=1…60000
W_{j20}=1.5*{50+uniform(.8,1.2)*R_j} $\forall$ j=1…60000
all values: V_j = 223+2*$\sqrt{\sum_{i=1}^{20} W_{ji}^2}$ * $\left|normal(0,0.25)\right|$ $\forall$ j=1…60000

Again normalization should cancel the scaling, if not normalized alignment should improve results.

MMKP_Pr8: Baseline with reduced bounds first and last constraints scaled
all bounds 70%: C_i=0.7*50*3000 $\forall$ i=2…19
C_1=0.5*0.7*50*3000, C_{20}=1.50*1.05*50*3000
all weights: R_j = uniform(-20,20) $\forall$ j=1…60000
W_{ji}=50+uniform(.8,1.2)*R_j $\forall$ i=2…19 j=1…60000
W_{j1}=0.5*{50+uniform(.8,1.2)*R_j} $\forall$ j=1…60000
W_{j20}=1.5*{50+uniform(.8,1.2)*R_j} $\forall$ j=1…60000
all values: V_j = 223+2*$\sqrt{\sum_{i=1}^{20} W_{ji}^2}$ * $\left|normal(0,0.25)\right|$ $\forall$ j=1…60000

Low bounds should cause 'low initial' to perform better. If not normalized alignment and directed algorithm would give improved results.

PROBLEM 9: *SHOW EFFECT OF CRITICAL DIMENSIONS*
MMKP_Pr9: Baseline with half of resources scaled by .6 and half by 1.4
all bounds 105%: C_i=1.05*50*3000 $\forall$ i=1…20
all weights: R_j = uniform(-20,20) $\forall$ j=1…60000
W_{ji}=0.6*{50+uniform(.8,1.2)*R_j} $\forall$ i=1…10, j=1…60000
W_{jj}=1.4*{50+uniform(.8,1.2)*R_j} $\forall$ i=11…20, j=1…60000
all values: V_j = 223+2*$\sqrt{\sum_{i=1}^{20} W_{ji}^2}$ * $\left|normal(0,0.25)\right|$ $\forall$ j=1…60000

There are abundant resources (those scaled by .6) and scarce resources (scaled 1.4). Normalization will have no effect. Directed algorithm might perform better.

PROBLEM 10-13: *SHOW EFFECT OF DIFFERENT VALUE CALCULATION*

MMKP_Pr10: Baseline with bounds scaled to 90% and value based on sum of resources
all bounds 90%: C_i=0.9*50*3000 $\forall$ i=1…20
all weights: R_j = uniform(-20,20) $\forall$ j=1…60000
W_{ji}=50+uniform(.5,1.5)*R_j $\forall$ i=1…20 j=1…60000
all values: V_j = 50+ $\sum_{i=1}^{20} W_{ji}$ ****** $\left|normal(0,0.25)\right|$ ***$\forall$ j=1…60000***

Tests effect of a value depending on individual resources. Directed and alignment expected to perform more poorly.

MMKP_Pr11: Same as 10 however last 10 resources add more to value, scaling bigger
all bounds 90%: C_i=0.9*50*3000 $\forall$ i=1…20
all weights: R_j = uniform(-20,20) $\forall$ j=1…60000
W_{ji}=50+uniform(.5,1.5)*R_j $\forall$ i=1…20 j=1…60000
all values: V_j = 50+ $\left\{\sum_{i=1}^{10} W_{ji} + 10 * \sum_{i=11}^{20} W_{ji}\right\}$ ****** $\left|normal(0,0.25)\right|$ ***$\forall$ j=1…60000***

Tests effect of a value depending on individual resources. Directed and alignment expected to perform more poorly.

MMKP_Pr12: Baseline with bounds scaled to 70% and value random
all bounds 70%: C_i=0.7*50*3000 $\forall$ i=1…20
all weights: R_j = uniform(-20,20) $\forall$ j=1…60000
W_{ji}=50+uniform(.5,1.5)*R_j $\forall$ i=1…20 j=1…60000
all values: V_j = 223+uniform(-100,100) $\forall$ j=1…60000

Tests the effect of a random value combined with few resources.

MMKP_Pr13: Same as problem 12, value also based on resources
all bounds 70%: C_i=0.7*50*3000 $\forall$ i=1…20
all weights: R_j = uniform(-20,20) $\forall$ j=1…60000
W_{ji}=50+uniform(.5,1.5)*R_j $\forall$ i=1…20 j=1…60000
all values: V_j = 223+ uniform(-100,100)+2*$\sqrt{\sum_{i=1}^{20} W_{ji}^{\,2}}$ ****** $\left|normal(0,0.25)\right|$ ***$\forall$ j=1…60000***

Tests the effect of a random value combined with few resources, and resource influence on value.

PROBLEMS 14-15: *SHOW EFFECT OF RESSOURCE CONSUMPTION CORRELATION BEEING CLUSTERED AND CLUSTERS WITH DIFFERENT CHARACTERISTICS*

MMKP_Pr14: Resources consumption correlates in clusters
all bounds 80%: C_i=0.8*50*3000 $\forall$ i=1…20
all weights: R_{jk} = uniform(-45,45) $\forall$ j=1…60000, k=1…4
W_{ji}=50+uniform(.9,1.1)*R_{j1} $\forall$ i=1…5 j=1…60000
W_{ji}=50+uniform(.9,1.1)*R_{j2} $\forall$ i=6…10 j=1…60000
W_{ji}=50+uniform(.9,1.1)*R_{j3} $\forall$ i=11…15 j=1…60000
W_{ji}=50+uniform(.9,1.1)*R_{j4} $\forall$ i=16…20 j=1…60000

all values: V_j = 223+2* $\sqrt{\sum_{i=1}^{20} W_{ji}^{\,2}}$ * $\left|normal(0,0.25)\right|$ $\forall$ j=1…60000

Tests the effect of having resource consumption for each alternative correlate around 4 clusters.

MMKP_Pr15: Resources consumption correlates in clusters
all bounds 100%: C_i=100*3000 $\forall$ i=1…5
C_i=.75*100*3000 $\forall$ i=6…10
C_i=1.25*40*3000 $\forall$ i=11…15
C_i=40*3000 $\forall$ i=16…20
all weights: R_{j1} = uniform(-40,40) $\forall$ j=1…60000
R_{j2} = uniform(-44.4,44.4) $\forall$ j=1…60000
R_{j3} = uniform(-16,16) $\forall$ j=1…60000
R_{j4} = uniform(-16,16) $\forall$ j=1…60000
W_{ji}=100+uniform(.9,1.1)*R_{j1} $\forall$ i=1…5 j=1…60000
W_{ji}=100+uniform(.9,1.1)*R_{j2} $\forall$ i=6…10 j=1…60000
W_{ji}=40+uniform(.9,1.1)*R_{j3} $\forall$ i=11…15 j=1…60000
W_{ji}=40+uniform(.9,1.1)*R_{j4} $\forall$ i=16…20 j=1…60000

all values: V_j = $\left\{\sum_{i=1}^{5} W_{ji} + 4*\sum_{i=6}^{10} W_{ji} + 1.5*\sum_{i=11}^{15} W_{ji} + 2.5*\sum_{i=16}^{20} W_{ji}\right\}$ *uniform(0,1) $\forall$ j=1…60000

Tests the effect of having resource consumption for each alternative correlate around 4 clusters and have these clusters be of a resource that is more or less abundant and that impacts value differently.

The results are given in the following tables (see comment above table 5 in appendix 3 for description of tables):

Moser	1413938.193					
MMKP_Pr1_NC	***Low Res (NA-A)***	***High Val (NA-A)***	***Low Res (NA-A)***	***High Val (NA-A)***	***Eq Low res***	***Swap***
Generic	1413934 - 1413934	1413938 - 1413938	1413934 - 1413934	1413938 - 1413938	1413935	1413938
Delta	1413934 - 1413933	1413938 - 1413938	1409312 - 1413042	1413938 - 1413938	1411512	1413938
Directed	1413933	1413938	1412782	1413938	1413215	1413938
Moser	1406289.124					
MMKP_Pr1_C	***Low Res (NA-A)***	***High Val (NA-A)***	***Low Res (NA-A)***	***High Val (NA-A)***	***Eq Low res***	***Swap***
Generic	1386215 - 1386205	1376924 - 1377003	1386215 - 1386205	1376924 - 1377003	1386116	1377980
Delta	1403738 - 1415532	***1421245 - 1421131***	1405339 - 1416173	***1421245 - 1421131***	1415810	***1421353***
Directed	1382247	***1421155***	1394919	***1421155***	1396853	***1421271***
Moser	0					
MMKP_Pr2_NC	***Low Res (NA-A)***	***High Val (NA-A)***	***Low Res (NA-A)***	***High Val (NA-A)***	***Eq Low res***	***Swap***
Generic	0 - 0	0 - 0	0 - 0	0 - 0	0	0
Delta	0 - 0	0 - 0	0 - 0	0 - 0	0	0
Directed	0	0	0	0	0	0
Moser	0					
MMKP_Pr2_C	***Low Res (NA-A)***	***High Val (NA-A)***	***Low Res (NA-A)***	***High Val (NA-A)***	***Eq Low res***	***Swap***
Generic	1299911 - 1299924	1266849 - 1267057	1299884 - 1299935	1266849 - 1267057	1300186	1268706
Delta	1305643 - 1300298	***1313567 - 1314252***	1305037 - 1299232	***1313567 - 1314252***	1295350	***1314497***
Directed	1299767	***1311028***	1292463	***1310902***	1286354	***1311066***
Moser	0					
MMKP_Pr3_NC	***Low Res (NA-A)***	***High Val (NA-A)***	***Low Res (NA-A)***	***High Val (NA-A)***	***Eq Low res***	***Swap***
Generic	0 - 0	0 - 0	0 - 0	0 - 0	0	0
Delta	0 - 0	0 - 0	0 - 0	0 - 0	0	0
Directed	0	0	0	0	0	0
Moser	0					
MMKP_Pr3_C	***Low Res (NA-A)***	***High Val (NA-A)***	***Low Res (NA-A)***	***High Val (NA-A)***	***Eq Low res***	***Swap***
Generic	1037966 - 1039292	1028895 - 1028184	1038953 - 1039208	1028895 - 1028184	1042492	1033483
Delta	***1061281 - 1055858***	1039924 - 1040489	***1060948 - 1055151***	1039924 - 1040629	***1050137***	1044152
Directed	1042189	1032526	1038100	1032442	1028185	1033433

Table 6: Problems 1-3; Changing availability of resources and inter-dimensional correlation

Moser	1440235.714					
MMKP_Pr4	***Low Res (NA-A)***	***High Val (NA-A)***	***Low Res (NA-A)***	***High Val (NA-A)***	***Eq Low res***	***Swap***
Generic	1425862 - 1425866	1416867 - 1416986	1425812 - 1426128	1417439 - 1417865	1424182	1418683
Delta	***1441522 - 1452310***	***1458369 - 1458710***	***1443524 - 1453550***	***1458408 - 1458649***	1452857	***1458742***
Directed	1416563	***1458417***	1435596	***1458228***	1433222	***1458256***
Moser	0					
MMKP_Pr5	***Low Res (NA-A)***	***High Val (NA-A)***	***Low Res (NA-A)***	***High Val (NA-A)***	***Eq Low res***	***Swap***
Generic	***1228974 - 1230156***	1191707 - 1193600	***1228010 - 1227707***	1192469 - 1191939	***1228960***	1199217
Delta	***1225548 - 1206271***	1191752 - 1200056	***1227984 - 1208510***	1191328 - 1198333	***1194260***	1203399
Directed	1178530	1171806	1180024	1157102	1161502	1158318
Moser	1203137.472					
MMKP_Pr6	***Low Res (NA-A)***	***High Val (NA-A)***	***Low Res (NA-A)***	***High Val (NA-A)***	***Eq Low res***	***Swap***
Generic	***1213484 - 1213707***	1179445 - 1180010	***1213704 - 1213933***	1179473 - 1178598	***1214934***	1180565
Delta	***1187606 - 1211979***	1182392 - 1184831	***1190870 - 1214399***	1182944 - 1185498	***1212997***	1186020
Directed	1167133	1180889	1183155	1167037	1177558	1167203
Moser	1421576.909					
MMKP_Pr7	***Low Res (NA-A)***	***High Val (NA-A)***	***Low Res (NA-A)***	***High Val (NA-A)***	***Eq Low res***	***Swap***
Generic	1396376 - 1396529	1393267 - 1393588	1396700 - 1396539	1393430 - 1393731	1395388	1394795
Delta	***1417505 - 1429135***	***1435894 - 1435880***	***1420820 - 1430477***	***1435953 - 1435683***	***1430685***	***1436099***
Directed	1391304	***1435831***	1410476	***1435729***	1408218	***1435752***
Moser	1053425.441					
MMKP_Pr8	***Low Res (NA-A)***	***High Val (NA-A)***	***Low Res (NA-A)***	***High Val (NA-A)***	***Eq Low res***	***Swap***
Generic	1043727 - 1043501	1034085 - 1033543	1042829 - 1042963	1035328 - 1034912	1047431	1040562
Delta	***1064983 - 1058568***	1047752 - 1047936	***1065837 - 1059373***	1047913 - 1048360	***1052951***	***1050746***
Directed	1048667	1042032	1043357	1041944	1029792	1042545

Table 7: Problems 4-8; looking at scaling bounds and constraints for various dimensions

Moser	0					
MMKP_Pr9	***Low Res (NA-A)***	***High Val (NA-A)***	***Low Res (NA-A)***	***High Val (NA-A)***	***Eq Low res***	***Swap***
Generic	***1152282 - 1152800***	1128955 - 1129405	***1152142 - 1152933***	1128955 - 1129405	1147616	1132237
Delta	***1149273 - 1157105***	1111304 - 983849	***1149981 - 1157690***	1111304 - 1049964	***1157246***	1086819
Directed	1132166	1111568	1132428	1111568	1131313	1111798
Moser	0					
MMKP_Pr10	***Low Res (NA-A)***	***High Val (NA-A)***	***Low Res (NA-A)***	***High Val (NA-A)***	***Eq Low res***	***Swap***
Generic	1542121 - 1541136	1473031 - 1471723	1543213 - 1542789	1473031 - 1471723	1550297	1484827
Delta	***1557316 - 1560153***	***1577919 - 1584087***	***1552837 - 1570597***	***1577919 - 1584087***	1563946	***1587445***
Directed	1517577	1555902	1457748	1556007	1427817	1556467
Moser	0					
MMKP_Pr11	***Low Res (NA-A)***	***High Val (NA-A)***	***Low Res (NA-A)***	***High Val (NA-A)***	***Eq Low res***	***Swap***
Generic	7771584 - 7740173	7369355 - 7365174	7762817 - 7748863	7369355 - 7365174	7739749	7455636
Delta	***7855603 - 7886125***	***7998559 - 7975699***	***7833307 - 7961261***	***7998559 - 7975699***	7893529	***8020345***
Directed	7692534	7877318	7219255	7883171	7171859	7884905
Moser	871749.7313					
MMKP_Pr12	***Low Res (NA-A)***	***High Val (NA-A)***	***Low Res (NA-A)***	***High Val (NA-A)***	***Eq Low res***	***Swap***
Generic	***860412 - 860194***	***859090 - 860077***	***861151 - 860137***	***859090 - 860077***	***862925***	***865504***
Delta	***870240 - 849786***	842888 - 838544	***870093 - 847089***	842858 - 838884	822389	848038
Directed	857491	843092	841250	841992	808342	842914
Moser	0					
MMKP_Pr13	***Low Res (NA-A)***	***High Val (NA-A)***	***Low Res (NA-A)***	***High Val (NA-A)***	***Eq Low res***	***Swap***
Generic	1091689 - 1090459	1097544 - 1095554	1091502 - 1089588	1097544 - 1095554	1093551	1105538
Delta	***1133843 - 1085783***	***1108610 - 1109701***	***1135614 - 1082957***	***1108610 - 1110481***	1051632	***1116001***
Directed	1092814	1089325	1064261	1089533	1028705	1090277
Moser	1285135.306					
MMKP_Pr14	***Low Res (NA-A)***	***High Val (NA-A)***	***Low Res (NA-A)***	***High Val (NA-A)***	***Eq Low res***	***Swap***
Generic	***1272451 - 1240377***	***1202779 - 1190955***	***1272451 - 1240377***	***1202779 - 1190955***	***1220997***	***1203874***
Delta	***1258587 - 1198449***	1177596 - 1161482	***1275092 - 1202368***	1177547 - 1162407	1067471	***1236544***
Directed	1111511	1161675	1125501	1164466	1027312	1165346
Moser	0					
MMKP_Pr15	***Low Res (NA-A)***	***High Val (NA-A)***	***Low Res (NA-A)***	***High Val (NA-A)***	***Eq Low res***	***Swap***
Generic	6617716 - 6597495	4889084 - 4977779	6720460 - 6764608	4726108 - 4717248	6122345	6935437
Delta	***6819707 - 6664098***	***5058538 - 3977367***	***6703089 - 6904065***	***4588726 - 4177190***	6415837	***6925046***
Directed	6581668	6880653	6366215	7300754	5903331	7325639

Table 8: Pr. 9 different dimensions criticality; Pr. 10-13 Changing value function relation with resources; Pr. 14-15 resources dimension correlate in four clusters

Appendix 7: Evaluating Literature Problems

Cplex/MRLS/Moser	364	364	315 (291*)	Worst/Best %:	72.5%	88.5%
I2	**Low Res (NA-A)**	**High Val (NA-A)**	**Low Res (NA-A)**	**High Val (NA-A)**	**Eq Low res**	**Swap**
Generic	318 - 316	303 - 303	318 - 316	303 - 303	332	**322**
Delta	**339 - 339**	275 - 263	**339 - 339**	275 - 263	337	302
Directed	294	**338**	284	316	287	**339**
Cplex/MRLS/Moser	1602	1602	1437 (1464*)	Worst/Best %:	73.2%	94.1%
I3	**Low Res (NA-A)**	**High Val (NA-A)**	**Low Res (NA-A)**	**High Val (NA-A)**	**Eq Low res**	**Swap**
Generic	**1508 - 1517**	1354 - 1226	**1508 - 1529**	1323 - 1226	1465	1476
Delta	1453 - 1417	1274 - 1059	1471 - 1405	1274 - 1112	1423	1422
Directed	1394	1306	1437	1306	1401	1393
Cplex/MRLS/Moser	3597	3597	3345 (3375*)	Worst/Best %:	81.3%	92.4%
I4	**Low Res (NA-A)**	**High Val (NA-A)**	**Low Res (NA-A)**	**High Val (NA-A)**	**Eq Low res**	**Swap**
Generic	3326 - 3405	2827 - 2914	**3338 - 3338**	2827 - 2826	3216	**3324**
Delta	3239 - 3379	3029 - 3106	3195 - 3271	3013 - 3105	3253	3254
Directed	3280	0	3315	0	3273	0
Cplex/MRLS/Moser	3906	3906	3905 (3905*)	Worst/Best %:	85.0%	100.0%
I5	**Low Res (NA-A)**	**High Val (NA-A)**	**Low Res (NA-A)**	**High Val (NA-A)**	**Eq Low res**	**Swap**
Generic	3882 - 3824	3798 - 3835	3882 - 3787	3749 - 3835	3843	3835
Delta	3817 - 3681	3905 - 3693	3816 - 3684	**3905 - 3479**	3671	3718
Directed	3495	3905	3322	**3905**	3751	**3905**
Cplex/MRLS/Moser	4799	4799	4733 (4115*)	Worst/Best %:	84.1%	99.8%
I6	**Low Res (NA-A)**	**High Val (NA-A)**	**Low Res (NA-A)**	**High Val (NA-A)**	**Eq Low res**	**Swap**
Generic	4742 - 4697	4526 - 4515	4742 - 4697	4526 - 4499	4521	4747
Delta	4671 - 4239	**4786 - 2906**	4671 - 4176	**4786 - 2464**	4245	4158
Directed	4217	**4786**	4038	**4786**	4058	**4788**
Cplex/MRLS/Moser	24584	24587	23537 (23556*)	Worst/Best %:	75.7%	94.9%
I7	**Low Res (NA-A)**	**High Val (NA-A)**	**Low Res (NA-A)**	**High Val (NA-A)**	**Eq Low res**	**Swap**
Generic	23258 - 23403	19812 - 20362	23320 - 23403	19812 - 20362	**23328**	**23642**
Delta	23287 - 23546	19744 - 22077	**23440 - 23393**	19640 - 21321	**23483**	**23496**
Directed	22517	22640	23196	22623	**23405**	**23454**
Cplex/MRLS/Moser	37869	36877	35299 (35373*)	Worst/Best %:	73.7%	93.5%
I8	**Low Res (NA-A)**	**High Val (NA-A)**	**Low Res (NA-A)**	**High Val (NA-A)**	**Eq Low res**	**Swap**
Generic	**35384 - 35219**	30279 - 30968	**35384 - 35219**	30279 - 30968	**35324**	**35258**
Delta	34758 - 35231	30832 - 32394	35274 - 35183	30832 - 32921	35137	**35353**
Directed	33340	34188	35058	34248	**35418**	**35326**
Cplex/MRLS/Moser	49155	49167	46970 (47205*)	Worst/Best %:	76.8%	95.8%
I9	**Low Res (NA-A)**	**High Val (NA-A)**	**Low Res (NA-A)**	**High Val (NA-A)**	**Eq Low res**	**Swap**
Generic	46796 - 47367	40279 - 41663	46861 - 47316	40279 - 41663	**47072**	**47157**
Delta	46878 - 46946	41230 - 45744	46878 - 47049	41400 - 45129	46779	**47054**
Directed	45300	45678	46677	46135	46882	**47034**

Table 9: Literature I2-I9 (* denotes Moser solutions in (Hifi et al., 2004)) (see comments table 5)

For the literature problems INST1-INST20 only best and worst performance are shown. Only the following set of algorithms was considered for solving the problems:

- *Delta-Low-A:* Delta heuristic (see section 2) with low resource initial solution, normalized constraints, and alignment penalty in effect
- *Delta-Low-NA:* Delta heuristic (see section 2) with low resource initial solution, normalized constraints, and no alignment penalty in effect
- *Generic-Low-A:* Generic heuristic (see section 2) with low resource initial solution, normalized constraints, and alignment penalty in effect
- *Generic-Low-NA:* Generic heuristic (see section 2) with low resource initial solution, normalized constraints, and no alignment penalty in effect
- *Delta-Swap:* Swap heuristic for delta (see section 2), constraints are normalized alignment penalty is used, initial is high value than solution is used as low resource initial
- *Generic-Swap:* Swap heuristic for generic (see section 2), constraints are normalized alignment penalty is used, initial is high value than solution is used as low resource initial

Prob/Res	***CPLEX***	***MRLS***	***MOSER***	***Best***	***Algo***	***percent***	***Worst***	***Algo***	***percent***
INST1	10709	10714	10104	10133	Delta-Low-A	94.58%	9992	Delta-Swap	93.26%
INST2	13597	13598	13074	13097	Delta-Low-A	96.32%	12772	Delta-Low-NA	93.93%
INST3	10934	10943	0	10393	Generic-Swap	94.97%	10319	Delta-Swap	94.30%
INST4	14422	14499	13626	13571	Generic-Swap	93.60%	13522	Delta-Swap	93.26%
INST5	17041	17053	16171	16238	Delta-Swap	95.22%	16005	Delta-Low-NA	93.85%
INST6	16815	16823	16049	16089	Delta-Low-A	95.64%	15687	Generic-Swap	93.25%
INST7	16407	16423	15503	15582	Delta-Swap	94.88%	15406	Delta-Low-NA	93.81%
INST8	17484	17506	16611	16624	Generic-Low-NA	94.96%	16472	Delta-Low-NA	94.09%
INST9	17747	17754	16703	16931	Generic-Low-A	95.36%	16484	Delta-Low-NA	92.85%
INST10	19285	19314	18326	18440	Generic-Low-NA	95.47%	18222	Delta-Low-A	94.35%
INST11	19424	19431	18252	18489	Delta-Low-NA	95.15%	18320	Generic-Swap	94.28%
INST12	21725	21730	20419	20653	Generic-Swap	95.04%	20384	Delta-Swap	93.81%
INST13	21569	21569	20348	21458	Delta-Swap	99.49%	20483	Generic-Swap	94.96%
INST14	32866	32869	31115	32526	Delta-Swap	98.96%	31147	Generic-Swap	94.76%
INST15	39154	39148	37335	38789	Delta-Swap	99.08%	37046	Generic-Swap	94.63%
INST16	43357	43354	41509	43000	Delta-Swap	99.18%	41091	Generic-Swap	94.78%
INST17	54349	54349	52104	53922	Delta-Swap	99.21%	51599	Generic-Swap	94.94%
INST18	60455	60456	58617	59949	Delta-Swap	99.16%	57637	Generic-Swap	95.34%
INST19	64919	64921	62775	64397	Delta-Low-NA	99.19%	61804	Generic-Swap	95.20%
INST20	75603	75603	73638	74955	Delta-Low-NA	99.14%	72123	Generic-Swap	95.40%
Average	---	---	---	---	---	96.73%	---	---	94.25%
Max	---	---	---	---	---	99.49%	---	---	95.40%
Min	---	---	---	---	---	93.60%	---	---	92.85%

Table 10: Best worst performance of selected heuristics on literature problems INST1-INST20

Appendix 8: Matlab .m files brief description

MMKP_get.m:

Loads variables of MMKP problem, either a predefined MMKP problem such as the test problem, or a problem recorded in an xls sheet in predefined format.

MMKP_probGen.m:

Generates MMKP problems and records them in predefined .xls format, previous version of the files are deleted, problems to be generated can be chosen.

MMKP_solve.m:

Given a MMKP problem, attempts to solve it either using the created DLL or Matlab's linear programming modules or the CPLEX interface

MMKP_swap.m:

Uses MMKP_solve to first get a solution for high value algorithm; then uses this solution as the initial solution for the low resource algorithm. The algorithm can be generic, delta or aligned.

MMKP_allAlgo.m:

Uses *MMKP_solve* repeatedly for all algorithms and their combinations with initial solutions, penalty and normalization settings.

MMKP_allCnstrSets.m:

Uses *MMKP_allAlgo* repeatedly for all given bound sets

MMKP_allCnstrSetsEq.m:

For all bound sets uses *MMKP_solve* to get solutions for the equalized low resource initial solution.

MMKP_allLitProb.m:

Uses *MMKP_allAlgo* repeatedly on different sets of MMKP from literature.

MMKP_allMMKPProb.m:

Uses *MMKP_allAlgo* repeatedly on the chosen generated problems.

MMKP_debug_lists.m:

If dll currently loaded with a problem, uses debug interface to get for the first five groups the first five elements. Does this for both the problem list and the temporary list. If the dll is modified to return after a number of iteration in finding a solution this can be used to see where it has gotten.

MMKP_getDataPLUSexcell.m:

Reads excel tables for the Dutch retailer case.

MMKP_parseDataPLUSdll.m:

Parses the data from *MMKP_getDataPLUSexcell* to the MMKP problem format.

MMKP_solvePLUSPr.m:

Solves problem obtained with *MMKP_parseDataPLUSdll*

MMKP_solvePLUSPrSet.m:

Solves a set of PLUS MMKP problems with different deliveries and schedules

MMKP_out.m, MMKP_out_Lit.m, MMKP_out_Prob.m, , MMKP_out_Sol.m::

Output methods to record results to xls files

readProbData.m, saveProb.m:

Used to read and convert problems from literature in text format

test2_createProb.m:

Used to record a MMKP problem to an xls sheet in predefined format

Appendix 9: Van Buel model [Buel 2009]

4.5.1 Store Operations Model

Consider a single stockpoint (shelf location) where a single SKU is stocked. Time is divided into periods of equal length; the length of each period is assumed to be one weekday. The review period R is determined by the delivery schedule. The goal of the store operations model is to find the number of planned replenishment actions from the backroom to the shop floor for a SKU and the number of order lines and quantities per weekday given a delivery schedule.

The number of planned replenishment moments per weekday does not have to be calculated for the products with excess shelf space. For these products, the number of idle replenishments can be set to zero, which in the clustering step will lead to an allocation to a low delivery frequency schedule. The excess shelf space formula (Broekmeulen et al., 2004) represents the difference between the shelf capacity of a product and its maximum on hand inventory:

(3) $E_i = V_i - S_i$

(4) $S_i = s_i + Q_i - 1$

(5) $s_i = MIN_i + \lambda * \sigma_i + \mu_i * MAX(L+R)$

Where:

i = SKU 1, …., n

E_i = Excess shelf space;

V_i = Shelf capacity;

S_i = Maximum on hand inventory;

s_i = Reorder level;

MIN_i = Minimal inventory level (commercial considerations);

λ = Safety factor as set in the ASO system;

σ_i = Standard deviation of the weekly demand (safety stocks are calculated in the ASO system based on the review period);

μ_i = Average of weekly demand;

MAX (L+R) = Maximum leadtime plus review period;

Q_i = Case pack size.

Since the ASO system of PLUS controls inventory of SKU i with a (R,s,nQ) policy, a replenishment order is created only when the inventory position at a review moment is strictly below reorder level s_i. The value of n is chosen such that the inventory position just after a replenishment decision is at or above s, but strictly less than $s_i + Q_i$. So the maximum inventory position just after a replenishment decision is $s_i + Q_i - 1$. In case no demand occurs for a product from the moment of ordering to the delivery moment,

> the expected sales during the lead time and the review period have to fit together on the shelf. In case of PLUS Retail, the maximum review period is equal to six days (i.e. one delivery to the store per week). When operating with a maximum review period, a maximum safety stock has to fit also on the shelf ($\lambda*\sigma_i$) in the case of the ASO system of PLUS). Besides this, the minimal inventory level has to be always on the shelf due to commercial considerations. When the shelf capacity is equal or larger than the

maximum on hand inventory, no leftovers can exist for that product on the condition that the instore process is well coordinated. With the below mentioned formulas, the on hand inventories and replenishment moments can be calculated for the SKU's given a delivery schedule.

On hand inventory in the building (store + backroom):
(5) $X_{it} = Y_{i(t-1)} + PU_{(it-L)} * Q_i$
(6) $Sales_{it} = \min(X_{it}, D_{it})$
(7) $Y_{it} = X_{it} - Sales_{it}$
Where:
X_{it} = Inventory on hand in the building at the beginning of day t
Y_{it} = Inventory on hand in the building at the end of selling day t, but before an external delivery is added
PU_{it} = Order advice in case packs generated by the ASO system for SKU i in period t
L = Lead time
$Sales_{it}$ = Sold number of products of SKU i during period t
D_{it} = Demand for SKU i during period t

Shelf inventory:
(8) $SX_{it} = \min(X_{it};V_i)$
Where:
SX_{it} = Shelf inventory at the beginning of period t

Backroom inventory:
(9) $BX_{it} = X_{it} – SX_{it}$
Where:
BX_{it} = Inventory on hand of SKU i in the backroom at the beginning of selling day t

Concurrent replenishments:
(10) $ConRep_{it} = Sales_{it} / V_i$
Where:
$ConRep_{it}$ = The number of concurrent replenishments during period t

End backroom inventory:
(11) $BY_{it} = \max(BX_{it} – ConRep_{it} * V_i; 0)$
Where:
BY_{it} = Inventory on hand of SKU i in the backroom at the end of a selling day t, but before an external delivery is added

Output
Planned replenishments can only be executed once a day per SKU. A replenishment action takes place when the backroom inventory for a SKU after a possible external delivery is positive:
(12) $BY_{it} + PU_{it-L}*Q_i > 0$
The delivered quantity in consumer units per SKU on a delivery moment is equal to:
(13) $DCU_{it} = PU_{it-L}*Q_i$
A SKU is delivered to the store (order line) on a delivery day when:
(14) $DCU_{it} > 0$
Where:
DCU_{it} = The delivered number of consumer units of SKU i on day t

The order quantities per product and the number of order lines given a delivery schedule are generated by the ASO system of PLUS as presented in Appendix K.

4.5.2 Clustering

As stated in the model description, the modelled planned replenishments by the Store Operations Model have to be translated into idle replenishments, which are the instore cost driver in this research. To come to the number of idle replenishments, the average number of order lines per SKU per week should be subtracted from the average number of planned replenishment moments per SKU per week given a delivery schedule. The idle replenishment costs can be calculated by multiplying the cost for an idle trip with the number of idle replenishments per week:

(15) $C_{ij}^{H} = (H_{ij} - OL_{ij}) * C^{idle}$

Where:

C_{ij}^{H} = Idle replenishment costs for SKU i when delivered according to schedule j
H_{ij} = Average weekly number of planned replenishments
OL_{ij} = Average weekly number of order lines
C^{idle} = Costs for an idle trip from the backroom to the shop floor

The clustering step is built up out of different modules to come ultimately to clusters of products with the same delivery schedule. With a given set of schedules (63 possible delivery schedules), this problem can be classified as a multidimensional multiple-choice knapsack problem (Akbar et al., 2006).

Objective function:

$$(16)\ \min\left\{\sum_i \sum_j C_{ij}{}^{h} x_{ij} + C_{drop} \sum_d z_d\right\}$$

Subject to:

1. Each SKU is assigned to exactly one schedule:

$$(17)\ \forall i: \sum_j x_{ij} = 1$$

2. Each SKU can only be assigned to active schedules:

$$(18)\ \forall j: \sum_j x_{ij} < BigM * y_j$$

3. Maximum number of active schedules A (clusters):

$$(19)\ \sum_j y_j <= A$$

4. The active schedules determine the delivery days:

$$(20)\ \forall d: z_d <= \sum_j D_{jd} * y_j$$

5. The delivery volume depends on the schedule assignment and has to fit always in a truck:

$$(21)\ \forall d: \sum_i \sum_j D_{jd} * y_j$$

Where:

x_{ij} = Assignment of SKU i to schedule j {0,1}
C_{drop} = Transportation costs per drop

z_d = Base schedule (delivery on weekday d) {0,1}
y_j = Schedule j in use by store y {0,1}
A = Number of active schedules
D_{jd} = Delivery on day d according to delivery schedule j {0,1}
W = Truck capacity

The delivery volume can be calculated as follows:
(22) $V_{ijd} = OL_{ijd} * DPU_{ijd} * Vol_i$
Where:
V_{ijd} = Delivery volume on weekday d for SKU i according to delivery schedule j (m_3)
OL_{ijd} = Average number of order lines on weekday d
DPU_{ijd} = Delivered number of case packs on weekday d
Vol_i = Volume of the case pack of SKU i (m_3)

Appendix 10: Dutch case results retailer

Base Schedule	Days	Store: 199				Store: 563				Store: 729			
		generic null initial	generic swap	generic equalized	directed swap	generic null initial	generic swap	generic equalized	directed swap	generic null initial	generic swap	generic equalized	directed swap
1	000001	---	---	---	---	---	---	---	---	---	---	---	---
2	000010	---	---	---	---	---	---	---	---	---	---	---	---
3	000100	---	---	---	---	---	---	---	---	---	---	---	---
4	001000	---	---	---	---	---	---	---	---	---	---	---	---
5	010000	---	---	---	---	---	---	---	---	---	---	---	---
6	100000	---	---	---	---	---	---	---	---	---	---	---	---
7	000011	0	0	0	0	0	0	0	0	10057	10885	9989	9571
8	000101	0	0	0	0	0	0	0	0	7789	7777	7758	*7545*
9	000110	0	0	0	0	0	0	0	0	9834	10697	9764	9350
10	001001	0	0	0	0	0	0	0	0	7331	7326	7303	*7117*
11	001010	0	0	0	0	0	0	0	0	8014	8013	7994	*7759*
12	001100	0	0	0	0	0	0	0	0	10108	10963	10082	9627
13	010001	0	0	0	0	0	0	0	0	8781	8841	8755	8478
14	010010	0	0	0	0	0	0	0	0	7118	7117	7106	*6933*
15	010100	0	0	0	0	0	0	0	0	8477	8485	8451	8172
16	011000	0	0	0	0	0	0	0	0	10426	11152	10367	9893
17	100001	0	0	0	0	0	0	0	0	10579	11125	10546	10086
18	100010	0	0	0	0	0	0	0	0	8234	8236	8218	7960
19	100100	0	0	0	0	0	0	0	0	7192	7197	7170	*6996*
20	101000	0	0	0	0	0	0	0	0	8705	10532	8673	8383
21	110000	0	0	0	0	0	0	0	0	10524	11274	10476	10012
22	000111	0	0	0	0	10297	10647	10230	9729	7918	7921	7864	7353
23	001011	0	0	0	0	8844	8845	8784	8118	6655	6658	6603	5992
24	001101	0	0	0	0	8978	8984	8853	8204	6823	6832	6715	6097
25	001110	0	0	0	0	10492	10445	10517	9912	8247	8245	8187	7629
26	010011	0	0	0	0	9207	9337	9170	8870	7087	7091	6882	6276
27	010101	0	0	0	0	7983	7977	7945	*7493*	5874	5872	5832	*5394*
28	010110	0	0	0	0	8858	8928	8815	*8209*	6920	6934	6860	*6261*
29	011001	0	0	0	0	9802	10052	9778	9252	7206	7253	7286	6795
30	011010	0	0	0	0	9225	8809	8950	8414	7134	7156	7011	6381
31	011100	0	0	0	0	11062	11469	11287	10479	8741	8748	8721	8168
32	100011	0	0	0	0	11226	11459	11211	10564	8502	8503	8399	7905
33	100101	0	0	0	0	8972	9060	8910	8385	6892	6906	6807	6128
34	100110	0	0	0	0	8970	8976	8749	8104	6787	6794	6607	5972
35	101001	0	0	0	0	9648	12014	9752	9430	7205	7255	7323	6777
36	101010	0	0	0	0	7797	7797	7764	*7346*	5736	5736	5706	*5294*
37	101100	0	0	0	0	9586	9546	9458	8852	7162	7192	7148	6625
38	110001	0	0	0	0	11656	12557	11895	11160	8755	8756	8883	8481
39	110010	0	0	0	0	9358	9373	9348	8782	6874	6922	6883	6358
40	110100	0	0	0	0	9470	11060	9509	9099	7374	7382	7257	6650
41	111000	0	0	0	0	11400	11749	11739	10827	8961	8964	8959	8402
42	001111	10175	0	10891	9617	8482	8485	8478	7570	6476	6482	6474	5685
43	010111	8498	0	8539	*7837*	7467	7469	7428	*6568*	5611	5613	5547	*4785*
44	011011	9177	9576	9464	*8617*	7798	7800	7781	*6958*	5748	5752	5730	*5002*
45	011101	9600	11482	10397	9373	8115	8116	8060	7277	6065	6068	6025	5279
46	011110	10879	0	0	10837	8742	8850	8843	7945	6959	6964	6958	6133
47	100111	10449	0	10803	9751	8772	8776	8747	7894	6567	6568	6546	5785
48	101011	9093	0	10443	*8281*	7859	7868	7759	*6969*	5717	5726	5660	*4925*
49	101101	9754	0	10999	10454	8159	8160	8153	7286	6035	6034	6026	5230
50	101110	9017	9067	9114	*8397*	7743	7746	7657	*6774*	5850	5856	5794	*5000*
51	110011	11252	0	12356	10881	9322	9127	9248	8596	6918	6922	6923	6194
52	110101	9584	0	12085	0	7874	8004	7959	7352	5902	5917	5907	5307
53	110110	9146	0	9670	9133	7720	7727	7700	6872	5788	5801	5762	5032
54	111001	12002	0	13898	12104	9651	9684	9665	9172	7149	7176	7146	6712
55	111010	9422	0	0	9784	8029	8034	8001	7142	5965	5973	5948	5214
56	111100	11678	0	0	11392	9460	9465	9507	8659	7327	7336	7327	6583
57	011111	8159	8159	8153	*6953*	7089	7090	7082	*6099*	5359	5360	5358	*4510*
58	101111	8261	8265	8245	*6974*	7167	7167	7156	*6170*	5302	5302	5298	*4432*
59	110111	8176	8307	8186	*7013*	7094	7102	7106	*6233*	5202	5209	5208	*4445*
60	111011	8731	9428	9005	8118	7551	7556	7548	6624	5467	5472	5460	4761
61	111101	9493	10257	10332	9187	7905	7905	7901	6988	5856	5857	5854	5079
62	111110	8630	9533	8932	7916	7407	7407	7404	6465	5648	5651	5647	4841
63	111111	7238	7237	7230	*5893*	6381	6382	6376	*5377*	4644	4645	4638	*3772*

Table 11: Replenishments for one warehouse with unlimited storage

Shop	Delivery Frequency	Base Schedule	Days	Replenish-ments	Cost	Delivery Cost	Total cost
Shop 729	2	14	010010	6933	1733	320	2053
	3	36	101010	5294	1324	480	1804
	4	43	010111	4785	1196	640	1836
	5	57	011111	4510	1128	800	1928
	6	63	111111	3772	943	960	1903
Shop 563	3	36	101010	7346	1837	480	2317
	4	43	010111	6568	1642	640	2282
	5	57	011111	6099	1525	800	2325
	6	63	111111	5377	1344	960	2304
Shop 199	4	43	010111	7837	1959	640	2599
	5	57	011111	6953	1738	800	2538
	6	63	111111	5893	1473	960	2433

Table 12: Total cost for best schedule for each shop from table 11

Base Sched.	Truck per day	Store: 199				Store: 563				Store: 729			
		Gen. null	Gen. swap	Gen. equal.	Dir. swap	Gen. null	Gen. swap	Gen. equal.	Dir. swap	Gen. null	Gen. swap	Gen. equal.	Dir. swap
14	020020	11403	11403	11432	11219	9542	9539	9524	9332	7118	7117	7106	6933
19	200200	11520	16462	11580	11481	9545	9550	9532	9332	7192	7197	7170	6996
14	020030	11403	11403	11396	11219	9542	9539	9530	9332	7118	7117	7107	6933
14	030030	11404	11402	11395	11219	9542	9539	9524	9332	7118	7117	7106	6933
36	101020	9318	10441	0	9385	7797	7797	7793	7347	5736	5736	5735	5287
36	102020	9094	9093	9092	8613	7797	7797	7796	7347	5736	5736	5735	5287
36	102030	9094	9093	9091	8613	7797	7797	7795	7347	5736	5736	5735	5287
43	010121	8498	0	8462	7835	7467	7469	7413	6568	5611	5613	5544	4780
43	010212	8437	10081	8399	7825	7467	7469	7452	6569	5611	5613	5594	4779
57	011121	8159	8159	8136	6934	7089	7090	7074	6082	5359	5360	5349	4480

Table 13: Replenishments if more than one truck per day

Base Sched.	Truck per day	Store: 199				Store: 563				Store: 729			
		Gen. null	Gen. swap	Gen. equal.	Dir. swap	Gen. null	Gen. swap	Gen. equal.	Dir. swap	Gen. null	Gen. swap	Gen. equal.	directed swap
2	000060	17142	17142	0	17142	13783	13783	0	13783	11256	11256	0	11256
5	060000	17051	17051	0	17051	13630	13630	0	13630	11293	11293	0	11293
14	060060	10892	10890	10886	10738	9234	9232	9218	9050	7003	7001	6991	6825
19	600600	10972	10978	10957	10802	9217	9221	9203	9032	7072	7077	7050	6887
27	010101	0	0	0	0	7707	7699	7640	7223	5781	5779	5739	5293
36	101010	0	0	0	0	7550	7549	7508	7079	5649	5649	5618	5194
43	010111	8046	0	8002	7393	7210	7213	7167	6315	5520	5521	5455	4691
57	011111	7760	7756	7754	6538	6845	6846	6839	5852	5271	5272	5270	4418
58	101111	7847	7849	7830	6561	6918	6918	6908	5923	5208	5209	5203	4341
63	111111	6847	6847	6841	5495	6152	6153	6148	5137	4565	4566	4559	3684

Table 14: Replenishments if concurrent replenishments are ignored for different day frequency

Base Sched. reg|nat	Truck day reg.	Truck day nat.	Store: 199				Store: 563				Store: 729			
			Gen. null	Gen. swap	Gen. equal.	Dir. swap	Gen. null	Gen. swap	Gen. equal.	Dir. swap	Gen. null	Gen. swap	Gen. equal.	Dir. swap
14 |14	010010	010010	0	0	0	0	0	0	0	0	7118	7113	7112	7117
14 | 5	010010	010000	0	0	0	0	0	0	0	0	7324	7706	7668	7362
36 | 36	101010	101010	0	0	0	0	7797	7795	7783	7796	5736	5736	5735	5737
36 | 11	101010	001010	0	0	0	0	7793	8098	8094	7795	5733	5852	5801	5735
36 | 20	101010	101000	0	0	0	0	7984	8268	8360	7838	5765	5909	5960	5769
36 | 6	101010	100000	0	0	0	0	0	9447	9144	8686	6233	6431	6597	6255
43 | 43	010111	010111	8502	0	8518	8231	7467	7468	7466	7226	5611	5612	5611	5328
43 | 27	010111	010101	8570	0	8630	9045	7430	7429	7410	7232	5574	5560	5575	5327
43 | 28	010111	010110	8422	0	8627	8531	7441	7507	7424	7466	5586	5587	5594	5598
43 | 14	010111	010010	0	0	9907	8673	7400	7430	7633	7393	5561	5544	5553	5371
43 | 15	010111	010100	0	0	0	0	7714	7938	8040	7712	5612	5566	5847	5517
43 | 5	010111	010000	0	0	0	0	0	9741	8847	9490	6150	6629	6534	6602
57 | 57	011111	011111	8159	8158	8162	6921	7089	7091	7089	6086	5359	5361	5359	4497
57 | 45	011111	011101	8128	8207	8129	7273	7057	7066	7056	6246	5325	5326	5324	4574
57 | 23	011111	001011	8071	8248	8413	8170	7067	7089	7065	6409	5338	5340	5344	4708
57 | 15	011111	010100	0	0	10210	0	7399	7688	7805	7483	5386	5370	5689	5228
57 | 2	011111	000010	0	0	0	0	0	0	8143	0	5438	6042	5788	6054
63 | 63	111111	111111	7238	7238	7237	5876	6381	6382	6381	5346	4644	4646	4644	3749
63 | 43	111111	010111	7219	7329	7223	6632	6384	6372	6381	5714	4644	4642	4644	4023
63 | 29	111111	011001	8086	0	8280	8175	6455	6452	6569	6350	4597	4613	4595	3926
63 | 10	111111	001001	0	0	8882	7946	6535	6834	6991	6293	4605	4626	4848	3908
63 | 6	11111	10000	0	0	0	0	0	0	8159	0	5468	6110	5805	6101

Table 15: Replenishments given delivery days regionally for different national delivery frequencies (each extra national delivery costs 50 euro or the equivalent of 200 replenishments)

Appendix 11: Model for considering all shops and heuristic for finding a feasible solution

Equation 9 shows how the problem is complicated by the consideration that location must be the same for all the shops.

Maximize $z = \sum_{s=1}^{t}\sum_{i=1}^{k}\sum_{j=1}^{m_x} p_{ijs}x_{ijs} + \sum_{s=1}^{t}\sum_{i=1}^{k}\sum_{j=1}^{m_y} r_{ijs}y_{ijs}$

Subject to $\sum_{i=1}^{k}\sum_{j=1}^{m_x} n_{ijls}x_{ijs} \le n_{ls} \quad \forall \; l=1,n; s=1,t$

$$\sum_{i=1}^{k}\sum_{j=1}^{m_y} w_{ijsl}y_{ijs} \le w_{ls} \quad \forall \; l=1,n; s=1,t$$

$$\sum_{j=1}^{m_x} x_{ijs} = o_i \quad \forall \; i=1,k; s=1,t$$

$$\sum_{j=1}^{m_y} y_{ijs} = 1-o_i \quad \forall \; i=1,k; s=1,t$$

Variables:

$$x_{ijs}, y_{ijs}, o_i \in \{0,1\} \quad \forall \; i=1,k; j=1,m; s=1,t$$

Equation 9: The problem considering all shops

Where:

t: is the number of shops

k: is the number of items

m_x: is the number of schedules considered for regional warehouse

m_y: is the number of schedules considered for national warehouse

o_i: is 1 if item i is stored regionally and 0 if it is stored nationally

x_{ijs}: is 1 if item i selects schedule j for shop s and is stored regionally

y_{ijs}: is 1 if item i selects schedule j for shop s and is stored nationally

w_{ls}: is the bound, constraint for dimension l at store s given national storage

n_{ls}: is the bound, constraint for dimension l at store s given regional storage

w_{ijsl}: resource weight on l dimension by item i if schedule j at shop s (stored regionally)

n_{ijsl}: resource weight on l dimension by item i if schedule j at shop s (stored nationally)

p_{ijs}: value for alternative j of item i at store s given regional storage

r_{ijs}: value for alternative j of item i at store s given national storage

The generic and delta greedy heuristics used to solve the MMKP problem can be adapted to find a solution to the problem in equation 10. The directed heuristic would take very long. Since delta is not useful because aggregate difference of resource consumption is always 0 in the case study; only implementing the generic heuristic is presented below:

1. Load the initial solution.
2. For every shop consider every item twice, once as regionally stored and once as nationally stored. Each time calculate a ranking that will not change for all alternatives. Keep these alternatives sorted by their ranking.
3. Repeat until end criteria
    a. Calculate the aggregate of selecting the best improvements for an item at all stores twice: once for selecting best improvements in the regional storage alternative lists and once for selecting best improvements in the national storage alternative list.
    b. Choose not yet updated item with highest aggregate and change solution for the item at all stores concomitantly.
    c. Mark item as updated.
    d. If all items are updated mark all items as not updated.