Two-Stage Memory Allocation using AHP & Knapsack at PT Berca Hardayaperkasa

Khanis Satya¹, Indriati N. BISONO¹, Hanijanto Soewandi²

¹International Business Engineering Program, Universitas Kristen Petra, Surabaya, Indonesia
²MicroStrategy, Tysons Corner, VA, USA

Abstract
We propose to manage a (MicroStrategy) Business Intelligence Server in term of RAM allocation for its Intelligent Cubes as a two-stage resource allocation problem in which the first stage is formulated as an multi criteria problem that can be solved using Analytic Hierarchy Process (AHP) and the second stage is multiple (several) 0-1 classic Knapsack problems with the constraints that are obtained using the result from the first stage. This Approach happens to have advantage in term of computational complexity as well, it reduces from O(nM) to O(max{nj}max{Mj}) when calculated in parallel. We illustrate our proposal with a numerical example based on our experience.

Keywords: Business Intelligence Server; Analytic Hierarchy Process; Knapsack problem.

INTRODUCTION
In recent years, Business Intelligence (BI) is growing very rapidly in Indonesia. Inkwood Research predicted that between 2017 – 2022, the compound annual growth rate (CAGR) is 9.7%. Recent prediction for global BI world-wide is also expected to grow further from USD 23.1B to USD 33.3B from 2020 to 2025. Therefore, it is not surprising that many companies and organizations in Indonesia started to adopt BI technology. One of the Business Intelligence is the MicroStrategy BI enterprise software which is the subject of our research.

Each MicroStrategy Project essentially starts with a collection of lookup tables, relationship tables, and fact tables from Data Warehouse (or Datamart). These tables are then imported and from these tables a BI Architect will create a set of schema objects, i.e., Attributes (grouping of data, e.g., Item, Region, Month, etc.) and Facts (measures of interest, e.g., Cost, Profit, etc.). The Facts (together with aggregation functions/other type of calculations, e.g., Sum, Avg, Min, Max, etc.) are then used to construct Metrics (e.g., Revenue, Profit, etc.). To provide those business analysts with access to the data, the most common method is to use several Intelligent Cubes (I-Cubes) within a MicroStrategy Project. Figures 1 and 2 are very high-level pictures of MicroStrategy BI Server in the context of I-Cubes, Project, usages, and some statistics.

In Figure 1, an Intelligent Cube in a particular project is shared as a single in-memory dataset, among the different reports created by many users. Multiple reports are built that gather data from the Intelligent Cube instead of querying the data warehouse.

Corresponding author
khanissatya@gmail.com
DOI: https://doi.org/10.31098/bmss.v2i1.539

This is an open access article under the CC-BY-NC license
Two-Stage Memory Allocation using AHP & Knapsack at PT Berca Hardayaperkasa
Khanis Satya, Indriati N. Bisono, Hanijanto Soewandi

Figure 1. High-level MicroStrategy BI Server with respect to I-Cubes for a particular Project

Figure 2. Eight I-Cubes that belong to three Projects with various Status

Figure 2 illustrates the fact that in a single MicroStrategy BI Server, there could be multiple projects (in the above example, there are 3 projects) and each project will have multiple I-Cubes (2 I-Cubes belong to “Finance” project, 4 I-Cube belongs to “Human Resource Analysis” project, and 2 I-Cubes belong to “Marketing” project). Furthermore, it is worth to point out that each I-Cube will have its own size, i.e., use memory, and it can have different Status, namely: A = Active, F = File, L = Loaded (to memory). There is also “hit count” concept to illustrate how often a particular cube is being used in the past.

Now, imagine the task for a BI Administrator to manage this (MicroStrategy) BI Server. The BI Administrator is given a computer with a certain amount of memory (e.g., 32 GB, 128 GB, or several TBs in real large-scale implementation) in which (s)he needs to load multiple I-Cubes that are grouped in multiple (MicroStrategy) Projects to serve many users (business analysts) so that they can create their Reports/Dashboards. This is the problem that we consider in this paper.

We have to consider this problem as a two-stage resource allocation problem because considering all I-Cubes may lead to a situation in which some particular projects do not have any I-Cube loaded into the memory. Similarly, loading all I-Cubes from a particular important project may leave other project with very little (or even no) I-Cubes being loaded. Furthermore, there are multiple criteria that need to be considered among those projects.

LITERATURE REVIEW

This type of problem is commonly known as Resource Allocation problem – a well-known problem. In this particular case, there are two stages, i.e., the first stage is how to allocate computer memory at the Project level considering multiple factors, and then the second stage is how to distribute further those memory to load certain set of I-Cubes. Even though, numerous papers have been published for two-stage resource allocation problem, none fits well with our problem. Nonetheless, here are some that we review.

Wang et.al. (2020) presented a Mathematical Programming formulation for a problem of scheduling surgeons and his/her assistant surgeons in the context of health care as two-stage resource allocation optimization problem. Hong & Li (2020) considered the cloud resource provisioning problem and they formulated as the problem as a two-stage stochastic programming problem. This two-stage stochastic programming problem can be transformed into a deterministic integer program and solved by exact methods such as: branch & bound and cutting plane methods, or heuristic methods such as: genetic algorithm, particle swarm optimization, and hybrid
algorithms. Lin & Gen (2008) considered multi-criteria human resource allocation for solving multistage combinatorial optimization problem. They propose a multi-objective hybrid genetic algorithm (mohGA) approach based on the multi-stage decision-making model for solving combinatorial optimization problems. All the above literatures are elegant and appropriate to their own problems, but not suitable for ours.

The closest papers in term of application that we can find are: Singh & Dutta (2015) and Revathy and Sekar (2018). In the first paper, they considered AHP to solve multi criteria nature of Cloud Computing. However, their problem is just a simple single stage selection of Cloud Computing resources. The second one is equally interesting as they consider how to allocate Virtual Machines (VMs) to a particular job considering multiple criteria. They also use AHP to find out a good balance. But, again, the problem is just a single stage resource allocation.

On industrial application, Sharma & Dubey (2010) and Mohammadi et.al. (2015) are two papers that combined AHP and Knapsack to solve industrial problems. Sharma & Dubey also considered two-stage approach like ours. Their application is on carton sourcing. However, they use the weight obtained from AHP as the coefficient of the constraint in the Knapsack problem. Ours is slightly different, we will use the weight of the AHP to decide on the capacity of the knapsack. We will have to solve multiple knapsack problem, while Sharma & Dubey only need to solve one. Unfortunately, we could not find the paper by Mohammadi et.al. on a language that we can understand.

METHODOLOGY

The detail of our problem can be depicted in Table 1. We have 30 I-Cubes that are grouped into 5 MicroStrategy Projects (for privacy & security reasons of our client, we call them Project 1 – Project 5 and Cube 11 to Cube 54 respectively). The Server machine that hosts MicroStrategy I-Server has 32 GB of RAM and those 5 projects will use up 3.6 GB to load their Schema Objects. Similarly, we plan to allocate:

- 2 GB for Object cache (across 5 projects) – see Figure 3 (red box),
- 2 GB for Element cache (across 5 projects) – see Figure 3 (red box),
- 4 GB for Report & Document caches (across 5 projects) – see Figure 3 (red box), and
- 8 GB for processing/calculation.

Therefore, the total available memory will only be 12.4 GB (= 32 – 3.6 – 2 – 2 – 4 – 8) to load some out of 30 I-Cubes (notice that the sum of RAM for all 30 I-Cubes = 16053 MB > 12.4 GB). Hence, the need for an optimization. A naïve approach would be to formulate a Knapsack problem with all 30 I-Cubes and it will result in loading all I-Cubes in Project 1 and Project 5 as indicated by the solution in green in Table 1 (24 I-Cubes will be loaded and 6 I-Cubes are not loaded at the start-up of Intelligent Server).

At this point, it is important to understand that MicroStrategy I-Server has some governing rules that need to be set. Most of those governing rules are per project as shown in Figure 3 (the green box indicates that it is per project). The red box in Figure 3 shows where the Object, Element, & Report/Document (Result) caches can be set, and finally the black box indicates where the RAM allocation per project for I-Cubes can be set.

In Figure 3, the check-box option that says: “Load Intelligent Cubes on startup” is not an option that we want to do since there is NOT enough RAM to load all Cubes. Therefore, we have to selectively
choose which I-Cubes to load. Hence, our motivation to solve this problem as two-stage optimization problem.

Table 1. Thirty I-Cubes that are grouped into 5 Projects

| MicroStrategy Project | Cube Name | Size (MB) | Hit Count |
|-----------------------|-----------|-----------|-----------|
| Project 1             | Cube 11   | 408       | 271       |
|                       | Cube 12   | 694       | 383       |
|                       | Cube 13   | 625       | 475       |
|                       | Cube 14   | 360       | 431       |
|                       | Cube 21   | 412       | 23        |
|                       | Cube 22   | 951       | 273       |
|                       | Cube 23   | 639       | 30        |
|                       | Cube 24   | 667       | 309       |
|                       | Cube 25   | 811       | 181       |
|                       | Cube 26   | 870       | 258       |
|                       | Cube 31   | 586       | 157       |
|                       | Cube 32   | 398       | 331       |
|                       | Cube 33   | 380       | 12        |
|                       | Cube 34   | 526       | 125       |
|                       | Cube 35   | 383       | 171       |

| MicroStrategy Project | Cube Name | Size (MB) | Hit Count |
|-----------------------|-----------|-----------|-----------|
| Project 3             | Cube 36   | 278       | 315       |
|                       | Cube 37   | 462       | 253       |
|                       | Cube 41   | 708       | 66        |
|                       | Cube 42   | 707       | 224       |
|                       | Cube 43   | 500       | 325       |
|                       | Cube 44   | 714       | 269       |
|                       | Cube 45   | 618       | 49        |
|                       | Cube 46   | 393       | 252       |
|                       | Cube 47   | 370       | 461       |
|                       | Cube 48   | 591       | 180       |
|                       | Cube 51   | 324       | 328       |
|                       | Cube 52   | 444       | 455       |
|                       | Cube 53   | 337       | 318       |
|                       | Cube 54   | 326       | 125       |
|                       | Cube 55   | 371       | 155       |

First-Stage Multi-Criteria Problem

The first-stage problem then is clearly how to allocate 12.4 GB memory across 5 projects. For this, we will use Analytical Hierarchy Process (AHP) since there are multiple criteria that we need to consider. We skipped reviewing/explaining AHP since there are already numerous books, journal articles on this topic. Readers who are interested to learn about AHP can visit AHP Tutorial on Teknomo’s website (Teknomo, 2006). For the first-stage problem, the formulation can be presented as in Figure 5.

After talking to various managers at PT Berca Hardayaperkasa, we found out that these criteria, namely: due date of the projects, the numbers of business analysts/users for each project, numbers of objects (in particular Reports/Dashboards/Hypercards) in each project, processing speeds, and overall system performance are factors that everybody wants to have. It is important to point out that three of these criteria, e.g., Due Date, Perceived Response Time of Dashboards, and Perceived Response Time of the System (Browsing, etc.) are subjective (or qualitative) in nature. The other two criteria, i.e., Number of Users and Number of Objects, can be measured quantitatively. Obviously, the more users the more important, and similarly, the more objects in a project the more important it is. Hence, both quantitative criteria are supposed to be maximized. We can use Super Decisions or AHPHybrid package in R to solve this problem.

For the relative importance of one criterion to another and qualitative criteria among projects, we then construct AHP questionnaires given to a director who oversees the whole system. The result is presented in the next section. Generally speaking, using AHP, we can calculate $w_i\forall i = 1, ..., 5$ that satisfy $\sum_{i=1}^{5} w_i = 1$ where $w_i$ is the normalized weight for every project. Obviously, a very simple RAM allocation can then be made by multiplying $w_i$ with 12.4 GB.
Two-Stage Memory Allocation using AHP & Knapsack at PT Berca Hardayaperkasa
Khanis Satya, Indriati N. Bisono, Hanijanto Soewandi

Once we have allocated RAM into each project (the result of 1st stage problem), we can then formulate a Knapsack problem to decide on which 1-Cubes within a project to load as our 2nd stage problem. Mathematically, for every project, we can write the problem as:

**Second Stage Knapsack Problem**

Once we have allocated RAM into each project (the result of 1st stage problem), we can then formulate a Knapsack problem to decide on which 1-Cubes within a project to load as our 2nd stage problem. Mathematically, for every project, we can write the problem as:
Two-Stage Memory Allocation using AHP & Knapsack at PT Berca Hardayaperkasa
Khanis Satya, Indriati N. Bisono, Hanijanto Soewandi

\[ \text{max} \sum_{j=1}^{n} p_j x_j \]
\[ \text{s.t.} \sum_{j=1}^{n} c_j x_j \leq w_i M \]  

where: \( x_j \in \{0,1\} \), \( p_j \) is the (historical) hit count of I-Cube \( j \), \( c_j \) is the memory requirement of I-Cube \( j \), \( w_i \) is the normalized weight for every project as the result of AHP, and \( M = 12.4 \text{ GB} \). Again, Knapsack is a very well-known problem that had been studied extensively. Even though, it is still an NP-Complete problem, it actually belongs to the class of pseudo polynomial. Readers are referred to a classic and excellent book by Martello & Toth (1990) for detail. We simply use R packages: adagio for this purpose.

**FINDINGS AND DISCUSSION**

First-Stage AHP Result
The result of our questionnaire for the qualitative subjects can be summarized in Table 2 and Table 3. For the other 2 quantitative criteria, the result is given in Table 4. The quantitative criteria can be easily converted into normalized weight directly using the following formulation:

\[ w_i = \frac{x_i}{\sum_{j=1}^{n} x_j} \] for maximization \hfill (2a)

or

\[ w_i = \frac{(\sum_{j=1}^{n} x_j) - x_i}{\sum_{j=1}^{n} x_j} \] for minimization \hfill (2b)

where: \( x_i \) is the value of quantitative value.

| Criteria \( i \) | Criteria \( j \) |
|-----------------|-----------------|
| Due Date        | # of Users accessing the Project |
| Due Date        | # of Objects in the Project |
| Due Date        | Perceived Response Time of Dashboards |
| # of Users accessing the Project | # of Objects in the Project |
| # of Users accessing the Project | Perceived Response Time of Dashboards |
| # of Objects in the Project | Perceived Response Time of Browsing |
| Perceived Response Time of Dashboards | # of Objects in the Project |
| Perceived Response Time of Dashboards | Perceived Response Time of Browsing |

Table 2. Comparison Across Five Criteria
From the input, we can obtain the result as in Table 5 using AHPHybrid package. Without any surprise, the perceived performance of both the Dashboard (or Report/Hypercard) is the most important follows by the perceived browsing (overall system) performance, and then the number of users, and objects. Finally, the due date came at the very bottom of the list. It is also important to point out that all pair-wise comparison seems to meet consistency ratio.
Once the RAM for Intelligent Cube had been allocated for every project, we can easily proceed solving 5 Knapsack problems. At this point, we would like to draw readers’ attention that the weight for every project above can also be used to distribute RAM across five different projects for caching the Object, Element, & Report/Document (Result) – see Figure 3. Basically, any resource allocation that needs to be distributed across five different projects can be done using the above weights.

**Second-stage Knapsack Result**

The formulation of five knapsacks problem is relatively straightforward. We presented Table 6 for the problem and the shaded blue part as the solution to each independent Knapsack problem. Please note that this is still the same traditional 0-1 Knapsack problem, and NOT the 0-1 multiple knapsack problem. We just happened to assign the constraints per project using AHP. However, one can clearly see the advantage of this breakdown in term of computational complexity (in particular in conjunction with parallel computation). The traditional 0-1 Knapsack problem has the complexity O(nM) where n = 30 and M = 12698 (12.4 GB = 12698 MB) in our original example, after the assignment of memory (RAM) across 5 different projects, the problem will reduce to O(n4M4) where: n4 = 8 and M4 = 5499.

The solution to each Knapsack problem is marked in green in Table 6. We can immediately notice there is a different in term of decision to which I-Cubes to load, when (MicroStrategy) BI Server starts, compared to the original solution in table 3. This allocation of RAM makes sure that Project 4 and Project 3 which are two of the most important projects have all their I-Cubes loaded to memory (of course, at the expense at other I-Cubes).

Very careful readers will immediately notice that there are some left over RAM from Project 3 and Project 4 since all I-Cubes will only need 3193 + 4601 = 7794 MB, while we assign 3871 + 5499 = 9370 MB of RAM to Projects 3 and 4. Similarly, we have some unused memory from initial assignment in Projects 1, 2, and 5. Therefore, we can further optimize by redistributing the remaining RAM (131 + 87 + 678 + 898 + 41 = 1835 MB). At this point, we propose to solve another auxiliary Knapsack problem by combining the remaining RAM as well as considering unassigned I-Cubes’ hit-count and memory. Hence, we have the auxiliary 0-1 Knapsack problem. The problem formulation and solution (marked in yellow) are given Table 7.

| Table 6. Five independent 0-1 Knapsack problem that can be solved in parallel |
|---|
| Project 1 |
| Hit Count | 271 | 385 | 475 | 421 |
| Memory | 408 | 654 | 825 | 360 |
| Objective: to be maximized |
| <= 1118 MB |
| Project 2 |
| Hit Count | 23 | 273 | 30 | 393 |
| Memory | 181 | 255 |
| Objective: to be maximized |
| <= 1393 MB |
| Project 3 |
| Hit Count | 157 | 331 | 12 | 125 |
| Memory | 171 | 312 | 255 |
| Objective: to be maximized |
| <= 3871 MB |
| Project 4 |
| Hit Count | 66 | 224 | 325 | 259 |
| Memory | 49 | 252 | 461 | 180 |
| Objective: to be maximized |
| <= 5499 MB |
| Project 5 |
| Hit Count | 328 | 455 | 318 | 155 |
| Memory | 324 | 444 | 327 | 326 |
| Objective: to be maximized |
| <= 809 MB |

**Table 7. The auxiliary 0-1 Knapsack problem**
After the last aux Knapsack problem being solved, we have the following assignment of I-Cubes that will be loaded from each Project as in Table 8. The amount in the last column (in red) can be used to fill in the RAM governing in MicroStrategy BI Server in Figure 4. We will configure to load 25 I-Cubes into Intelligent Server memory, and keep the remaining 5 I-Cubes as Active, but not loaded into memory yet. We can contrast the final solution in Table 8 to the original single knapsack problem in Table 1 as in Table 9.

Table 8. The final RAM assignment for all 5 Projects

| I-Cube | Hit Count | Memory | Assigned RAM |
|--------|-----------|--------|--------------|
| Project 1 | 271 385 475 431 | 408 694 625 360 | 2087 MB |
| Project 2 | 23 273 30 253 | 412 951 639 667 | 1306 MB |
| Project 3 | 137 331 12 122 171 315 235 | 566 398 580 326 278 462 | 3193 MB |
| Project 4 | 65 224 325 269 49 252 457 180 | 708 707 500 714 628 393 370 381 | 4001 MB |
| Project 5 | 328 455 318 122 155 | 324 444 357 326 371 | 1496 MB |

Table 9. Some Statistics comparisons between single criterion vs. multi-criteria solutions

| Statistics                      | Single Criteria - Single Knapsack | Multi Criteria - Multi Knapsack |
|---------------------------------|-----------------------------------|---------------------------------|
| Hit Count Objective             | 6994                              | 8439                            |
| Memory Usage (MB)               | 12560                             | 12683                           |
| Project 1 Memory Setting        | 2087                              | 2087                            |
| Project 2 Memory Setting        | 3299                              | 1306                            |
| Project 3 Memory Setting        | 2087                              | 3193                            |
| Project 4 Memory Setting        | 3265                              | 4001                            |
| Project 5 Memory Setting        | 1322                              | 1496                            |
| Unused Memory (MB)              | 128                               | 5                               |
| Loaded I-Cubes                  | 24                                | 25                              |
| Unloaded (but Active I-Cubes)   | 6                                 | 5                               |
CONCLUSION AND FURTHER RESEARCH

We have demonstrated a two-stage approach to manage RAM allocation across several different projects in a (MicroStrategy) Business Intelligent Server that incorporates several criteria (both qualitative and quantitative). The approach is not limited to AHP, but it can also be extended to other methodology as long as it can provide a reasonable weight that can be used to allocate memory at the first stage. The result of the first stage multi criteria problem is also useful since it can be used to allocate RAM for Object, Element, and Result caches as well (not just limited to I-Cubes that are loaded when Intelligent Server starts).

A second stage approach using Knapsack becomes much simpler in term of computational complexity once the problem is broken down into multiple projects. We use the last auxiliary problem to squeeze the available RAM so that we can load as much I-Cubes as possible.

This simple multi-criteria optimization is able to satisfy more objectives with a bit extra memory usage, but it is able to load more I-Cubes into memory.

We would like to point out that the use of AHP (& its pairwise comparison) has many criticisms, in particular when it comes to criteria that is quantitative (see: Barzilai 1998, Saari & Sieberg 2004, Rezaie 2015, etc.). However, it also has many supports (see: Whitaker 2007). We do not intend to take side one way or the other. Our approach is generic enough and the AHP can be replaced by any other multi-criteria methodology if one likes to do so (e.g., McCaffrey 2009, etc.). Nonetheless, we choose AHP to demonstrate since it remains one of the most popular methods for multi-criteria problem to illustrate our approach to the problem that we face.

Furthermore, in this paper, we have not considered the stochastic nature of the demand. In reality, the setting needs to allow I-Cubes to grow up to certain percentage. So, the constraint parameter of the knapsack problem is actually a random variable. This may provide different perspective to the system and could be the subject for further research.

REFERENCES

Wang, J., Li, X., Chu, J. and Tsui, K.L., 2020. A two-stage approach for resource allocation and surgery scheduling with assistant surgeons. IEEE Access, 8, pp.49487-49496.

Chen, J. and Li, H., 2020. A Two-Phase Cloud Resource Provisioning Algorithm for Cost Optimization. Mathematical Problems in Engineering, 2020.

Lin, C.M. and Gen, M., 2008. Multi-criteria human resource allocation for solving multistage combinatorial optimization problems using multiobjective hybrid genetic algorithm. Expert Systems with Applications, 34(4), pp.2480-2490.

Singh, A. and Dutta, K., 2015. Apply AHP for resource allocation problem in cloud. Journal of Computer and Communications, 3(10), p.13.

Revathy, C. and Sekar, G., 2018. Analytic hierarchy process for resource allocation in cloud environment. Journal of Cyber Security and Mobility, pp.25-38.

Sharma, S. and Dubey, D., 2010. Multiple sourcing decisions using integrated AHP and knapsack model: a case on carton sourcing. The International Journal of Advanced Manufacturing Technology, 51(9), pp.1171-1178.
Mohammadi, S., Kheirkhah, A.S., & Behnamian, J. (2015), “Providing an Integrated Fuzzy AHP and Knapsack Method in Decision-Making and Resource Allocation to Suppliers,” Arth prabandh: A Journal of Economics and Management, 4, pp. 38 - 60.

Teknomo, K., 2006. Analytic hierarchy process (AHP) tutorial. Revoedu.com, 6(4), pp.1-20.

Martello, S. and Toth, P., 1990. Knapsack problems: algorithms and computer implementations. John Wiley & Sons, Inc.

Barzilai, J., 1998. On the decomposition of value functions. Operations Research Letters, 22(4-5), pp.159-170.

Saari, D.G. and Sieberg, K.K., 2004. Are partwise comparisons reliable?. Research in Engineering Design, 15(1), pp.62-71.

Rezaei, J., 2015. Best-worst multi-criteria decision-making method. Omega, 53, pp.49-57.

Whitaker, R., 2007. Criticisms of the Analytic Hierarchy Process: Why they often make no sense. Mathematical and Computer Modelling, 46(7-8), pp.948-961.

McCaffrey, J.D., 2009, April. Using the Multi-Attribute Global Inference of Quality (MAGIQ) technique for software testing. In 2009 Sixth International Conference on Information Technology: New Generations (pp. 738-742). IEEE.