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

by Khanis Satya

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

Khanis Satya[1], Indriati N. Bisono[2], Hanijanto Soewandi[3]

[1,2] International Business Engineering Program, Universitas Kristen Petra, Surabaya, Indonesia
[3] MicroStrategy, Tysons Corner, VA, USA
[1]khanissatya@gmail.com, [2]mlindri@petra.ac.id, [3]hsoewandi@microstrategy.com

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 $O(max\{n_j\}max\{M_j\})$ 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*

I. 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.

When many employees (business analysts) from a large organization (or company) start to adopt BI technology, the Information Technology (IT) department usually will set up a BI Server for economics of scale as well as to make sure that analysis is done on uniformly accepted data by the entire organization. Those employees who want to analyze similar subject are usually grouped together and given a set of Data Warehouse tables as the source of data. On the database side, this is often referred as Datamart. On the Business Intelligence side, in particular MicroStrategy BI enterprise software, usually these Datamart (or even Data Warehouse) tables are imported to form MicroStrategy Project.

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, in which Attributes and Metrics are placed together to be dragged & dropped to create Report/Dashboard. 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. A set of data is returned from the data warehouse and saved directly to the Intelligence Server memory. Multiple reports are built that gather data from the Intelligent Cube instead of querying the data warehouse.

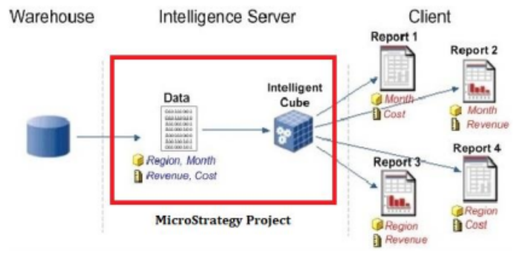
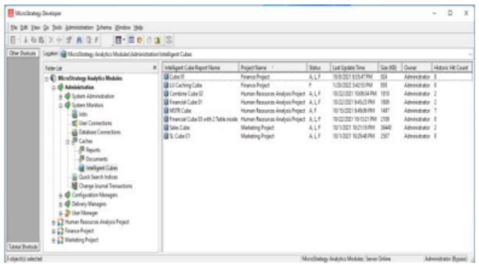


Figure 1. High-level MicroStrategy BI Server with respect to I-Cubes for a particular Project



Intelligent Cube Report Name	Project Name	Status	Last Update Time	Size (KB)	Owner	Historic Hit Count
Cube 01	Finance Project	A, L, F	10/8/2021 8:53:47 PM	504	Administrator	0
LU Caching Cube	Finance Project	F	1/20/2022 3:42:53 PM	858	Administrator	0
Combine Cube 02	Human Resources Analysis Project	A, L, F	10/22/2021 10:09:34 PM	1810	Administrator	2
Financial Cube 01	Human Resources Analysis Project	A, L, F	10/22/2021 9:45:23 PM	1808	Administrator	2
MSTR Cube	Human Resources Analysis Project	A, F	10/15/2021 9:49:08 PM	1487	Administrator	7
Financial Cube 01 with 2 Table inside	Human Resources Analysis Project	A, L, F	10/22/2021 10:13:21 PM	2108	Administrator	0
Sales Cube	Marketing Project	A, L, F	10/1/2021 10:21:19 PM	3840	Administrator	2
SL Cube 01	Marketing Project	A, L, F	10/1/2021 10:26:48 PM	2587	Administrator	0

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 (or a set of computers if clustering is configured) 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. In real life, the number of projects in a MicroStrategy I-Server typically less than 10. However, the number of Cubes could range from a handful numbers to several hundreds.

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.

II. 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. In the first stage, they proposed an integer programming formulation on how to allocate a (pair of) surgeon(s) to a particular operation, and in the second stage they propose another mathematical programming formulation for the start time of the surgery. Obviously, they consider all constraints related to the subject. Our problem is certainly different both in the context as well as the approach. In our problem, there are numerous criteria that needs to be considered.

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. While their proposed approach is elegant, we still cannot adapt to our problem since we have our Virtual Machine provision already, and we want to make sure that all projects have enough RAM allocated.

Lin & Gen (2008) considered 6 multi-criteria human reso6ce 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. We believe our problem is much simpler, and we do not want to rely on Genetic Algorithm that may take some time due to mutation of chromosome. Furthermore, some of our criteria is ordinal, *i.e.*, more difficult to quantify.

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.

In term of methodology, we found out that Olfati *et.al*. (2018) is using a combination of AHP and Linear Programming in two-stage problem. However, their approach to the problem is different from ours. They presented two-stage Linear Programming problem to obtain weight to the AHP formulation. Several other papers are also in this category: Balachandran & Golden (2005), Patel *et.al*. (2016) are some examples.

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.

III. PROBLEM FORMULATION AND RESEARCH METHOD

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	x_{ij}	I-Cube Name	Size (MB)	Hit Count	MicroStrategy Project	x_{ij}	I-Cube Name	Size (MB)	Hit Count
Project 1	x_{11}	Cube 11	408	271	Project 3	x_{36}	Cube 36	278	315
	x_{12}	Cube 12	694	385		x_{37}	Cube 37	462	255
	x_{13}	Cube 13	625	475	Project 4	x_{41}	Cube 41	708	66
	x_{14}	Cube 14	360	431		x_{42}	Cube 42	707	224
Project 2	x_{21}	Cube 21	412	23		x_{43}	Cube 43	500	325
	x_{22}	Cube 22	951	273		x_{44}	Cube 44	714	269
	x_{23}	Cube 23	639	30		x_{45}	Cube 45	628	49
	x_{24}	Cube 24	667	393		x_{46}	Cube 46	393	252
	x_{25}	Cube 25	811	181		x_{47}	Cube 47	370	467
	x_{26}	Cube 26	870	258		x_{48}	Cube 48	581	180
Project 3	x_{31}	Cube 31	566	157	Project 5	x_{51}	Cube 51	324	328
	x_{32}	Cube 32	398	331		x_{52}	Cube 52	444	455
	x_{33}	Cube 33	580	12		x_{53}	Cube 53	357	318
	x_{34}	Cube 34	526	125		x_{54}	Cube 54	326	125
	x_{35}	Cube 35	383	171		x_{55}	Cube 55	371	155

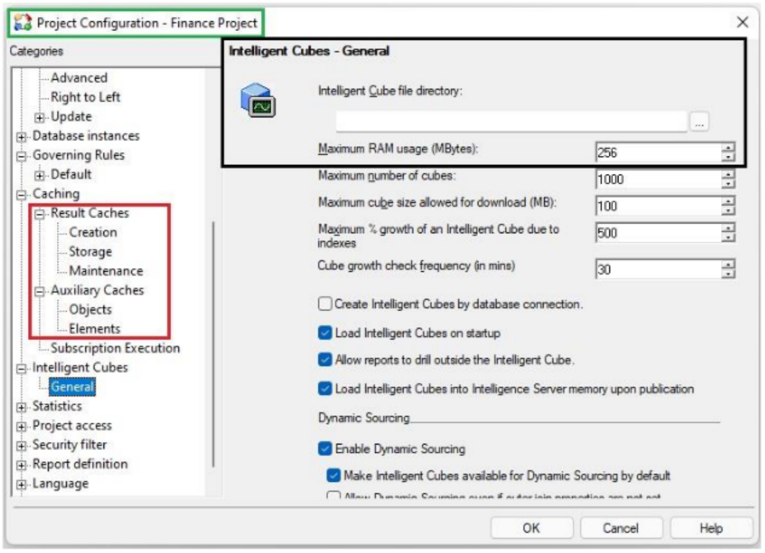
First-Stage Multi-Criteria Problem

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. Analytical Hierarchy Process (AHP) is a structured technique for organizing and analyzing complex decisions, based on mathematics and psychology. It was developed by Thomas Saaty in the 1970s (see Forman & Gass 2001 for an excellent review). It represents an approach to quantifying the weights of decision criteria. Individual experts' experiences are utilized to estimate the relative magnitudes of factors through pair-wise comparisons. Each of the respondents compares the relative importance each pair of items using a specially designed questionnaire.

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 $w_i \forall i = 1, \ldots, 5$ 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.



Intelligent Cubes - General

Intelligent Cube file directory:

Maximum RAM usage (MBytes): 256

Maximum number of cubes: 1000

Maximum cube size allowed for download (MB): 100

Maximum % growth of an Intelligent Cube due to indexes: 500

Cube growth check frequency (in mins): 30

☐ Create Intelligent Cubes by database connection.

☑ Load Intelligent Cubes on startup

☑ Allow reports to drill outside the Intelligent Cube.

☑ Load Intelligent Cubes into Intelligence Server memory upon publication

Dynamic Sourcing

☑ Enable Dynamic Sourcing

☑ Make Intelligent Cubes available for Dynamic Sourcing by default

Figure 3. MicroStrategy per Project Memory Allocation/Governing

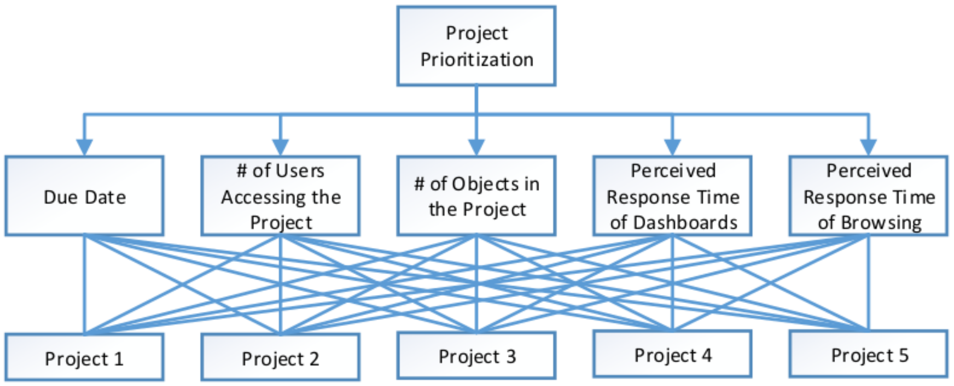

Project Prioritization
Due Date
# of Users Accessing the Project
# of Objects in the Project
Perceived Response Time of Dashboards
Perceived Response Time of Browsing
Project 1
Project 2
Project 3
Project 4
Project 5

Figure 4. Multi-criteria AHP Formulation for 1st Stage Problem

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 I-Cubes within a project to load as our 2nd stage problem. Mathematically, for every project, we can write the problem as:

$$\begin{aligned} &\max \textstyle\sum_{j=1}^{n_i} p_j x_j \\ &\text{s.t.} \textstyle\sum_{j=1}^{n_i} c_j x_j \le w_i M \end{aligned} \tag{1}$$

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 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.

IV. 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:

$$w_i = \frac{x_i}{\sum_{j=1}^{5} x_j} \quad \text{for maximization} \tag{2a}$$

or

$$w_i = \frac{\left(\sum_{j=1}^{5} x_j\right) - x_i}{\sum_{j=1}^{5} x_j} \quad \text{for minimization} \tag{2b}$$

where: x_i is the value of quantitative value.

Table 2. Comparison Across Five Criteria

Criteria *i*			**Criteria *j***
Due Date		**7**	# of Users accessing the Project
Due Date		2	# of Objects in the Project
Due Date		9	Perceived Response Time of Dashboards
Due Date		9	Perceived Response Time of Browsing
# of Users accessing the Project	5		# of Objects in the Project
# of Users accessing the Project		6	Perceived Response Time of Dashboards
# of Users accessing the Project		5	Perceived Response Time of Browsing
# of Objects in the Project		8	Perceived Response Time of Dashboards
# of Objects in the Project		**9**	Perceived Response Time of Browsing
Perceived Response Time of Dashboards	1		Perceived Response Time of Browsing

Table 3. Pairwise comparison across three qualitative criteria

Due Date Cirteria

Project *i*			**Project *j***
Project 1	4		Project 2
Project 1		3	Project 3
Project 1		4	Project 4
Project 1	6		Project 5
Project 2		7	Project 3
Project 2		8	Project 4
Project 2	3		Project 5
Project 3		2	Project 4
Project 3	5		Project 5
Project 4	6		Project 5

Perceived Dashboards Response

Project *i*			**Project *j***
Project 1		**3**	Project 2
Project 1		6	Project 3
Project 1		7	Project 4
Project 1		2	Project 5
Project 2		3	Project 3
Project 2		4	Project 4
Project 2	2		Project 5
Project 3		2	Project 4
Project 3	4		Project 5
Project 4	5		Project 5

Perceived Browsing Response

Project *i*			**Project *j***
Project 1	2		Project 2
Project 1		3	Project 3
Project 1		3	Project 4
Project 1	3		Project 5
Project 2		6	Project 3
Project 2		6	Project 4
Project 2	1		Project 5
Project 3	1		Project 4
Project 3	6		Project 5
Project 4	6		Project 5

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.

Table 4. Quantitative criteria for five projects (both are maximizing criteria)

Project	# of Users accessing the Project	# of Objects in the Project
Project 1	12	9
Project 2	40	21
Project 3	29	77
Project 4	105	122
Project 5	7	20

Table 5. AHP Result for Criteria and Overall Project Ranking

Criteria	Weight
Due Date	0.032
# of Users accessing the Project	0.139
# of Objects in the Project	0.046
Perceived Response Time of Dashboards	0.395
Perceived Response Time of Browsing	0.388

Project	Weight	RAM (GB)
Project 1	0.088	1.09
Project 2	0.110	1.36
Project 3	0.305	3.78
Project 4	0.433	5.37
Project 5	0.064	0.79

Nonetheless, the result in Table 5 provide a way to allocate available memory across 5 different projects as we have explained previously. The RAM allocation for every project is given in the last column of Table 5.
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 straight forward. 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(n_4M_4) where: $n_4 = 8$ and $M_4 = 5499$.

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	I-Cube	x_{11}	x_{12}	x_{13}	x_{14}	
	Hit Count	271	385	475	431	to be maximized
	Memory	408	694	625	360	<= 1116 MB

Project 2	I-Cube	x_{21}	x_{22}	x_{23}	x_{24}	x_{25}	x_{26}	
	Hit Count	23	273	30	393	181	258	to be maximized
	Memory	412	951	639	667	811	870	<= 1393 MB

Project 3	I-Cube	x_{31}	x_{32}	x_{33}	x_{34}	x_{35}	x_{36}	x_{37}	
	Hit Count	157	331	12	125	171	315	255	to be maximized
	Memory	566	398	580	526	383	278	462	<= 3871 MB

Project 4	I-Cube	x_{41}	x_{42}	x_{43}	x_{44}	x_{45}	x_{46}	x_{47}	x_{48}	
	Hit Count	66	224	325	269	49	252	467	180	to be maximized
	Memory	708	707	500	714	628	393	370	581	<= 5499 MB

Project 5	I-Cube	x_{51}	x_{52}	x_{53}	x_{54}	x_{55}	
	Hit Count	328	455	318	125	155	to be maximized
	Memory	324	444	357	326	371	<= 809 MB

Table 7. The auxiliary 0-1 Knapsack problem

BI Server	I-Cube	x_{11}	x_{12}	x_{21}	x_{22}	x_{25}	x_{26}	x_{53}	x_{54}	x_{55}	
	Hit Count	271	385	23	273	181	258	318	125	155	to be maximized
	Memory	408	694	412	951	811	870	357	326	371	<= 1835 MB

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

Project 1	I-Cube	x_{11}	x_{12}	x_{13}	x_{14}	**Assigned RAM**
	Hit Count	271	385	475	431	
	Memory	408	694	625	360	**2087 MB**

Project 2	I-Cube	x_{21}	x_{22}	x_{23}	x_{24}	x_{25}	x_{26}	
	Hit Count	23	273	30	393	181	258	
	Memory	412	951	639	667	811	870	**1306 MB**

Project 3	I-Cube	x_{31}	x_{32}	x_{33}	x_{34}	x_{35}	x_{36}	x_{37}
	Hit Count	157	331	12	125	171	315	255

	Memory	566	398	580	526	383	278	462		**3193 MB**
Project 4	**I-Cube**	x_{41}	x_{42}	x_{43}	x_{44}	x_{45}	x_{46}	x_{47}	x_{48}	
	Hit Count	66	224	325	269	49	252	467	180	
	Memory	708	707	500	714	628	393	370	581	**4601 MB**
Project 5	**I-Cube**	x_{51}	x_{52}	x_{53}	x_{54}	x_{55}				
	Hit Count	328	455	318	125	155				
	Memory	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	6439
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	4601
Project 5 Memory Setting	1822	1496
Unused Memory (MB)	128	5
Loaded I-Cubes	24	25
Unloaded (but Active I-Cubes)	6	5

V. CONCLUSION

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.

Limitation & Further Research

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, Rezaii 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.

Paydar, M.M. and Olfati, M., 2018. Designing and solving a reverse logistics network for polyethylene terephthalate bottles. *Journal of cleaner production*, *195*, pp.605-617.

Chandran, B., Golden, B. and Wasil, E., 2005. Linear programming models for estimating weights in the analytic hierarchy process. *Computers & Operations Research*, *32*(9), pp.2235-2254.

Patel, G., Mjema, G.D. and Godwin, K.M., 2016. Linear programming models for estimating weights in analytic hierarchy process and for optimization of human resource allocation. *International Journal of the Analytic Hierarchy Process*, *8*(2).

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.

Forman, E.H. and Gass, S.I., 2001. The analytic hierarchy process—an exposition. *Operations research*, *49*(4), pp.469-486.

Teknomo, K., 2006. Analytic hierarchy process (AHP) tutorial. *Revoledu. 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.

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

ORIGINALITY REPORT

6%	5%	5%	3%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	www.hindawi.com Internet Source	1%
2	www.tutorialspoint.com Internet Source	1%
3	glazerconstruction.com Internet Source	1%
4	en.wikipedia.org Internet Source	1%
5	www.ijsht-journals.org Internet Source	1%
6	Lin, C.M.. "Multi-criteria human resource allocation for solving multistage combinatorial optimization problems using multiobjective hybrid genetic algorithm", Expert Systems With Applications, 200805 Publication	1%
7	Submitted to University of Bedfordshire Student Paper	1%