Available online at: https://journals.researchsynergypress.com/index.php/lomr
Logistic and Operation Management Research (LOMR)
e-ISSN 2830-2680/p-ISSN 2830-3334
Volume 1 Number 2 (2022): 1-19

Implementation of AHP & Knapsack for Better Process in Making MicroStrategy Hypercard

Khanis S. Nugraha[1], Indriati N. Bisono[1], Hanijanto Soewandi[2]

[1]International Business Engineering Program, Universitas Kristen Petra, Surabaya, Indonesia

[2]MicroStrategy, Tysons Corner, VA, USA

Abstract

MicroStrategy Hypercard is a type of dashboard developed by MicroStrategy Business Intelligence that has successfully implemented zero-click analytics. However, the authors of this paper have found a weakness in the process of creating a MicroStrategy Hypercard. Specifically, the authors refer to the process of loading Intelligent Cubes. The weakness that the authors found in this process are that MicroStrategy Intelligent Cube loading capability is limited to RAM size on the user PC for data storage combined with the loading of schema objects and handle caches where the user must input all of these things manually. If the cubes' memory sizes exceed those of PC RAM, then the MicroStrategy System will unload the cube from the last most interacted with even though this cube needs to be made into Hypercard than other cubes. Therefore, we propose a two-stage resource allocation problem for handling RAM allocation in loading Intelligent Cubes for creating MicroStrategy Hypercard, with the first stage formulated as a multi-criteria problem that can be solved using Analytic Hierarchy Process (AHP) and the second stage being multiple (several) 0-1 classic Knapsack problems with constraints obtained from the first stage. When calculated in parallel, this method reduces computational complexity from O(nM) to O(1) (maxnjmaxMj). This method could make the process of creating a Hypercard more user-friendly. We explain our recommendation using a numerical example based on our experience.

Keywords: *Business Intelligence Server; Analytic Hierarchy Process; Knapsack problem, MicroStrategy Hypercard, MicroStrategy Intelligent Cube*

INTRODUCTION

Nowadays, many companies around the world use data to run a business. Regardless of whether the company is a small startup or a big enterprise, they use data to make important business decisions, solve company-related problems, increase business performance, and understand the market. Based on a survey made by Deloitte, 49% of respondents said analytics helps them make better decisions, 16% say that it better enables key strategic initiatives, and 10% say it helps them improve relationships with both customers and business partners (Panoho, 2019).

Based on the Deloitte survey, we can see that many companies need to use data to make business decisions and key strategic initiatives. However, not all stakeholders within a company or organization work in a data-related field (e.g., business analyst, data engineer, data scientist, data architect); thus, it is important to give data insights with zero click analysis. Davis (2019) defines zero click analysis as a type of data analytics where clean data with the work schema is coming to

Corresponding author
khanissatya@gmail.com

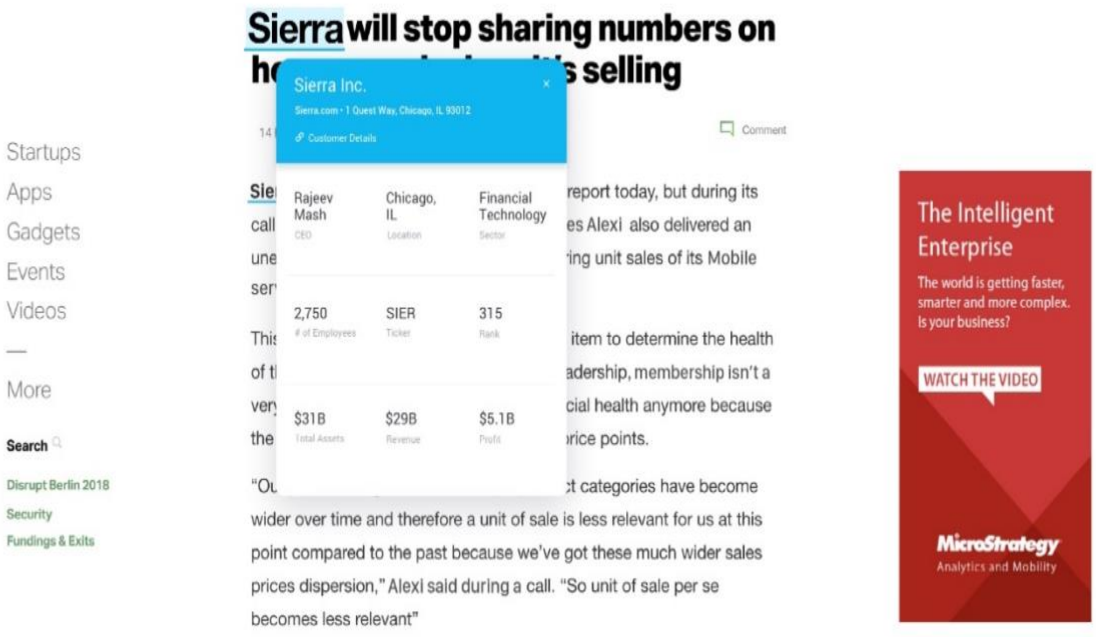
the user, and the user can interpret the data intuitively without the need for the user to be data savvy.

One of the business intelligence tools that have successfully implemented zero click analysis is MicroStrategy Business Intelligence with one of their solution called MicroStrategy Hypercard. MicroStrategy Hypercard is a type of dashboard that shows pre-assigned data where the dashboard would appear by moving the cursor of the user's PC into a pre-designated keyword in the user search engine. Based on what we read in the survey made by Deloitte, we can say that the use of MicroStrategy Hypercard is important for enterprises of today and enterprises of tomorrow.

The process of creating a MicroStrategy Hypercard begins with the process known as Extract, Transform, and Load (ETL). Extract refers to extracting raw data that the user wants to use into data management software (e.g., SQL). Transform refers to the process of transforming the raw data into the data that we want to show in the hypercard. Load refers to the process of loading the already transformed data into MicroStrategy Developer, which is a software that is used to create a set of schema objects known as Facts (measures of interest, e.g., dividend, net profit, etc.) and Attributes (grouping of data, e.g., product name, producer, dates, etc.) where these two schema objects make up the MicroStrategy Project. Facts are then combined with aggregation functions/other types of calculations, e.g., Sum, Avg, Min, Max, etc.) to create Metrics (e.g., Revenue, Profit, etc.).

Once this process is completed, then the next step is to place together Attributes and Metrics to create a report which will be converted to Intelligent Cube or Intelligent Cubes. The already made Intelligent Cube or Intelligent Cubes must then be loaded to MicroStrategy Enterprise, which is software for assigning keywords and creating Hypercard using the Intelligent Cube. It is to be noted that one Intelligent Cube can only be used to make one Hypercard (the user cannot mix two Intelligent Cubes into one Hypercard).

Finally, the user must connect the MicroStrategy Enterprise to the search Engine using a MicroStrategy search engine extension. This is done so that the Hypercard or Hypercards could appear when the user types the designated keyword and move the cursor of their pc to the designated keyword once the search result is shown in the search engine. Below is an example of a Hypercard with a "Sierra" as a designated keyword showing information such as CEO, Location, # of employees, etc.


Sierra will stop sharing numbers on h… …'s selling

Sierra Inc.
Sierra.com • 1 Quest Way, Chicago, IL 93012
Customer Details

Rajeev Mash	Chicago, IL	Financial Technology
CEO	Location	Sector
2,750	SIER	315
# of Employees	Ticker	Rank
$31B	$29B	$5.1B
Total Assets	Revenue	Profit

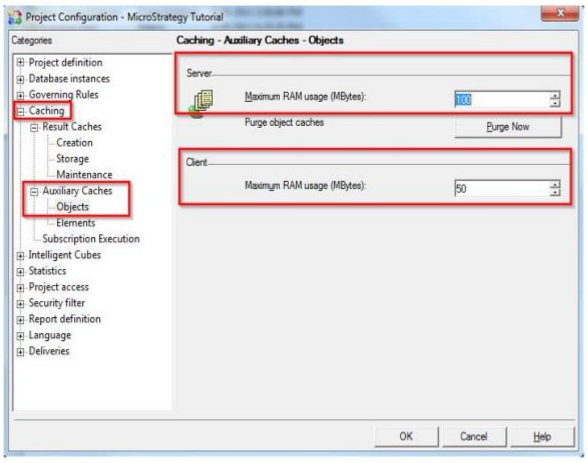
Sie… report today, but during its call… es Alexi also delivered an une… ing unit sales of its Mobile ser…

This… item to determine the health of t… adership, membership isn't a very… cial health anymore because the… rice points.

"Ou… t categories have become wider over time and therefore a unit of sale is less relevant for us at this point compared to the past because we've got these much wider sales prices dispersion," Alexi said during a call. "So unit of sale per se becomes less relevant"

Figure 1. Example of MicroStrategy Hypercard

The problem that we encounter (which is a topic that is discussed in this paper) is in the second process, particularly the process of RAM allocation of an intelligent cube. Within a real-life situation of using MicroStrategy, to create a project user must allocate portions of the RAM for caches within the Project (e.g., object cache, element cache, report & document caches). Furthermore, there could be multiple projects within an I-Server, and each project could have multiple numbers of cubes where each cube will have a different size of allocated RAM. RAM allocation must be done manually by the user, which could be difficult given the circumstances of each project, as explained before, especially for users with limited PC RAM size. The figures below give a clear picture of these situations:

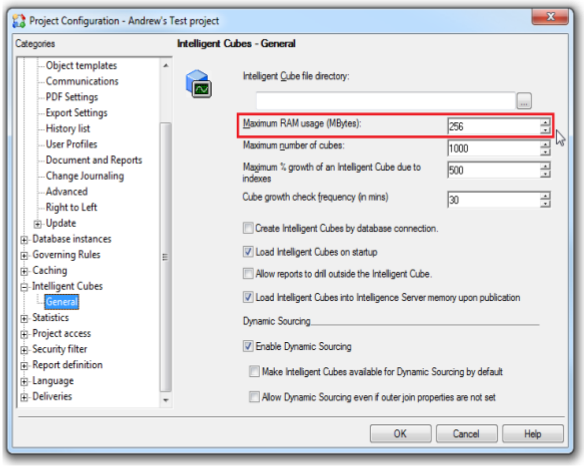


Caching - Auxiliary Caches - Objects

Server
Maximum RAM usage (MBytes): 100
Purge object caches — Purge Now

Client
Maximum RAM usage (MBytes): 50

Figure 2. Project Caches RAM Allocation



Intelligent Cubes - General

Intelligent Cube file directory:
Maximum RAM usage (MBytes): 256
Maximum number of cubes: 1000
Maximum % growth of an Intelligent Cube due to indexes: 500
Cube growth check frequency (in mins): 30

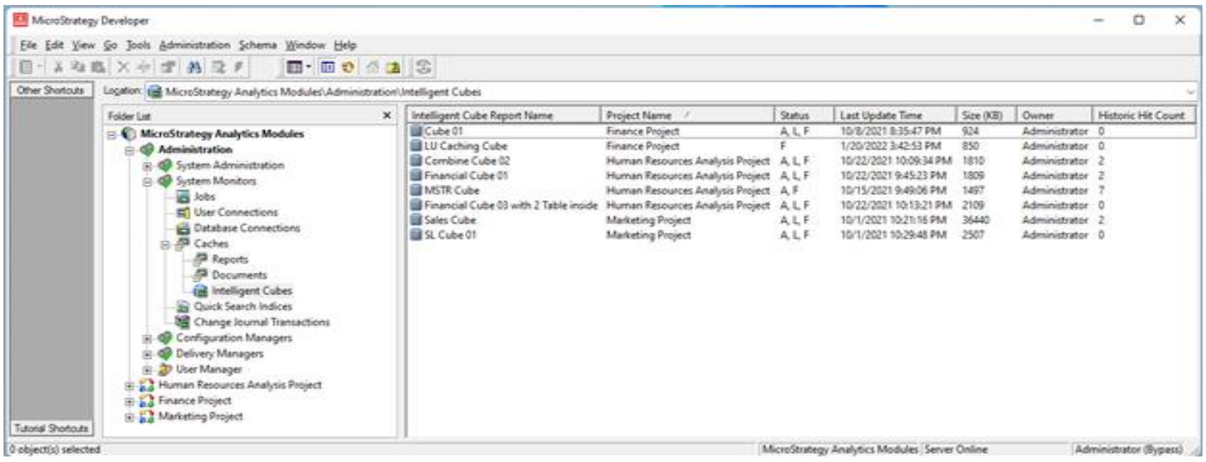
☐ Create Intelligent Cubes by database connection.
☑ Load Intelligent Cubes on startup
☐ Allow reports to drill outside the Intelligent Cube.
☑ Load Intelligent Cubes into Intelligence Server memory upon publication

Dynamic Sourcing
☑ Enable Dynamic Sourcing
☐ Make Intelligent Cubes available for Dynamic Sourcing by default
☐ Allow Dynamic Sourcing even if outer join properties are not set

Figure 3. Intelligent Cube RAM Allocation

Figure 2 illustrates the process of RAM allocation for the MicroStrategy Project that is usually done by a user in MicroStrategy. Note that the configuration process is per Project, which means that the User must configure the project one by one. The things that must be configured here are known as caches which consist of result caches and auxiliary caches. The result cache is a cache of an executed report or document that is stored on the Intelligence Server. The auxiliary cache is a cache that consists of results that are calculated and processed for the first time and the value of attributes in the lookup table.

Figure 3 illustrates the process of RAM allocation for MicroStrategy Intelligent Cube (I-Cube). The Maximum RAM usage here represents the total RAM allocated for Intelligent Cubes as a whole. The default number of Maximum RAM usage is set to 256 MB and may be changed manually.



Location: MicroStrategy Analytics Modules\Administration\Intelligent Cubes

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 8:35:47 PM	924	Administrator	0
LU Caching Cube	Finance Project	F	1/20/2022 3:42:53 PM	850	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	1809	Administrator	2
MSTR Cube	Human Resources Analysis Project	A, F	10/15/2021 9:49:06 PM	1497	Administrator	7
Financial Cube 03 with 2 Table inside	Human Resources Analysis Project	A, L, F	10/22/2021 10:13:21 PM	2109	Administrator	0
Sales Cube	Marketing Project	A, L, F	10/1/2021 10:21:16 PM	36440	Administrator	2
SL Cube 01	Marketing Project	A, L, F	10/1/2021 10:29:48 PM	2507	Administrator	0

0 object(s) selected | MicroStrategy Analytics Modules Server Online | Administrator (Bypass)

Figure 4. Example of I-Server Could Have Various Projects and Projects Could Have Many Cubes

Figure 4 depicts the fact that a single MicroStrategy BI Server may have several projects. Each of these contains numerous Intelligent Cubes (2 Intelligent Cubes belong to the "Finance" project, 4 I-Cube belongs to the "Human Resource Analysis" project, and 2 Intelligent Cubes belong to the "Marketing" project). Furthermore, it is worth pointing out that each I-Cube can have its own RAM usage size, and it can have different Statuses, namely: A = Active, F = File, and L = Loaded. Active here means that the cube is ready to be used, loaded means that the cube is being used, and file means that the cube is ins storage. There is also a "historic hit count" that represents how often a particular cube has been used in the past.

Consider a user who has to manage this (MicroStrategy) BI Server. The user is given a computer (or a group of computers if clustering is enabled) with a certain amount of memory (e.g., 18 GB, 64 GB, or several TBs in a real large-scale implementation) on which to load multiple I-Cubes that are grouped in multiple (MicroStrategy) Projects to serve many users (business analysts) so that they can create their Hypercard/Hypercards. This is the issue we will look at in this study. The number of projects in a MicroStrategy I-Server is often fewer than ten in real life. The number of Cubes, on the other hand, might range from a few dozen to several hundred.

To solve this problem, we have found a solution by applying AHP and Knapsack as two-stage resource allocation problems. Considering that the process of loading Intelligent Cubes may lead to a situation in which some particular projects do not have any I-Cube loaded into the memory, which indicates there is a limit to how many cubes we can load, we can use a 0-1 knapsack concept by applying weight and value to the cubes. Similarly, loading all Intelligent Cubes from a particularly important project may leave another project with very little (or even no) Intelligent Cubes being loaded, which indicates that there are some projects that are more important compared to another project. The reasoning of project importance is obtained from multiple criteria that need to be considered; we can use the AHP concept to solve this problem. We believe that by using our method, the process of creating a MicroStrategy Hypercard could be more user-friendly.

LITERATURE REVIEW

The Analytic Hierarchy Process (AHP) is a math and psychology-based approach for organizing and evaluating complicated choices using a 1 to 9 scale representing score of priorities. It was created in the 1970s by Thomas L. Saaty and has subsequently been improved (refer to Forman & Gass 2001 for an excellent review). It consists of three parts: the ultimate aim or problem to be solved, all viable solutions (referred to as alternatives), and the criteria that will be used to evaluate the alternatives. Because of AHP's ability to incorporate tangible as well as non-tangible factors, especially where the subjective judgments of different individuals constitute an important part of the decision making the users, the writers choose this method. Readers who are interested to learn more about AHP can visit AHP Tutorial at https://people.revoledu.com/kardi/tutorial/AHP/ (Teknomo, 2006) since we skipped explaining in detail regarding AHP because there are already numerous books and journal articles on this topic.
The Knapsack Problem is a famous Dynamic Programming Problem that falls in the combinatorial optimization category. Knapsack problem works this way: Determine the quantity of each item included in a collection given a set of objects, each with a weight and a value, so that the total weight

is less than or equal to a given limit and the total value is as large as possible (Pissinger, 2005). For this paper, we will use the 0-1 knapsack where in this concept, the items are either completely or no items are filled in a knapsack. This means that we cannot take part or fraction of items (for this particular case, the items are Intelligent Cubes).

Regarding this paper, there are two stages; where the first stage is how to distribute RAM at the Project level considering multiple factors using AHP, and then the second stage is how to distribute RAM to load a certain set of Intelligent Cubes, which is using Knapsack. Intelligent Cube is a collection of data that may be shared as a single in-memory copy among several reports made by different users. Users can return sets of data from the data warehouse and save them straight to Intelligence Server memory rather than returning data from the data warehouse for a single report. This type of problem is commonly known as a two-stage resource allocation problem which is a very well-known problem. However, despite the fact that several articles have been published on the two-stage resource allocation problem, none of them are applicable to our situation because the context of these papers is either different from ours or the method that they use is different (not AHP and Knapsack).

The closest papers in terms of application that we can find are Singh & Dutta (2015) and Revathy and Sekar (2018). In the first paper, they considered AHP to solve the 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.

Olfati et al. (2018) use a two-stage problem with the combination of AHP and Linear Programming. However, they take a different approach to the problem than we do. To acquire weight for the AHP formulation, they proposed a two-stage Linear Programming problem.

On industrial application, Sharma & Dubey (2010) a paper that combined AHP and Knapsack to solve industrial problems. Sharma & Dubey also considered a two-stage approach like ours. However, their application is on carton sourcing. 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 problems, while Sharma & Dubey only need to solve one.

The paper that is close to this paper in terms of context is a study done by Satya, K., Bisono, I. N., and Soewandi, H. (2022). They discuss the possibility of a two-stage memory allocation system within MicroStrategy using AHP and Knapsack. Basically, they propose a concept of a RAM allocation system for loading Intelligent Cubes within MicroStrategy so that the allocation of RAM is ideal for loaded Intelligent Cubes within MicroStrategy and overall computer performance. Their paper, however, does other possible uses of using their method (in this case, for creating MicroStrategy Hypercard efficiently and in a user-friendly way).

Based on the method comparison of the other two-stage problem solving between AHP combined with knapsack and other methods for this particular problem, we believe that our method is better. For example, if we use linear programming combined with the AHP method, linear programming has two weaknesses (Sherman,2020). First, given a specific objective and a set of constraints, it is possible that the constraints may not be directly expressible as linear inequalities. Second, linear programming is an advanced mathematical method to use that requires an understanding of advanced mathematics. If we compare our method to the combination of AHP and lexicographic goal programming, lexicographic goal programming has a weakness which is creating a decision in the most economically efficient manner but does not imply equality or fairness (Romero,1991) even though scholars still debate this weakness.

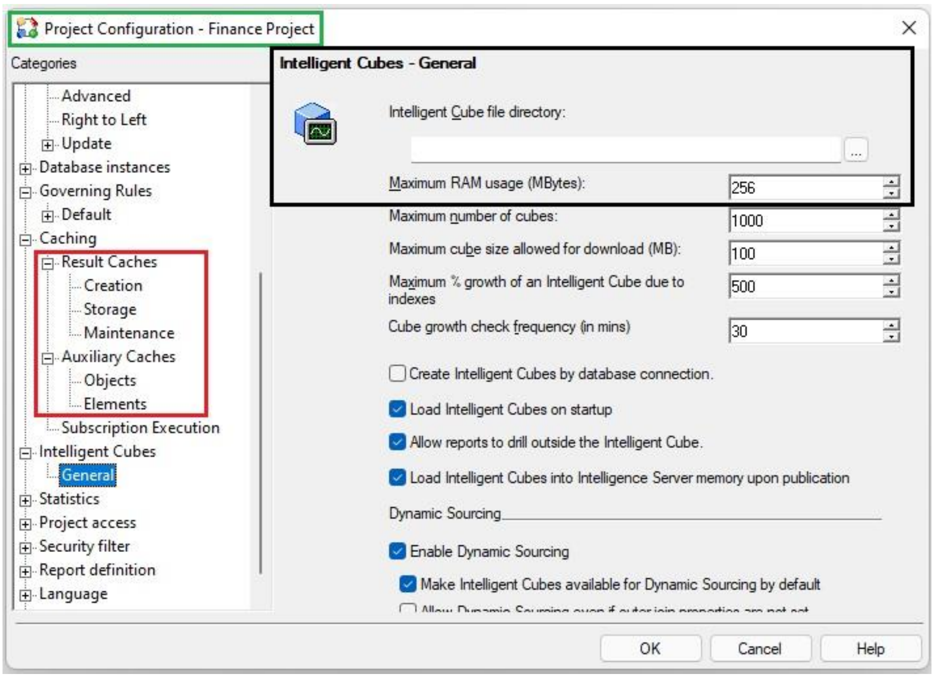
METHODOLOGY

Problem Formulation

To apply the solution that we proposed, the researchers will use data from the previous study about the creation of an Intelligent Cube (please refer to Satya, K., Bisono, I. N. and Soewandi, H. (2022)). Here, there are 30 I-Cubes organized into 5 MicroStrategy Projects, and call all these projects as Project A, Project B, Project C, Project D, and Project E. Each of these projects has four to eight cubes which we will call these cubes using codenames (e.g., A1, A2, A3). The Server system which manages MicroStrategy I-Server has 32 GB of RAM, where user also needs to allocate:

- 2 GB for Object cache (across 5 projects) – see Figure 2 (red box),
- 2 GB for Element cache (across 5 projects) – see Figure 2 (red box),
- 4 GB for Report & Document caches (across 5 projects) – see Figure 2 (red box), and
- 8 GB for processing/calculation.
- 3.6 GB of RAM to load Schema Objects

By deducting the RAM that is needed to manage the I-Server with the RAM needed to handle caches, processing, and loading then, he will need 12.4 GB to load some out of the 30 I-Cubes. Notice that the sum of RAM for all 30 I-Cubes = 16053 MB > 12.4 GB. Hence, there is a need for optimization.
The majority of those controlling rules (particular caches) are project-specific, as seen in Figure 4. (the green box indicates that it is per project). In Figure 4, the red box illustrates where the Object, Element, and Report/Document (Result) caches may be configured, and the black box indicates where the RAM allocation per project for I-Cubes can be configured. Also, the checkbox option "Load Intelligent Cubes on startup" is not an option we wish to use because there is insufficient RAM to load all Cubes, which will be solved using the two–stage allocation method.



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 4. MicroStrategy per Project Memory Allocation/Governing

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

MicroStartegy Project	Xij	Size	Hit Count	MicroStartegy Project	Xij	Size	Hit Count
Project A	XA1	408	271	Project C	XC6	278	315
	XA2	694	385		XC7	462	255
	XA3	625	475	Project D	XD1	708	66
	XA4	360	431		XD2	707	224
Project B	XB1	412	23		XD3	500	325
	XB2	951	273		XD4	714	269
	XB3	639	30		XD5	628	49
	XB4	667	393		XD6	393	252
	XB5	811	181		XD7	370	467
	XB6	870	258		XD8	581	180
Project C	XC1	566	157	Project E	XE1	324	328
	XC2	398	331		XE2	444	455
	XC3	580	12		XE3	357	318
	XC4	526	125		XE4	326	125
	XC5	383	171		XE5	371	155

First Stage Problem (AHP)

The first stage starts with the creation of the AHP model, particularly finding criteria for the AHP model. The object of focus here is the projects (in this case Project A to Project E) where the user must find criteria for what the project is to be worked on first (the work here is loading cubes).

There are many ways that users could do this depending on his/her situations, such as using personal judgment, asking his/her supervisor, and asking an expert. However, for simplification purpose, we will use the information provided by an expert from a data management company that has used MicroStrategy. Based on the expert opinion, there are five criterions a project important where these criteria are:

- Project deadline
- Number of users accessing the project
- Number of objects in a project
- Response time of report making
- Response time of overall computer performance

After finding the AHP criteria, users can now begin making the AHP model. To make this process easier, the researchers of this paper suggest using an AHP model processing app, SuperDecisions. The goal of the first stage problem is to find the local priority value, which is a score of priority in the form of decimal numbers and a comparison of each criterion to another in terms of importance. Three of these criteria, project deadline, the response time of report making, and the response time of overall computer performance, are qualitative in nature. The other two criteria, namely the number of users and the number of objects, are quantitative in nature. The AHP graph can be seen in Figure 5 below.

Note that both quantitative criteria are supposed to be maximized. To do this, we can use the AHPHybrid package in R. Using AHP, we can calculate $w_i \forall i = 1, \dots, 5$ that satisfy $\sum_{i=1}^{5} w_i = 1$ where wi is the normalized weight for every project. A very simple RAM allocation can then be made by multiplying w_i with 12.4 GB.

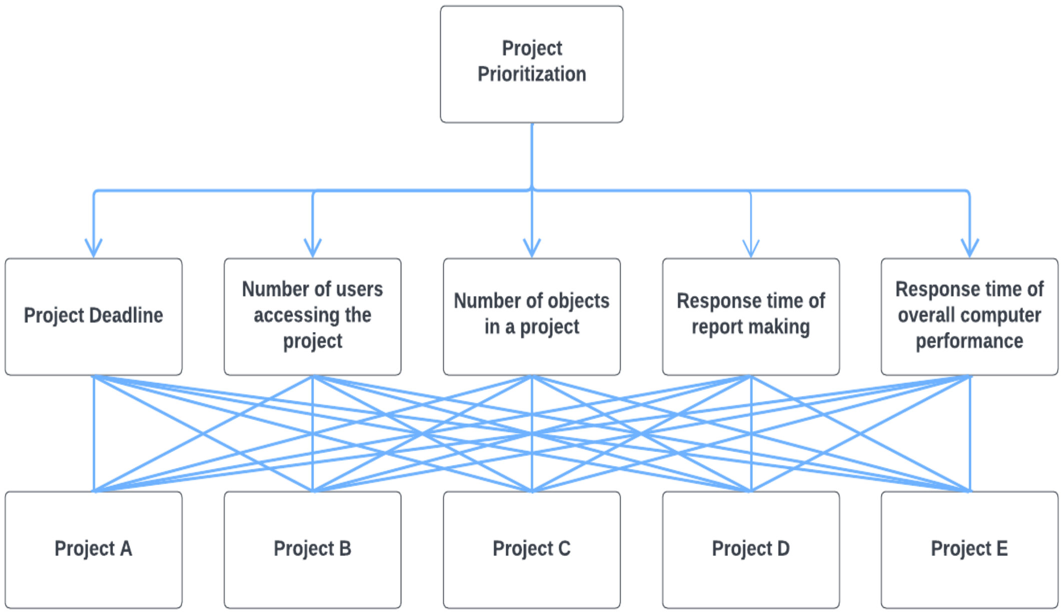

Project Prioritization
Project Deadline
Number of users accessing the project
Number of objects in a project
Response time of report making
Response time of overall computer performance
Project A
Project B
Project C
Project D
Project E

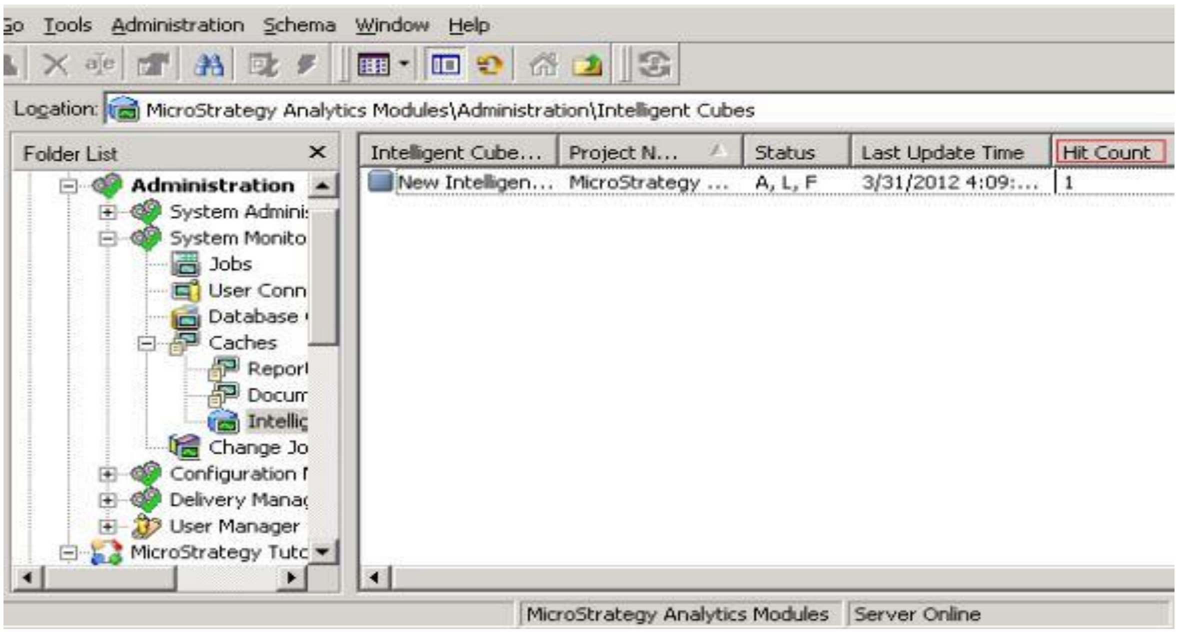
Figure 5. Multi-criteria AHP Diagram for 1st Stage Problem

Second Stage Problem (0-1 Knapsack)

Once RAM allocation of projects is complete (the result of 1st stage problem), we can move on to 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:

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

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. Using the R packages: adagio in R, we can solve this problem. Hit count is a Hit count' that refers to the number of times the Intelligent Cube has been used. The hit count would increase every time the report gets executed and hits the cache. Regarding the detailed usage of the adagio package in R, please refer to https://cran.r-project.org/web/packages/adagio/adagio.pdf.



Location: MicroStrategy Analytics Modules\Administration\Intelligent Cubes

Intelligent Cube...	Project N...	Status	Last Update Time	Hit Count
New Intelligen...	MicroStrategy ...	A, L, F	3/31/2012 4:09:...	1

Figure 6. Intelligent Cube Hit Count

FINDINGS AND DISCUSSION

First-Stage AHP Result

Tables 2 and 3 describe the results of our scoring for qualitative subjects. Table 4 shows the results for the other two quantitative criteria. Using the following formula, the quantitative criteria can be simply translated into normalized weight:

$$w_i = \frac{x_i}{\sum_{j=1}^{5} x_j} \qquad \text{for maximization} \qquad (2a)$$

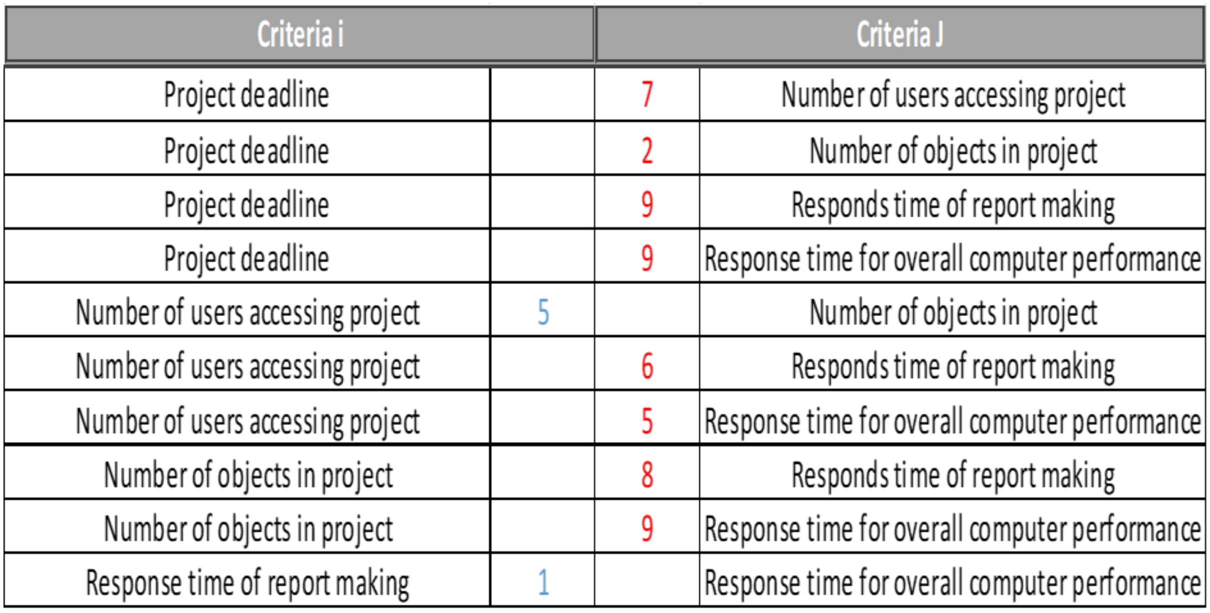
or

$$w_i = \frac{(\sum_{j=1}^{5} x_j) - x_i}{\sum_{j=1}^{5} x_j} \qquad \text{for minimization}$$

(2b)

where: x_i is the value of quantitative value.

Table 2. Comparison Across Five Criterions



Criteria i			Criteria J
Project deadline		7	Number of users accessing project
Project deadline		2	Number of objects in project
Project deadline		9	Responds time of report making
Project deadline		9	Response time for overall computer performance
Number of users accessing project	5		Number of objects in project
Number of users accessing project		6	Responds time of report making
Number of users accessing project		5	Response time for overall computer performance
Number of objects in project		8	Responds time of report making
Number of objects in project		9	Response time for overall computer performance
Response time of report making	1		Response time for overall computer performance

Table 3. Pairwise comparison across three qualitative criterions

Project Deadline Criteria

Criteria *i*			**Criteria *J***
Project A	4		Project B
Project A		3	Project C
Project A		4	Project D
Project A	6		Project E
Project B		7	Project C
Project B		8	Project D
Project B	3		Project E
Project C		2	Project D
Project C	5		Project E
Project D	6		Project E

Report Making

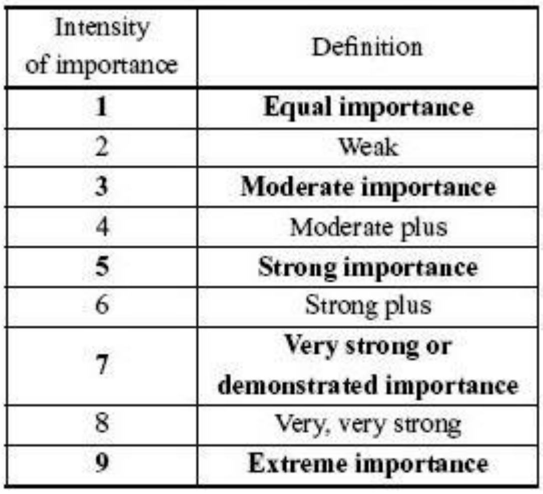
Criteria *i*			**Criteria *J***
Project A		3	Project B
Project A		6	Project C
Project A		7	Project D
Project A		2	Project E
Project B		3	Project C
Project B		4	Project D
Project B		2	Project E
Project C	2		Project D
Project C		4	Project E
Project D		5	Project E

Overall computer performance

Criteria *i*			**Criteria *J***
Project A	2		Project B
Project A		3	Project C
Project A		3	Project D
Project A	3		Project E
Project B		6	Project C
Project B		6	Project D
Project B	1		Project E
Project C	1		Project D
Project C	6		Project E
Project D	6		Project E

Table 2 and Table 3 represent the comparison across five criteria and the pairwise comparison across three qualitative criteria. The way to read the scores in both tables is by using the principle of AHP scoring proposed by Thomas L. Saaty. If the number of the score is on criteria *i*, then the score is in favor of criteria *i* (ex: Project A has demonstrated importance compared to project E) and vice versa. Saaty gives details regarding the numbers of the score as these:

Table 4. Scoring of AHP According to Thomas L. Saaty



Intensity of importance	Definition
1	**Equal importance**
2	Weak
3	**Moderate importance**
4	Moderate plus
5	**Strong importance**
6	Strong plus
7	**Very strong or demonstrated importance**
8	Very, very strong
9	**Extreme importance**

Table 5. Quantitative criteria for five projects (both are maximizing criteria)

Project	Number of users accessing project	Number of objects in project
Project A	12	9
Project B	40	21
Project C	29	77
Project D	105	122
Project E	7	20

Table 6. AHP Result for Criteria and Overall Project Ranking

Criterion	Weight		Project	Weight	RAM (GB)
Project deadline	0.032		Project A	0.032	1.09
Number of users accessing the project	0.139		Project B	0.139	1.36
Number of objects in a project	0.046		Project C	0.046	3.78
Response time of report making	0.395		Project D	0.395	5.37
Response time of overall computer performance	0.388		Project E	0.388	0.79

Using the results in Table 6, we must now allocate available RAM among five separate projects, as previously explained. Table 5's right side table shows the RAM distribution for each project. We can simply solve the 5 Knapsack issues after the RAM for Intelligent Cube has been assigned to each project. At this point, we'd like to point out to readers that the weight for each project listed above may also be utilized to allocate RAM among five separate projects for caching the Object, Element, and Report/Document (Result) - see Figure 2. Essentially, any resource allocation that has to be dispersed among five separate projects may be accomplished using the weights listed above.

Second Stage Knapsack Result

The formulation of the five knapsacks issue is quite simple. Table 6 was provided for the problem, and the shaded blue area was given as the answer to each individual Knapsack problem. Please keep in mind that this is the classic 0-1 Knapsack problem, not the 0-1 multiple knapsack problem. We merely happened to use AHP to apply the limitations per project. However, the advantage of this decomposition in terms of computing complexity is evident (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.

Very keen readers will see that there is some RAM left over from Projects 3 and 4 since all I-Cubes will only require 3193 + 4601 = 7794 MB, but we allot 3871 + 5499 = 9370 MB of RAM to Projects 3 and 4. Similarly, in Projects 1, 2, and 5, we have some unused RAM from the original assignment. As a result, we may optimize further by dispersing the remaining RAM (= 131 + 87 + 678 + 898 + 41 = 1835 MB). At this point, we suggest solving another auxiliary Knapsack problem by merging the remaining RAM and taking into account the hit count and memory of unassigned I-Cubes. As a result, the auxiliary 0-1 Knapsack issue arises. Table 7 shows the issue formulation and answer (in yellow).

Table 7. Five independent 0-1 Knapsack problems that can be solved in parallel

Project A	**I-Cube**	XA1	XA2	XA3	XA4	
	Hit Count	271	385	475	431	To be Maximized
	Memory	408	694	625	360	<=1116 MB

Project B	**I-Cube**	XB1	XB2	XB3	XB4	XB5	XB6	
	Hit Count	23	273	30	393	181	258	To be Maximized
	Memory	412	951	639	667	811	870	<=1393 MB

Project C	**I-Cube**	XC1	XC2	XC3	XC4	XC5	XC6	XC7	
	Hit Count	157	331	12	125	171	315	255	To be Maximized
	Memory	566	398	580	526	383	278	462	<=3871 MB

Project D	**I-Cube**	XD1	XD2	XD3	XD4	XD5	XD6	XD7	XD8	
	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 E	**I-Cube**	XE1	XE2	XE3	XE4	XE5	
	Hit Count	328	455	318	125	155	To be Maximized
	Memory	324	444	357	326	371	<=809 MB

Table 8. The auxiliary 0-1 Knapsack problem

BI Server	**I-Cube**	XA1	XA2	XB1	XB2	XB5	XB6	XE3	XE4	XE5	
	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

Following the resolution of the last auxiliary Knapsack problem, we have the following assignment of I-Cubes that will be loaded from each Project, as shown in Table 8. The amount in the last column (in red) may be used to fill the RAM governing in MicroStrategy BI Server, as shown in Figure 4.

We will set the Intelligent Server to load 25 I-Cubes into memory while leaving the remaining 5 I-Cubes active but not yet loaded into memory. As in Table 9, we can compare the final answer in Table 8 to the initial single knapsack issue in Table 1.

Table 9. RAM Assignments for All 5 Projects

Project A	**I-Cube**	XA1	XA2	XA3	XA4	
	Hit Count	271	385	475	431	Assigned RAM
	Memory	408	694	625	360	2087 MB

Project B	**I-Cube**	XB1	XB2	XB3	XB4	XB5	XB6	
	Hit Count	23	273	30	393	181	258	Assigned RAM
	Memory	412	951	639	667	811	870	1306 MB

Project C	**I-Cube**	XC1	XC2	XC3	XC4	XC5	XC6	XC7	
	Hit Count	157	331	12	125	171	315	255	Assigned RAM
	Memory	566	398	580	526	383	278	462	3193 MB

Project D	**I-Cube**	XD1	XD2	XD3	XD4	XD5	XD6	XD7	XD8	
	Hit Count	66	224	325	269	49	252	467	180	Assigned RAM
	Memory	708	707	500	714	628	393	370	581	4601 MB

Project E	**I-Cube**	XE1	XE2	XE3	XE4	XE5	
	Hit Count	328	455	318	125	155	Assigned RAM
	Memory	324	444	357	326	371	1496 MB

Creating MicroStrategy Hypercard from The Result of the Second Stage

The process of creating a MicroStrategy Hypercard based on the result of the second stage is quite simple. First, the user must choose to load the intelligent cubes that have been chosen according to the second stage knapsack result. Then, the user must connect their MicroStrategy Developer environment to Microstrategy Workstation by using the URL of MicroStrategy Library or by a MicroStrategy mstrc file. After the connection have been established, user can begin to make MicroStrategy Hypercard according to their needs. Below is the step-by-step process of making MicroStrategy Hypercard:

1. Open the Workstation window.
2. In the Navigation pane, click Create a New Card next to Cards (refer to the "+" icon next to cards in the MicroStrategy Workstation.
3. Select an in-memory cube.
4. Click OK. The Card Editor appears.
5. On the Template tab, click Change Template to use one of the five-card templates (see Figure 7). Regarding which type of hypercard the user should use, please look at Table 10

6. On the Format tab, click the Header drop-down to customize your card's header. Depending on the template you selected, other drop-down options are available to customize.
7. On the Widgets panel, drag and drop List, Matrix, Ring, or Text Box widgets on your card. To remove widgets, drag the widget off of the card. Please refer to Table 11 on widget usage.
8. Drag the attribute you want to serve as the keyword attribute from the left pane to the header of the card template. A star appears next to the chosen keyword attribute. The header displays sample data using one data point from your attribute.
9. Use the icons in the header to add a link or a contextual link, clear content, set keyword matching, or specify the display of attribute forms for titles and subtitles.
10. Add information to the card by dragging objects from the dataset to the card template. You can change the order of information on the card by dragging a specific widget or list item to a different location.
11. Click Save.

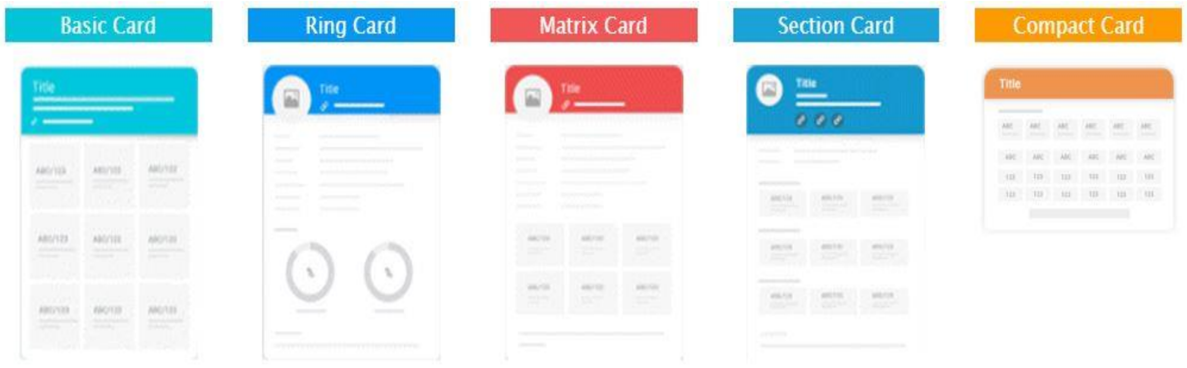
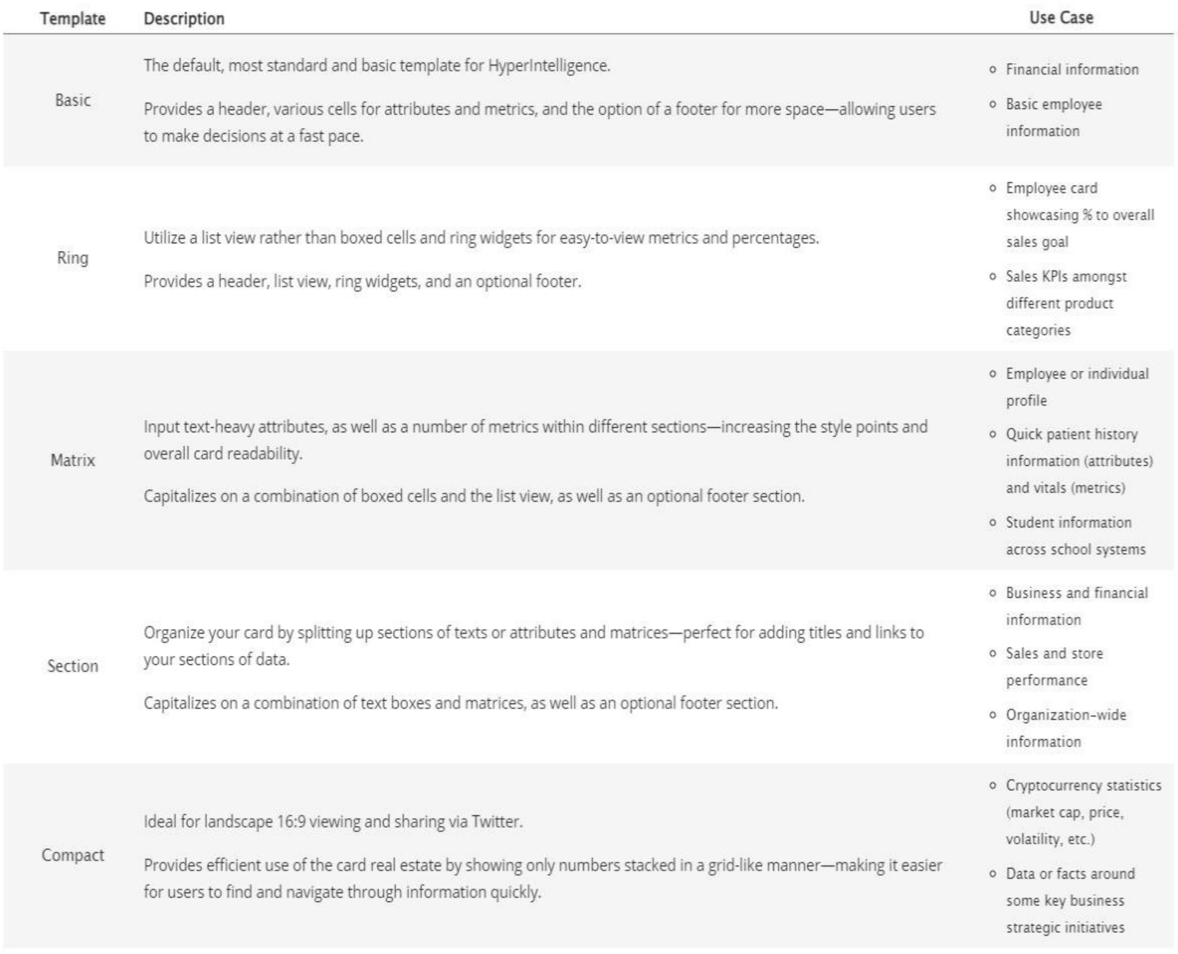


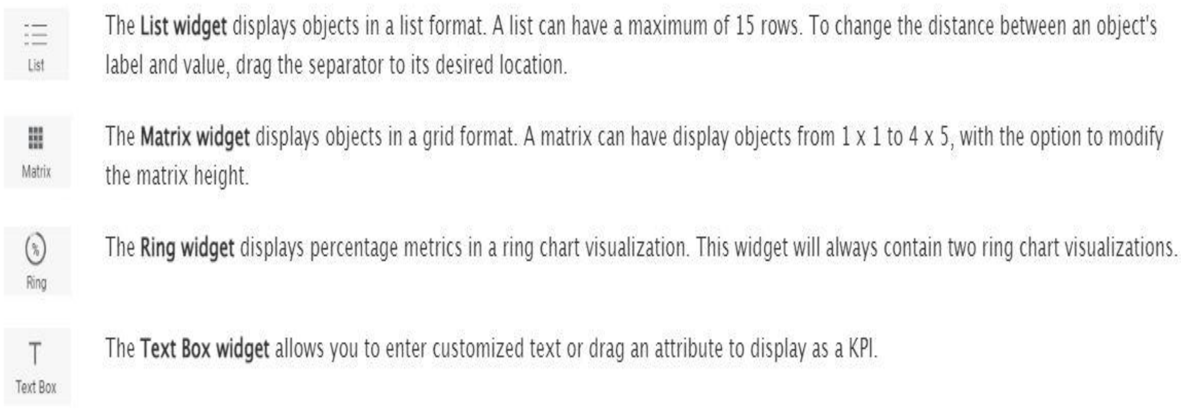
Figure 7. MicroStrategy Hypercard Variants

Table 10. Choosing MicroStrategy Hypercard by User Needs



Template	Description	Use Case
Basic	The default, most standard and basic template for HyperIntelligence. Provides a header, various cells for attributes and metrics, and the option of a footer for more space—allowing users to make decisions at a fast pace.	◦ Financial information ◦ Basic employee information
Ring	Utilize a list view rather than boxed cells and ring widgets for easy-to-view metrics and percentages. Provides a header, list view, ring widgets, and an optional footer.	◦ Employee card showcasing % to overall sales goal ◦ Sales KPIs amongst different product categories
Matrix	Input text-heavy attributes, as well as a number of metrics within different sections—increasing the style points and overall card readability. Capitalizes on a combination of boxed cells and the list view, as well as an optional footer section.	◦ Employee or individual profile ◦ Quick patient history information (attributes) and vitals (metrics) ◦ Student information across school systems
Section	Organize your card by splitting up sections of texts or attributes and matrices—perfect for adding titles and links to your sections of data. Capitalizes on a combination of text boxes and matrices, as well as an optional footer section.	◦ Business and financial information ◦ Sales and store performance ◦ Organization-wide information
Compact	Ideal for landscape 16:9 viewing and sharing via Twitter. Provides efficient use of the card real estate by showing only numbers stacked in a grid-like manner—making it easier for users to find and navigate through information quickly.	◦ Cryptocurrency statistics (market cap, price, volatility, etc.) ◦ Data or facts around some key business strategic initiatives

Table 11. Widget Usage in Creating MicroStrategy Hypercard



List	The **List widget** displays objects in a list format. A list can have a maximum of 15 rows. To change the distance between an object's label and value, drag the separator to its desired location.
Matrix	The **Matrix widget** displays objects in a grid format. A matrix can have display objects from 1 x 1 to 4 x 5, with the option to modify the matrix height.
Ring	The **Ring widget** displays percentage metrics in a ring chart visualization. This widget will always contain two ring chart visualizations.
Text Box	The **Text Box widget** allows you to enter customized text or drag an attribute to display as a KPI.

Connecting MicroStrategy Hypercard to Search Engine

The process of connecting Hypercard to a user search engine begins with downloading a search engine extension called MicroStrategy Hyperintelligence. After that user must connect their extension to the same environment as the MicroStrategy Developer. After this, there should be an option to turn on the reaction of Hypercard in the extension based on the card title so that when the user types the keyword of the hypercard/hypercards in their search engine, the dashboard will appear.

CONCLUSION

Concluding Remarks

The purpose of this paper is to propose a user-friendly method for users, particularly business analysts, data scientists, and data architects, to create MicroStrategy Hypercard by applying a two-stage problem-solving solution using AHP and knapsack. Based on the result in chapter four, the researchers have successfully demonstrated a possibility of 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 first stage of the RAM allocation is to create the AHP model of each project and create a prioritization criterion. The solution to the first stage multi-criterion problem is also important since it may be used to allocate RAM for Object, Element, and Result caches (not just limited to I-Cubes that are loaded when Intelligent Server starts).

The second stage is to create a knapsack model for prioritizing the cube/cubes that need to be loaded. A second-stage solution employing Knapsack becomes significantly simpler in terms of computing complexity once the issue is divided into several phases, the first of which is cube allocation based on AHP results and the second of which is loading the remaining cubes depending on available RAM space.

After the two-stage process is done, the complete user could continue the process of loading the intelligent cubes and connect it to the MicroStrategy Enterprise so that users can continue the process of making a hypercard and utilize it in their search engine.

Limitation & Further Research

Some researchers are against the use of AHP (and its pairwise comparison), especially when it comes to quantitative criteria (see: Barzilai 1998, Saari & Sieberg 2004, Rezaii 2015, etc. It does, however, have a lot of supporters (see: Whitaker 2007). We are not going to support one side over the other. Our method is sufficiently broad, and if required, the AHP might be replaced by any other multi-criteria methodology (e.g., McCaffrey 2009, etc.). Nonetheless, we picked AHP to present since it is still one of the most frequent methodologies for multi-criteria issue resolution, and we wanted to emphasize our perspective on the scenario at hand.

In this work, we also neglected to account for the stochastic nature of demand. In reality, the settings should allow I-Cubes to extend within a specific percentage. As a result, the constraint parameter of the knapsack problem is a random variable. This might provide a new perspective on the system and serve as the subject of future study.

REFERENCES

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

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.

Davis, R., 2019. How 'zero-click analytics' will empower enterprise-wide data adoption. TechNative.io

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

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.

Panoho, K., 2019. Council post: The age of analytics and the importance of data quality. Forbes.com

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).

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.

Pisinger, D., 2005. Where are the hard knapsack problems? Computers &amp; Operations Research, 32(9), pp.2271–2284.

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

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

Romero, C., 1991. Handbook of Critical Issues in goal programming, Pergamon Press.

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

Satya, K., Bisono, I. N., & Soewandi, H. (2022). Two-Stage Memory Allocation using AHP &amp; Knapsack at PT Berca Hardayaperkasa. RSF Conference Series: Business, Management and Social Sciences, 2(1), 231–241. https://doi.org/10.31098/bmss.v2i1.539

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.

Sherman, F., 2020. The disadvantages of Linear Programming. Sciencing. Available at: https://sciencing.com/info-12195571-disadvantages-linear-programming.html [Accessed July 4, 2022].

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

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

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.