International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 2 Issue 2, February- 2013

Effective Distributed Dynamic Load Balancing For The Clouds

Y. Ranjith Kumar[1], M. Madhu Priya[2] , K. Shahu Chatrapati[3]
[1] Department of CSE, DMSSVH College of Engineering, Machilipatnam, INDIA
[2] Department of CSE, DMSSVH College of Engineering, Machilipatnam, INDIA
[3]Department of CSE, JNTUH.C.E., Hyderabad, INDIA

Abstract
”Cloud computing” is a term, which involves virtualization, distributed computing, networking, software and web services. A cloud consists of several elements such as clients, datacenter and distributed servers. It includes fault tolerance, high availability, scalability, flexibility, reduced overhead for users, reduced cost of ownership, on demand services etc. Central to these issues lies the establishment of an effective load balancing algorithm. The load can be CPU load, memory capacity, delay or network load. Load balancing is the process of distributing the load among various nodes of a distributed system to improve both resource utilization and job response time while also avoiding a situation where some of the nodes are heavily loaded while other nodes are idle or doing very little work. Load balancing ensures that all the processor in the system or every node in the network does approximately the equal amount of work at any instant of time. This technique can be sender initiated, receiver initiated, symmetric (combination of sender initiated, receiver initiated types) static, dynamic centralized or distributed type. Various studies show that up to 80% of the workstations are idle depending on time of day [12], therefore these are advantageous to use. The idle time and computing power of processors can be used to make the processing cost-effective.
Our objective is to explain the concept of load balancing, types of load balancing algorithms, general idea about dynamic load balancing algorithms and the different policies that can be used in it and gives an overall description of various distributed load balancing algorithms that can be used in case of clouds.

1. Introduction
In Cloud computing, services can be used from diverse and widespread resources, rather than remote servers or local machines. There is no standard definition of Cloud computing. Generally it consists of a bunch of distributed servers known as masters, providing demanded services and resources to different clients known as clients in a network with scalability and reliability of datacenter. The distributed computers provide on-demand services. Services may be of software resources (e.g. Software as a Service, SaaS) or physical resources (e.g. Platform as a Service, PaaS) or hardware/ infrastructure (e.g. Hardware as a Service, HaaS or Infrastructure as a Service, IaaS). Amazon EC2 (Amazon Elastic Compute Cloud) is an example of cloud computing services.[2]

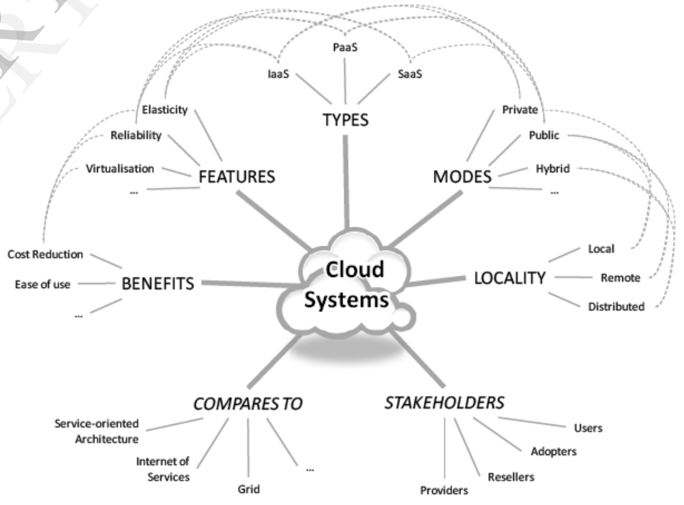

Figure 1.Cloud Systems

1.1 Cloud Components
A Cloud system consists of 3 major components such as clients, datacenter, and distributed servers. Each element has a definite purpose and plays a specific role.

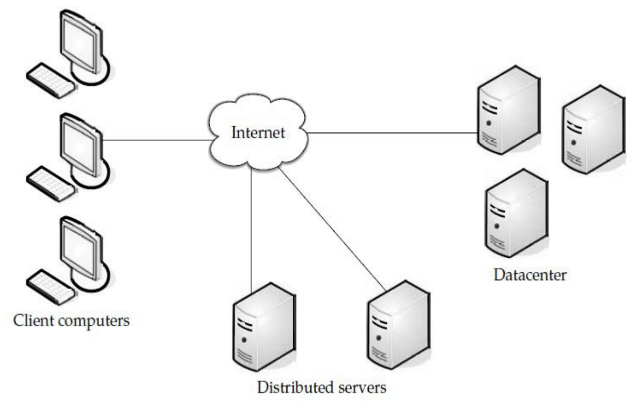

Figure 2.Cloud Components

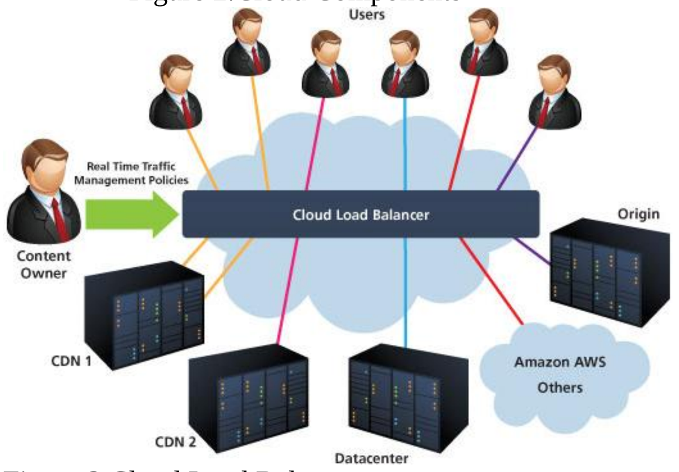

Figure 3.Cloud Load Balancer

2. Load Balancing

It is a process of reassigning the total load to the individual nodes of the collective system to make resource utilization effective and to improve the response time of the job, simultaneously removing a condition in which some of the nodes are over loaded while some others are under loaded.

A load balancing algorithm which is dynamic in nature does not consider the previous state or behavior of the system, that is, it depends on the present behavior of the system. The important things to consider while developing such algorithm are: estimation of load, comparison of load, stability of different system, performance of system, interaction between the nodes, nature of work to be transferred, selecting of nodes and many other ones [13] . This load considered can be in terms of CPU load, amount of memory used, delay or Network load. When a given workload is applied on any cluster's node, this given load can be efficiently executed if the available resources are efficiently used. So that, there must be a mechanism for choosing the nodes that have these resources. Scheduling is a component or a mechanism, which is responsible for the selection of a cluster node, to which a particular process will be placed. This mechanism will investigate the load balancing state [9,10]. Hence, scheduling needs algorithms to solve such problems. In real world, load balancing affected by 3 factors mainly [11]:

• The environment in which one wishes to balance the load.
• The nature of the load itself.
• The load balancing tools available.

2.1 Goals of Load balancing

As given in [4], the goals of load balancing are :

- To improve the performance substantially
- To have a backup plan in case the system fails even partially
- To maintain the system stability
- To accommodate future modification in the system

2.2 Types of Load balancing algorithms

Depending on who initiated the process, load balancing algorithms can be of three categories as given in [4]:

Sender Initiated: If the load balancing algorithm is initialized by the sender.
Receiver Initiated: If the load balancing algorithm is initiated by the receiver.
Symmetric: It is the combination of both sender initiated and receiver initiated.

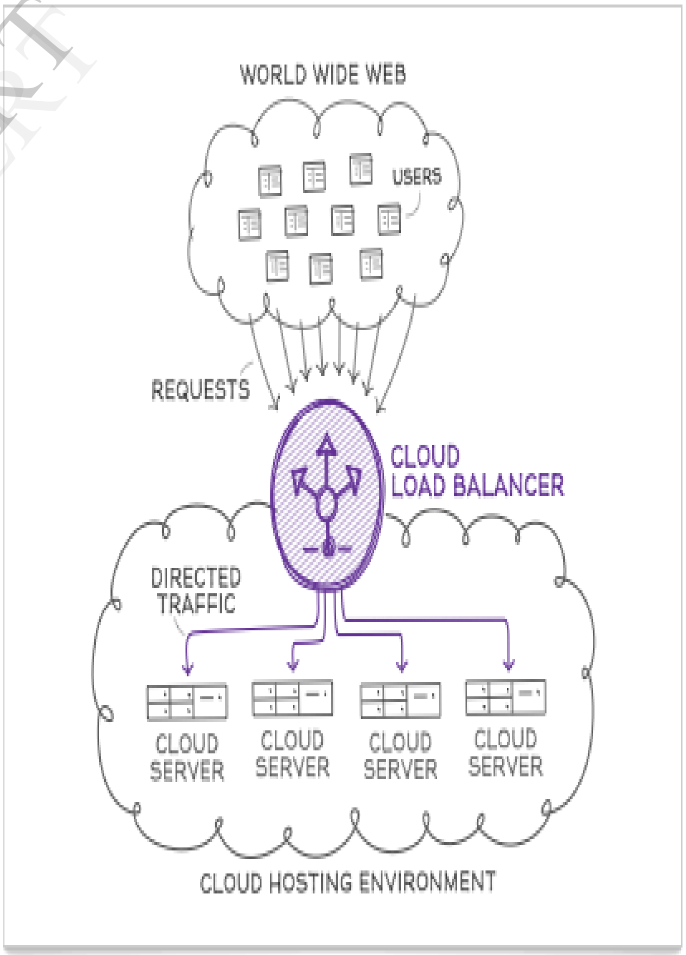

Figure 4.Cloud Hosting Environment and Load Balancer

Depending on the current state of the system, load balancing algorithms can be divided into 2 categories as given in [4]:

Static: It doesn’t depend on the current state of the system. Prior knowledge of the system is needed
Dynamic: Decisions on load balancing are based on current state of the system. No prior knowledge is needed. So it is better than static approach. Here we will discuss on various dynamic load balancing algorithms for the clouds of different sizes.

3. Dynamic Load balancing algorithm

In a distributed system, dynamic load balancing can be done in two different ways: distributed and non-distributed.

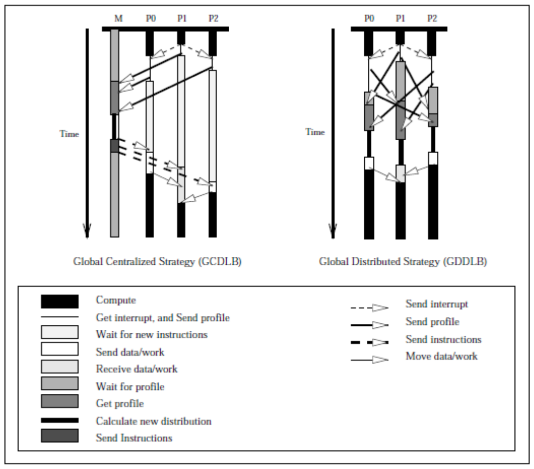

Figure 5.Centralized Vs Distributed Strategies

In the distributed one, the dynamic load balancing algorithm is executed by all nodes present in the system and the task of load balancing is shared among them. The interaction among nodes to achieve load balancing can take two forms: cooperative and non-cooperative [4].
In the first one, the nodes work side-by-side to achieve a common objective, for example, to improve the overall response time, etc.
In the second form, each node works independently toward a goal local to it, for example, to improve the response time of a local task.
Dynamic load balancing algorithms of distributed nature, usually generate more messages than the non-distributed ones because, each of the nodes in the system needs to interact with every other node.
A benefit, of this is that even if one or more nodes in the system fail, it will not cause the total load balancing process to halt; it instead would affect the system performance to some extent.
Distributed dynamic load balancing can introduce immense stress on a system in which each node needs to interchange status information with every other node in the system. It is more advantageous when most of the nodes act individually with very few interactions with others.
In non-distributed type, either one node or a group of nodes do the task of load balancing.
Non-distributed dynamic load balancing algorithms can take two forms: centralized and semi-distributed.
In the first form, the load balancing algorithm is executed only by a single node in the whole system: the central node. This node is solely responsible for load balancing of the whole system. The other nodes interact only with the central node.

In semi-distributed form, nodes of the system are partitioned into clusters, where the load balancing in each cluster is of centralized form. A central node is elected in each cluster by appropriate election technique which takes care of load balancing within that cluster. Hence, the load balancing of the whole system is done via the central nodes of each cluster [4].

Centralized dynamic load balancing takes fewer messages to reach a decision, as the number of overall interactions in the system decreases drastically as compared to the semi distributed case. However, centralized algorithms can cause a bottleneck in the system at the central node and also the load balancing process is rendered useless once the central node crashes. Therefore, this algorithm is most suited for networks with small size.

4. Policies or Strategies in Dynamic load balancing

There are 4 policies [4]:
Transfer Policy: The part of the dynamic load balancing algorithm which selects a job for transferring from a local node to a remote node is referred to as Transfer policy or Transfer strategy.
Selection Policy: It specifies the processors involved in the load exchange (processor matching)
Location Policy: The part of the load balancing algorithm which selects a destination node for a transferred task is referred to as location policy or Location strategy.
Information Policy: The part of the dynamic load balancing algorithm responsible for collecting information about the nodes in the system is referred to as Information
policy or Information strategy.

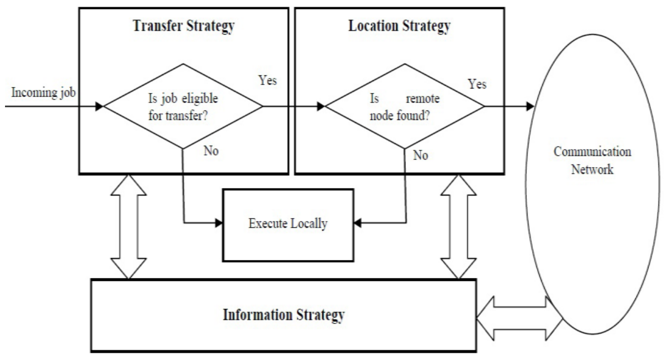

Figure 6: Interaction among components of a dynamic load balancing algorithm (adopted from [4])

5. Metrics for Load Balancing in Clouds

The existing load balancing techniques in clouds, consider various parameters like performance, response time, scalability, throughput, resource utilization, fault tolerance, migration time and associated overhead. But, for an energy-efficient load balancing, metrics like energy consumption and carbon emission should also be considered.

***Overhead Associated* -** determines the amount of overhead involved while implementing a load-balancing algorithm. It is composed of overhead due to movement of tasks, inter-processor and inter-process communication. This should be minimized so that a load balancing technique can work efficiently.
***Throughput* -** is used to calculate the no. of tasks whose execution has been completed. It should be high to improve the performance of the system.
Performance – is used to check the efficiency of the system. It has to be improved at a reasonable cost e.g. reduce response time while keeping acceptable delays.
Resource Utilization - is used to check the utilization of resources. It should be optimized for an efficient load balancing.
***Scalability* -** is the ability of an algorithm to perform load balancing for a system with any finite number of nodes. This metric should be improved.
***Response Time* -** is the amount of time taken to respond by a particular load balancing algorithm in a distributed system. This parameter should be minimized.
***Fault Tolerance* -** is the ability of an algorithm to perform uniform load balancing in spite of arbitrary node or link failure. The load balancing should be a good fault-tolerant technique.
Migration time - is the time to migrate the jobs or resources from one node to other. It should be minimized in order to enhance the performance of the system.
Carbon Emission (CE) - calculates the carbon emission of all the resources in the system. As energy consumption and carbon emission go hand in hand, the more the energy consumed, higher is the carbon footprint. So, for an energy-efficient load balancing solution, it should be reduced

5.1 Types of Load Balancing

Load Balancing can be achieved through two different methodologies, they are
1) Software load balancing
2) Hardware load balancing

Software Load Balancing:

The software load balancing fits in the application layer of the stack. In case of software load balancing software program works as an arbiter to decide on distributing the job to particular engine for computation. Ex: In case of a web application a simple proxy like HA Proxy can be setup in the web server such a way to route few requests to application server-x and other requests to application server-y such a way load balancing could be achieved on the software level.

Hardware Load Balancing:

The hardware load balancing is the method of using multiple computation nodes and delegating work to multiple units in order to increase efficiency which in turn improves the performance of the application. A load balancer software works as a facade and redirects the incoming request to the appropriate machine for effectual manipulation. Currently in this cloud age, cost of hardware is less than the need for the performance. So the hardware load balancing would be highly recommended for applications demanding eminent scalability.

6. Distributed Load Balancing for the Clouds

In complex and large systems, there is a tremendous need for load balancing. For simplifying load balancing globally (e.g. in a cloud), one thing which can be done is, employing techniques would act at the components of the clouds in such a way that the load of the whole cloud is balanced. For this purpose, we are discussing three types of solutions which can be applied to a distributed system [7]: honeybee foraging algorithm, a biased random sampling on a random walk procedure and Active Clustering.

6.1 Honeybee Foraging Algorithm

This algorithm is derived from the behavior of honey bees for finding and reaping food. There is a class of bees called the forager bees which forage for food sources, upon finding one, they come back to the

beehive to advertise this using a dance called waggle dance. The display of this dance, gives the idea of the quality or quantity of food and also its distance from the beehive. Scout bees then follow the foragers to the location of food and then began to reap it. They then return to the beehive and do a waggle dance, which gives an idea of how much food is left and hence results in more exploitation or abandonment of the food source.
In case of load balancing, as the web servers demand increases or decreases, the services are assigned dynamically to regulate the changing demands of the user. The servers are grouped under virtual servers (VS), each VS having its own virtual service queues. Each server processing a request from its queue calculates a profit or reward, which is analogous to the quality that the bees show in their waggle dance. One measure of this reward can be the amount of time that the CPU spends on the processing of a request. The dance floor in case of honey bees is analogous to an advert board here. This board is also used to advertise the profit of the entire colony.
Each of the servers takes the role of either a forager or a scout. The server after processing a request can post their profit on the advert boards.
Server can choose a queue of a VS by a probability of px showing forage/explore behavior, or it can check for advertisements and serve it, thus showing scout behavior. A server serving a request, calculates its profit and compare it with the colony profit and then sets its px. If this profit was high, then the server stays at the current virtual server; posting an advertisement for it by probability pr. If it was low, then the server returns to the forage or scout behavior.

1. Initialise population with random solutions.
2. Evaluate fitness of the population.
3. While (stopping criterion not met) //Forming new population.
4. Select sites for neighbourhood search.
5. Recruit bees for selected sites (more bees for best e sites) and evaluate fitnesses.
6. Select the fittest bee from each patch.
7. Assign remaining bees to search randomly and evaluate their fitnesses.
8. End While.

Figure 7: Algorithm used in Honey bee technique (adopted from [7])

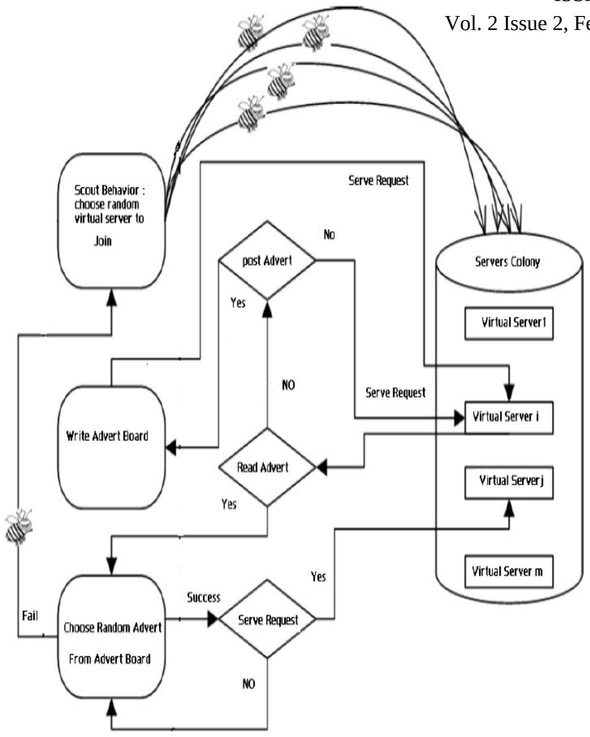

Figure 8: Server Allocations by Foraging in Honey bee technique (adopted from [14])

6.2 Biased Random Sampling

Here a virtual graph is constructed, with the connectivity of each node (a server is treated as a node) representing the load on the server. Each server is symbolized as a node in the graph, with each in degree directed to the free resources of the server.
Regarding job execution and completion, whenever a node does or executes a job, it deletes an incoming edge, which indicates reduction in the availability of free resource. After completion of a job, the node creates an incoming edge, which indicates an increase in the availability of free resource. The addition and deletion of processes is done by the process of random sampling.
The walk starts at any one node and at every step a neighbor is chosen randomly. The last node is selected for allocation for load. Alternatively, another method can be used for selection of a node for load allocation, that being selecting a node based on certain criteria like computing efficiency, etc. Yet another method can be selecting that node for load allocation which is under loaded i.e. having highest in degree. If b is the walk length, then, as b increases, the efficiency of load allocation increases. We define a threshold value of b, which is generally equal to log n experimentally.
A node upon receiving a job, will execute it only if its current walk length is equal to or greater than the

threshold value. Else, the walk length of the job under consideration
is incremented and another neighbor node is selected randomly. When, a job is executed by a node then in the graph, an incoming edge of that node is deleted. After completion of the job, an edge is created from the node initiating the load allocation process to the node which was executing the job.
Finally what we get is a directed graph. The load balancing scheme used here is fully decentralized, thus making it apt for large network systems like that in a cloud.

6.3 Active Clustering

Active Clustering works on the principle of grouping similar nodes together and working on these groups.
The process involved is:
A node initiates the process and selects another node called the matchmaker node from its neighbors satisfying the criteria that it should be of a different type than the former one. The so called matchmaker node then forms a connection between neighbors of it which is of the same type as the initial node.
The matchmaker node then detaches the connection between itself and the initial node.
The above set of processes is followed iteratively.

7. Conclusion

This paper explains the concept of load balancing, types of load balancing algorithms, general idea about dynamic load balancing algorithms and the different policies that can be used in it and gives an overall description of various distributed load balancing algorithms that can be used in case of clouds.

8. References

[1] Anthony T.Velte, Toby J.Velte, Robert Elsenpeter, Cloud Computing A Practical Approach, TATA McGRAW-HILL Edition 2010.
[2] Martin Randles, David Lamb, A. Taleb-Bendiab, A Comparative Study into Distributed Load Balancing Algorithms for Cloud Computing, 2010 IEEE 24th International Conference on Advanced Information Networking and Applications Workshops.
[3] Mladen A. Vouk, Cloud Computing Issues, Research and Implementations, Proceedings of the ITI 2008 30th Int. Conf. on Information Technology Interfaces, 2008, June 23-26.
[4] Ali M. Alakeel, A Guide to Dynamic Load Balancing in Distributed Computer Systems,IJCSNS International Journal of Computer Science and Network Security, VOL.10 No.6, June 2010.
[5]http://www-03.ibm.com/press/us/en/pressrelease /22613.wss
[6]http://www.amazon.com/gp/browse.html?node=201590011
[7] Martin Randles, Enas Odat, David Lamb, Osama Abu-Rahmeh and A. Taleb-Bendiab, A Comparative Experiment in Distributed Load Balancing, 2009 Second International Conference on Developments in eSystems Engineering.
[8] Peter S. Pacheco, "Parallel Programming with MPI", Morgan Kaufmann Publishers Edition 2008
[9] Rao, C.S., M. Naidu, K. Subbaiah and N.R. Reddy, 2007. Process migration in network of linux systems. Int. J. Comput. Sci. Network Security, 7: 213-219. http:// paper.ijcsns.org/07_book/html/200705/200705032.html.
[10]. Du, C., X.-H. Sun and M. Wu, May 2007. Dynamic scheduling with process migration. Proceedings of the 7th IEEE International Symposium on Cluster Computing and the Grid. May 14-17, IEEE Computer Society, Washington, DC, USA., pp: 92-99. DOI: 10.1109/CCGRID.2007.46
[11]. Barak, S.G.A. and R. Wheeler, 1993. The MOSIX Distributed Operating System: Load Balancing for UNIX. 1st Edn., Springer-Verlag, Inc., New York, pp: 221. ISBN-10: 0387566635
[12] R. D. Blumofe and D. S. Park. Scheduling large-scale parallel computations on network of workstations. *3rd IEEE Intl. Symp. High-Performance Distributed Computing*, Aug. 1994.
[13]. Walker, B. and D. Steel, 1999. Implementing a full single system image unixware cluster: Middle ware vs under ware. International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'99), June 1999, Monte Carlo Resort, Las Vegas, Nevada, USA., pp: 1-7.
[14] Pham D.T., Ghanbarzadeh A., Koç E., Otri S., Rahim S., and M.Zaidi "*The Bees Algorithm – A Novel Tool for Complex Optimisation Problems*"", Proceedings of IPROMS 2006 Conference, pp.454–461