A Problem Solving Environment for Combinatorial Optimization Based on Parallel Meta-heuristics

Rong Huang, Shurong Tong, Weihua Sheng, Zhun Fan

Abstract— **Computational grid offers a great potential solution to parallel meta-heuristics toward combinatorial optimization. However, it is quite difficult for specialists in combinatorial optimization to develop parallel meta-heuristics in extremely heterogeneous computational environment, starting from scratch without any toolkit. This paper presents a Problem Solving Environment for Combinatorial Optimization Based on Parallel Meta-heuristics (PSEPMH) to help specialists to harness heterogeneous computational resources and handle dynamic granularity control. PSEPMH requires specialist to decompose one problem into two sub-problems with divide-and-conquer framework just as generic sequential algorithm. Then compiler of PSEPMH generates mobile agent code that automatically forms adaptive multi-granularity parallel computing at runtime by cloning himself and distributing along dynamic, complex grid environment with the support of PSEPMH. Not only can PSEPMH relieve specialists' burden, but also make use of the computational resources more efficiently.**

I. INTRODUCTION

Combinatorial optimization [1] is the problem of finding the minimum or maximum of an objective function under some constraints, which is very important because of solving so many practical problems, such as traveling salesman, quadratic assignment, job-shop scheduling, etc. But it is proved that many combinatorial optimization problems are NP-hard, and it is unlikely able to find the efficient algorithms to obtain a globally optimal solution. Hence, the various algorithms categorized meta-heuristics [2, 3] such as Genetic Algorithm, Neural Network, Ant Algorithm, and Particle Swarm Optimization, etc., have been proposed in recent years to obtain near optimal solution whose measure is not too far from the optimum. The meta-heuristics represent effective and robust search algorithms and achieve good results in combinatorial optimization. Despite helps from the meta-heuristics, some large-scale problems still require a huge amount of computation time. Therefore, parallel programming techniques have been introduced generally and naturally to solve this kind of problems to reduce the processing time needed to reach an acceptable solution. All those meta-heuristics have something in common. They are inspired by nature having inherent parallelism, adaptability and weak synchronicity requirements. The idea of the ant system is based on the following observation. A colony of ants with limited capability is able to succeed in the task that is to collectively establish the shortest route between a source of food and their nest. Particle swarm optimization is inspired by social behavior of bird flocking or fish schooling. Whilst, Genetic Algorithm follows an intelligent evolution process for individuals. These algorithms are good candidates for parallel implementation [4, 5, 6]. Recently, the computational Grids have been the focus as promising infrastructure for high performance computing. Computational grids [7, 8, 9] provide one of the most attractive collaborative environments for running large compute-intensive applications by integrating geographically distributed, heterogeneous and inexpensive computational resources to work together in ways that were previously impossible. This paves the new way for parallel meta-heuristics to solve large-scale combinatorial optimization problems. However, It is extremely difficult for specialists of combinatorial optimization to develop parallel meta-heuristics in such high degree of heterogeneity Grid, starting from scratch without any tools kit. In this paper, we propose the Problem Solving Environment for Combinatorial Optimization Based on Parallel Meta-heuristics Toward (PSEPMH) which defines a set of appropriately layered abstractions and associated libraries, so irrelevant complexities are hidden from user, which allows specialists to concentrate on designing meta-heuristics, without having to become experts in computer science issues, such as networks, parallel computing or computational grid. The remainder of paper is organized as follows: Section 2 gives an explanation that the properties of mobile agent are naturally suitable for parallel computing and addresses how to form adaptive multi granularity parallel computing for meta-heuristics by cloning mobile agent. Section 3 presents architecture of PSEPMH. Finally, section 4 summarizes the paper.

II. MOBILE AGENT AND ADAPTIVE MULTI-GRANULARITY CONTROL ON GRID

A computational grid is a hardware and software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities, which distinguishes from conventional distributed computing by its focus on large-scale resource sharing and problem solving in dynamic, multi-institutional virtual organizations without considering the physical arrangement of computing resources. With the help of unprecedented computational power provided by computational grid, more efficient algo-

This paper has been supported by National Natural Science Foundation of China (NSFC) under the Grant 70462066 and Aeronautical Science Foundation of China under the Grant 03J53073.

Rong Huang is a PhD student at the School of Management, Northwestern Polytechnical University, Xi'an 710072, China. dr.huangrong@163.com

Shurong Tong is with the School of Management, Northwestern Polytechnical University, Xi'an 710072, China

Weihua Sheng is with the School of Electrical and Computer Engineering, Oklahoma State University, Stillwater 74074, USA

Zhun Fan is with the Department of Mechanical Engineering, Technical University of Denmark, Kgs.Lyngby, 2800 Denmark

rithms resolving large-scale combinatorial optimization can be potentially developed. Most of programs for combinatorial optimization have experimental character, that is, they always adjust parameters and code in order to try if they can get better performance or change the data for different problems. Nevertheless, in grid environment computing sources belong to different virtual organizations and scale is very large, so it is not realistic to let all members download the latest version of program and data, and manually update their own computers frequently. In proposed PSEPMH when program begin to run, mobile agents attached by all elements including Task, Data, Code, and Result as the whole are scattered along the network. At runtime, mobile agents distributed different hosts for parallel computing, so there is no problem about updating software. Furthermore, computational grid is concerned with large-scale pooling of computers, data, sensors, or people. Computational resources serve a lot of users, unlike other parallel systems monopolizing the computational resources. Therefore the computational resources change dynamically, even during the execution of program. The ability of mobile agent to react dynamically to adverse situations makes it easier to build fault tolerant behavior, especially in a highly distributed unpredictable system. Mobile agent [10, 11, 12, 13] is an autonomous software entity with the capability of roaming among computational nodes. Mobile agent can clone himself, then partition part of his task to the duplicate that moves to a new computational node for parallel computing. The following properties of mobile agent are naturally suitable for parallel computing in dynamic, complex grid environment.

1) Autonomy (to act on their own) gives opportunities for mobile agent to make a decision whether cloning himself and migrating duplicate to another computational node for part task or do the whole task by himself according to both environment and his goals at runtime.

2) Re-activity (to process external events) senses the change of environment such as appearing some computational resources, etc., then informs agent that there are some new computational nodes available, and the decision of migration can be made. Re-activity also accepts the synchronization signal from the duplicate. When the duplicate finishes the subtask, he will send the synchronization signal to the original agent. The original agent can perceive the signal, then collect and unify the result returned by the duplicate.

3) Pro-activity (to reach goals) achieves minimal completion time.

4) Co-operation (to efficiently and effectively solve tasks) is a primary advantage for parallel computing. Mobile agent can cooperate with the duplicate, while the duplicate and the original can also clone themselves again. The process is done recursively, which forms the multi granularity parallel computing.

5) Adaptation (to learn by experience) constantly reshapes mobile agent according to the dynamic environment and changes mobile agent's behavior based on experience via machine learning, knowledge discovery, statistical techniques, etc. According to the experience, mobile agent can know the capability and reliability of all computing nodes, then assign the appropriate computing tasks to them.

6) Mobility (migration to new places) is convenient vehicle for transporting subtasks from one computing node to another, by which subtasks can be distributed dynamically on different computing nodes at runtime, so that multiple computing nodes can work concurrently for the whole task.

Determining appropriate parallel granularity is one of the most important issues in parallel processing. The granularity of a task is informally used to indicate the size of the task. Too small granularity causes unnecessary overhead due to frequent context switching, creation and scheduling of tasks, while too large granularity leads to a loss of parallelism. Considering a lot of factors, such as the number of processors, network latency, designers can appoint granularity at compile time. This job is so difficult that many computer-aided software tools were developed to help designers make decisions. But the precondition is that designer must exactly know the number of computational resources and topology of them. But in dynamic, heterogeneous, opportunistic grid environment, there is not enough available information before programs run. Dynamic granularity control is good approach to resolve this problem. It adjusts granularity at runtime according to computational resources available and progress of working on problem, which demands the problem have potential to be flexibly partitioned. Fortunately, meta-heuristics have relatively loose, independent structures and weak synchronicity requirements, especially characteristics of population, which facilitates dynamic granularity control. Most of them can be generally represented by divide-and-conquer paradigms or its variations. The following addresses how to implement adaptive multi-granularity control in terms of divide-and-conquer paradigm by cloning mobile agent. Divide-and-conquer [14, 15, 16, 17, 18] algorithm partitions the complex problem into two separate, simpler sub-problems of roughly equivalent size and then combines these solutions into a solution for the whole. This process is applied recursively until the sub-problems are so simple to solve easily, which can be described generally as follows in pascal-like pseudocode:

```
   Procedure D_C(P)
1.  if (simple(P))            // tests if P is easy enough to solve directly
                               // instead of partitioning again
2.    then
3.      return base(P);       // give the solutions directly for the final
                               simple problems
4.    else
5.      begin
6.        (P1 ,P2) :=(f1(P),f2(P));   // partition the problem into
                                        sub-problems
7.            T :=combine(D_C(P1),D_C(P2));   //put the two
                                        sub-problems into together
8.        return(T);
9.      end
```

Where P is problem of the whole and P1, P2 are the partitioned sub-problems respectively.

D_C algorithm can also be represented as Figure 1 with

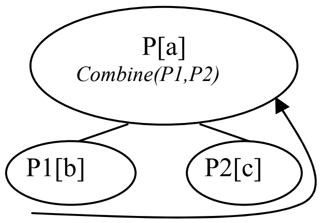

Fig. 1. D_C algorithm

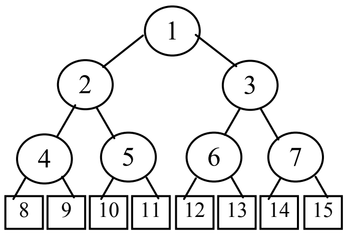

Fig. 2. Recursive solution

visiting nodes in order: b, c, a. Whilst the recursive solution to the problem is viewed as binary-tree with post-order traversal like Figure 2 whose leaves are the simple problems noted with squares. The path is 8, 9, 4, 10, 11, 5, 2, 12, 13, 6, 14, 15, 7, 3, 1. Let suppose that the problem is resolving at computational node 1. At the beginning of the process, only the left part of binary-tree is executed and the right part does not take part in the process. So there is an opportunity for mobile agent to clone himself and migrate the right branch sub-problem to another available computational node 2 for parallel computing, at the same time, computational node 1 is noticed that the right branch is treated as the simple problem, avoiding overlap computing. This process can be represented as in Figure 3. Furthermore, the origin or the duplicate can also either resolve the problem by himself or clone again. So adaptive multi-granularity can be formed at runtime. Adaptive multi- granularity control works beginning with very coarse task and progressively decreases granularity when some idle computational resources appear around. Various granularities are formed based on the number of computational resources and the capability of each computational resource. The computational resources having low capability constantly partition their task to other computational resources. Thus, the fewer the computational resources, the larger granularity, whilst, the higher capabilities, the larger granularity. This dynamic granularity control not only relieves designer's burden, but also makes use of the computational resources more efficiently.

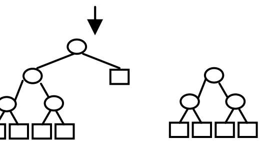

Fig. 3. Migrate the right branch

III. PSEPMH INFRASTRUCTURE

Grid integrates all kinds of network connected computational resources such as workstations, clusters, supercomuters, etc., into a global parallel computing infrastructure, which provides incredible computing powers for virtual organizations. If parallel meta-heuristics can be applied in Grid, it must be very promising for large-scale combinatorial optimization. However, it is extremely difficult to develop a parallel meta-heuristics in extremely heterogeneous computational environment, starting from scratch without any toolkit that provides basic service for managing resources. In this paper, the Problem Solving Environment for Combinatorial Optimization Based on Parallel Meta-Heuristics toward (PSEPMH) is introduced as a collaborative problem solving environment to help developer to implement parallel meta-heuristics for combinatorial optimization problem. PSEPMH can not only manage underlying heterogeneous computing resource but also form adaptive multi-granularity parallel application automatically using mobile agent in term of the principle presented in section 2. Developers can focus on enhancing meta-heuristics and tuning parameters for combinatorial optimization problem, just like designing sequential algorithm, regardless any issues about networks, parallel computing or computational grid. PSEPMH defines a set of appropriately layered abstractions and irrelevant complexities are hidden from higher layer, which is organized in hierarchic levels: grid service layer, mobile service layer and partition model layer. Grid service layer addresses issues of security, information discovery, resource management, data management, communication, and portability, which can efficiently harness highly heterogeneous and dynamic computational resources. Recently, several projects, such as Globus, Charlotte, Atlas and Legion are engaging in grid research. Globus toolkits are chosen in this paper, which abstract away the myriad complexities of heterogeneous environments and provide uniform protocols and APIs, with which mobile service layer can easily schedule the collection of computational resources. Mobile service layer is the central layer of the infrastructure, which built on top of grid service. By his ability to adapt to the prevailing circumstances, mobile agent will provide dynamic and robust services in grid environment. While, by his mobility, mobile agent can clone himself then attach subtask to the duplicate that migrates another computational node for parallel computing. Furthermore, the origin or the duplicate can also either resolve the problem by himself or clone again in terms of utilization, idle time, recover state, memory usage and different runtime events, etc. Therefore, the multi-granularity adapting dynamic partition at runtime is formed intelligently, which is transparent to PSEPMH user. Partition Model layer is responsible for translating formalized sequential code given by application layer into parallel computing code that will actually resolve the user' assigned combinatorial optimization problem with adoption

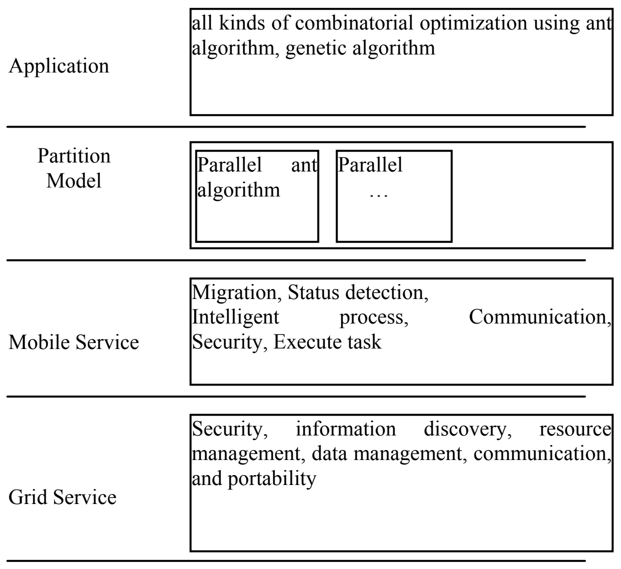
Application
all kinds of combinatorial optimization using ant algorithm, genetic algorithm
Partition Model
Parallel ant algorithm
Parallel …
Mobile Service
Migration, Status detection, Intelligent process, Communication, Security, Execute task
Grid Service
Security, information discovery, resource management, data management, communication, and portability

Fig. 4. PSEPMH infrastructure

of the mobile agent model. Application layer requires users to describe their combinatorial optimization problems with formalized Divide-and-conquer paradigm just as sequential code. Subsequently, various layers will be introduced in detail.

A. Grid Service Layer

The Globus Toolkit [19, 20] is a community-based, open architecture, open source set of services and software libraries that support grid applications, which are in use of hundreds of sites and by dozens of major Grid projects worldwide. In PSEPMH grid service layer involves information management, security, resources management, and data access services, as these services are elementary and essential to computational Grids and provide the foundation for building more advanced Grid services. The corresponding Globus services are Meta-computing Directory Service(MDS), Grid Security infrastructure(GSI), Grid Resource Allocation Manager(GRAM) and Globus Access to Secondary Storage(GASS).

B. Mobile Service Layer

Mobile service layer is responsible for forming multi-granularity parallel computing automatically by cloning mobile agent. In conventional way, an agent migrating from one host to another consists of a static part including the agent code and some static data, and a dynamic part including all the agent states such as program counter, stack, variables, etc., which describes at which point it has arrived, so that it is possible to reason about the new circumstances. In mobility of PSEPMH, there is a great difference from conventional approaches. Mobile agent clones himself and assigns subtask to the duplicate. The original one continues to run for the left subtask. On the other hand, the duplicate migrates another computational node with agent code and static data and subtask, etc., but not including dynamic part above-mentioned. There are two reasons for this. Firstly, in PSEPMH the dynamic part was left to the original one to keep on computing. While, the dynamic part is unnecessary for the duplicate to restart a new computing for the subtask. Secondly, the primary objective of Mobile Service layer is to improve parallelism by forming adaptive multi granularity parallel computing in order to minimize execution time. But it is time consuming to collect dynamic information according to the context. In PSEPMH, migration maybe occur frequently, so the dynamic part would not migrate in order to decrease the cost of migration as little as possible. In following scenario, when a mobile agent is computing with very heavy load in a computational node, a new more powerful computational node appears. The mobile agent will not migrate to the new node with the dynamic part for keep on computing. Instead, the mobile agent will partition his task, clone himself, then migrate the duplicate to the new node with static data (namely subtask, etc.) for parallel computing. In order to guarantee to absolutely realize the function of mobile agent layer, the following components must be included.

1) Migration: Migration is responsible for transferring the duplicate from one computational node to another with a transferable data structure that is not only for the duplicate to correctly run on a new computational node, but also for the original one to checkpoint to disk and recover later if the duplicate fails to finish the subtask.

2) Status detection: Status detection concerns itself with following aspects. First is to discover new computational nodes, to which a duplicate may be transported; second is to monitor the duplicate's activity status, checking whether the duplicate is active or inactive. If mobile agent finds the duplicate is inactive, alternative measures will be took. One is that the original agent will deal with the subtask formerly attached to the duplicate by himself. Another is that the duplicate will be recreated and migrated to another computational node with subtask and other information recovered from the checkpoint.

3) Intelligent process: Intelligent process revolves around two issues that are learning and making decision intelligently. The learning is to collect information regarding the benefits of the cloning and environmental properties, and statistically analyze them, then form the knowledge for future cloning. The making decision intelligently can be presented as follows. Reasoning about the task with respect to time restrictions, capability, resource requirements and learned knowledge, mobile agent makes a decision whether clone himself and transfer part tasks or do the whole task by himself. In this component a lot of intelligent techniques can be used, such as neural networks, Bayesian rules, etc.

4) Communication: Mobile agents would be little use if they were unable to communicate with other entities in a computational environment. In PSEPMH mobile agent can clone himself one or more times and the duplicate can also clone themselves again. Mobile agent just communicates with his duplicates, that is to check duplicates' status and get results worked out by clone agent, rather than directly communicate with the offspring created by his duplicates, which makes communication management simple and effective.

5) Security: Mobile agent basically provides two aspects of security. The first is the protection of host nodes from destructive mobile agents while the second is the protection of mobile agents from destructive hosts. Furthermore, in grid environment users share pool of computational resources, so users do not clearly know where their combinatorial optimization problems have been scattered by mobile agents. Therefore, a very important issue is that computational node that contains mobile agent cannot access the sensitive data of mobile agent despite it has privilege to kill the mobile agent process. The security of user's data must be guaranteed.

6) Execute Task: Execute Task is the most important component and other components provide auxiliary service to it, which control problem oriented computing for users' problems. It involves two Procedure D&C(P) and Procedure Partition. Procedure D&C(P) is variation of no-cursive form of D_C(P) mentioned-above appending two additional important functions. Firstly, it can adapt dynamic partition. Migrating the sub-problem will neither disturb previous computing result nor cause overlap computing. Secondly, when it is computing for user's problem, it can also indicate work states. That is, what part of problem has been finished so that Procedure Partition that is respond for dynamically partitioning subtask can decide what part of problem should be assigned to duplicate.

C. Partition Model Layer

After user describes their combinatorial optimization problem as Procedure D_C(P) where the Procedure f1, f2, simple, base and combine are implemented by designers according to practical problem, the problem still can't be parallelly resolved. Because it is just usual recursive procedure for sequential algorithm, so in order to form adaptive multi granularity parallel computing Partition Model layer is implement as compiler which tackles issues about translating Procedure D_C(P) into Execute Task Component of mobile agent layer with adoption of the mobile agent model.

IV. Process of Cloning Mobile Agent Based on PSEPMH

In Figure 5 MA is abbreviation of mobile agent. Host systems are various heterogeneous operation systems such as Linux, Windows, etc. Grid basic services that manage dynamic computational resources are implemented by Globus. MA environment is a software system that is registered as grid services over a network of heterogeneous computers. Its primary task is to provide an environment in which mobile agent can execute. At beginning MA original works on the whole task in MA environment 1, meanwhile, sends the request to UDDI Register now and then, querying whether a new MA environment can be found. If UDDI Register knows a new MA environment 2, it will give the reply. Then, MA environment 1 communicates with MA environment 2 further for more information. MA original will compare the cost of

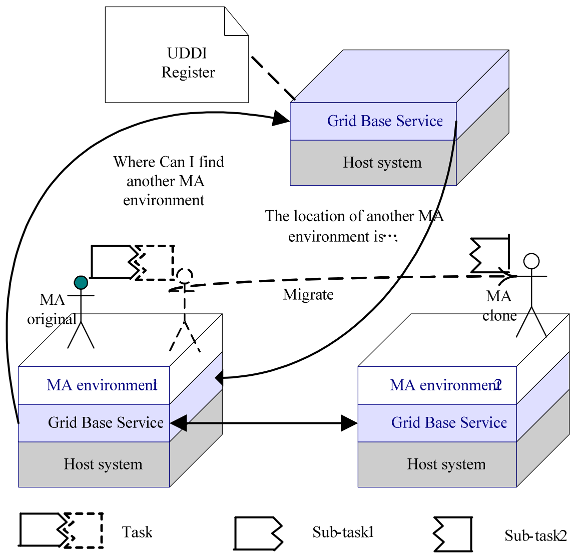

Fig. 5. Process of cloning mobile agent based on PSEPMH

cloning himself and assigning the subtask to the duplicate with that of working on the whole task by himself. If he thinks the former method can shorten the execution time, he will clone himself, then apportion half of task to the clone agent that migrate to MA environment 2 for parallel computing.

V. Summary

First, we explained why the properties of mobile agent are suitable for parallel computing in dynamic heterogeneous grid environment. In particular, we addressed the principle of adaptive dynamic multi-granularity parallel by cloning mobile agent. Second, based on the principle, we proposed PSEPMH infrastructure which can help users to implement parallel meta-heuristics for combinatorial optimization problem. With support of PSEPMH, when a new computing resource appears, application can adaptively partition part task to the new computing resource at runtime. Meanwhile, repartitioning the task can guarantee the previous computing result available, that is, repartitioning the task doesn't abandon the previous effort because of the reassigning the task. The process of partitioning task, cloning and migrating mobile agent is transparent to users. PSEPMH just requires users to decompose one problem into two sub-problems with divide-and-conquer paradigm just as generic sequential algorithm. Therefore PSEPMH not only relieves users' burden, but also makes use of the computational resources more efficiently.

VI. Acknowledgments

This paper has been supported by National Natural Science Foundation of China (NSFC) under the Grant 70462066 and Aeronautical Science Foundation of China under the Grant 03J53073. Many thanks to the reviewers for their contributions to improve this manuscript.

References

[1] C.H. Papadimitriou and K. Steiglitz. Combinatorial Optimization, Algorithms and Complexity. Prentice-Hall, Englewood Cliffs, 1982

[2] Osman, I. H. and Kelly, J. P. (1996a) (eds) Meta-Heuristics: Theory and Applications, Kluwer Academic Publishers, Norwell, MA, USA

[3] F. Ben Abdelaziz, S. Krichen, and J. Chaouachi. Meta-heuristics: Advances and trends in local search paradigms for optimization, chapter A hybrid heuristic for multi-objective knapsack problems, pages 205–212. Kluwer Academic Publishers, 1999

[4] W. Crompton, S. Hurley and N.M. Stephens, ”A Parallel Genetic Algorithm for Frequency Assignment Problems”, Proc. of IMACS SPRANN'94, pp81-84, 1994.

[5] Petty, C.B., Leuze, M.R., and Grefenstette, J.J. (1987). A Parallel Genetic Algorithm. Proceedings of the Second International Conference on Genetic Algorithms, pp. 155 - 161.

[6] E.Alba and M.Tomassini. Parallelism and evolutionary algorithms. IEEE Transaction on Evolutionary Computation,6(5):443-462, October 2002.

[7] I. Foster, C. Kesselman, and S. Tuecke. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. International Journal of Supercomputer Applications, 2001.

[8] I. Foster and C. Kesselman (editors). The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufmann Publishers. July 1998. 17

[9] I. Foster, C. Kesselman, J. Nick, and S. Tuecke. The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. Technical report, Open Grid Service Infrastructure WG, Global Grid Forum, June 2002.

[10] D.Kotz, R. Gray, and D. Rus, Future Directions for Mobile Agent Research, IEEE Distributed Systems Online, vol. 3, no. 8, 2002.

[11] G. Di Marzo, M. Muhugusa, and C.F. Tschudin. A Survey of Theories for Mobile Agents. World Wide Web Journal, pages 139–153, 1998.

[12] A. Fuggetta, G. P. Picco, and G. Vigna. Understanding Code Mobility. IEEE Trans. on Software Engineering, May 1998.

[13] L. Cardelli. Abstractions for Mobile Computations. In Secure Internet Programming, number 1603 in Lecture Notes in Computer Science, pages 51-94. Springer, 1999.

[14] M. J. Atallah, R. Cole, and M. T. Goodrich, Cascading Divide-and-Conquer: A Technique for Designing Parallel Algorithms, SIAM Journal of Computing, 18 (1989), pp. 499-532.

[15] G. Even, J. Naor, S. Rao, and B. Schieber. Divide-and-conquer Approximation Algorithms via Spreading Metrics. In 36th Annual Symposium on Foundations of Computer Science (FOCS96) , pages 62-71, Burlington, Vermont, 1996. IEEE Computer Society Press.

[16] K. S. Gatlin and L. Carter. Architecture-cognizant Divide and Conquer Algorithms. In SuperComputing '99. University of California San Diego, Computer Science and Engineering Department, November 1999.

[17] Renate Knecht. Implementation of Divide-and-conquer Algorithms on Multiprocessors. In Parallelism, Learning, Evolution Workshop on Evolutionary Models and Strategies, pages 121-136, Neubiberg, Germany, Springer-Verlag,1989.

[18] V.M.Lo, S.Rajopadhye, S.Gupta, D.Keldsen, M.Mohamed and J.Telle. Mapping Divide-and-conquer Algorithms to Parallel Architectures. In International Conference on Parallel processing. Vol. III, pages 128-135, CRC Press,1990

[19] I. Foster and C. Kesselman. Globus: A Metacomputing Infrastructure Toolkit. International Journal of Supercomputing Applications, 11(2):115–128, 1997

[20] I.Foster and C. Kesselman. The Globus project: A Status Report,Future Generation Computer Systems 15 (1999) pp 607-621