ISSN: 2150-8194 Vol. 2, No. 7, July 2010

Scientific Research

Intelligent Information Management

Editor-in-Chief: Dr.Bin Wu

ISSN: 2150-8194

www.scirp.org/journal/iim

Journal Editorial Board

ISSN: 2150-8194 (Print), 2150-8208 (Online)

http://www.scirp.org/journal/iim

Intelligent Information Management, **2010, 2, 391-436**
Published Online July 2010 in SciRes (http://www.SciRP.org/journal/iim/)

TABLE OF CONTENTS

Volume 2 Number 7 July 2010

Intelligent Information Management (IIM)

Journal Information

SUBSCRIPTIONS

The *Intelligent Information Management* (Online at Scientific Research Publishing, www.SciRP.org) is published monthly by Scientific Research Publishing, Inc., USA.

Subscription rates:

Print: $50 per issue.
To subscribe, please contact Journals Subscriptions Department, E-mail: sub@scirp.org

SERVICES

Advertisements
Advertisement Sales Department, E-mail: service@scirp.org

Reprints (minimum quantity 100 copies)
Reprints Co-ordinator, Scientific Research Publishing, Inc., USA.
E-mail: sub@scirp.org

COPYRIGHT

PRODUCTION INFORMATION

For manuscripts that have been accepted for publication, please contact:
E-mail: iim@scirp.org

Intelligent Information Management, **2010, 2, 391-397**
doi:10.4236/iim.2010.27048 Published Online July 2010 (http://www.SciRP.org/journal/iim)

Assessment of a Proposed Software Design for the Solution of Multi-Phase Mechanics Problems on Networked Laptops

Richard Harris, Thomas Impelluso
Department of Mechanical Engineering, College of Engineering, San Diego State University, San Diego, USA
E-mail: *tom.impelluso@sdsu.edu*
Received October 19, 2009; *revised April* 2, 2010; *accepted May* 18, 2010

Abstract

This paper presents the design of a computational software system that enables solutions of multi-phase and multi-scale problems in mechanics. It demonstrated how mechanicians can design "process-driven" software systems directly, and that such efforts are more suitable in solving multi-phase or multi-scale problems, rather than utilizing the "data-driven" approaches of legacy network systems. Specifically, this paper demonstrates how this approach can be used to solve problems in flexible dynamics. Then it suggests a view of mechanics algorithms as 'state equilibrium' enforcers residing as servers, rather than as computer programs that solve field equations. It puts forth the need for identical input/output files to ensure widespread deployment on laptops. Then it presents an assessment of the laptop platform. A software system such as the one presented here can also be used to supply virtual environments, animations and entertainment/education software with physics.

Keywords: Software Design, Multi-Phase Mechanics Problems, Networked Laptops

1. Introduction

The finite element (FE) method is used in computational mechanics to analyze the deformation of materials. In the FE method, the field differential equations are first converted into integral equations; then, the domain is discretized with interpolation functions (meshed). Algebraic algorithms (such as conjugate gradient methods or direct solvers) are used to solve the resulting system for the displacement variables, forces and other internal phenomena. While the method can be extrapolated to account for the motion of objects and their contact with each other, the fundamental algorithm—as deployed in solid mechanics—is to solve for the deformation of objects subjected to loads and sufficient boundary conditions which ensure a non-singular system of algebraic equations.

The multi-body dynamics method is the method used in computational mechanics to analyze the motion of rigid objects (as opposed to the FE method which is focused on the deformation of stationary objects). The motions of objects are constrained by formulating various types of joints—e.g.: revolute, ball, translational—as geometric constraints on the equations, and then using these constraint equations to reduce the degrees of freedom of the system. Algorithms such as Runge-Kutta are used to update the system configuration (position, velocity, acceleration and forces), incrementally as a time-stepping solver.

The term 'multi-phase' is a generalization of the techniques used to solve coupled problems in mechanics such as solid/fluid interaction or flexible linkages.

In January of 2003, the U.S. National Science Foundation Advisory Committee on Environmental Research and Education published a report [1] that identified cyber-infrastructure (CI) as a suite of tools and research essential to the study of engineering systems. The CI describes research environments supporting advanced data acquisition, data storage, data management, mining, visualization and other computing and information processing services. In scientific usage, the CI and its ancillary technologies enable solution to the problem of efficiently connecting data, computers, and people with the goal of enabling derivation of novel scientific theories and knowledge. This, then, enables researchers and de-

velopers to port data between various large-scale software systems. In general, the CI encompasses the low-level inter-process communication facilities that are bundled with operating systems, and the higher-level applications which extend them and build upon them.

Currently, some use these higher-level applications, such as Globus, to automate the transfer of data from one mechanics algorithm (say, finite element) to another (multi-body dynamics) to solve for coupled problems. All such methods adhere to a sense of 'data' as driving the solution; and this research demonstrates that there is a better approach.

Continuing, the term 'inter-process communication' encompasses the fundamental technologies of the CI. These include techniques to enable codes to replicate themselves, instantiate themselves on other computers, enable processes to communicate with one another and to control each other. This paper will step through the design of a software system built upon fundamental technologies of the CI (the low-level—but not in the pejorative sense—tools upon which higher order applications are built) to solve problems in flexible dynamics. This research advocates that such an approach can and should be taken in the solution of multi-phase and multi-scale problems by computational mechanicians.

2. Distributed Computing and Multi-Phase Mechanics

2.1. Distributed Computing

Distributed computing began in the early 1970's with the arrival of minicomputers and ubiquitous networks. While the smaller computers were less powerful than their larger mainframe cousins they were also much cheaper. An easy way of scaling a computation was to buy more of such machines and distribute the problem over the multiple CPUs. Distributed computing differs from other parallel computation models in that the CPU nodes are autonomous and that all connections and procedure calls are accomplished by message passing or socket communication.

Software for managing memory communication and process management can either be "homemade", commercially purchased or found within the open source community. Typical applications include: "smart instruments", teraflop desktops, collaborative engineering, and distributed supercomputing. The system consists of packages such as: resource allocation managers, authentication services, network analysis monitoring systems, and secondary storage systems. Globus, an open source grid computing library, enables researchers to share large scale software systems across a "global" network. Such large-scale software systems include heart models, physiology models, manufacturing models and many others.

2.2. Multi-Phase Mechanics

Three groups developed the analysis of multi-phase mechanical systems in the 1970s: Northwestern University, California Institute of Technology, and Lockheed Palo Alto Research Labs (by J. A. DeRuntz, C. A. Felippa, T. L. Geers and K. C. Park). This initial work focused on different applications and evolved into different problem-decomposition techniques. For example, Belytchko and Mullen at Northwestern considered node-by-node partitions whereas Hughes and Liu at Cal Tech developed what eventually became known as element-by-element methods. The Lockheed group began its investigation in the underwater-shock problem for the Office of Naval Research (ONR) in 1973. In this work an FE computational model of the submarine structure was coupled to Geers' "doubly asymptotic" boundary-element model of the exterior acoustic fluid, which functions as a silent boundary. In 1976 Park developed a *staggered solution procedure* to treat this coupling. Belytchko and Mullen give a more detailed historical overview as well as a summary of the methodology in the aforementioned work.

The formulation of the equations of motion for a flexible multi-body system invites coupling in many ways. Geradin and Cardona [2] formulated an approach in which the inertial frame becomes the reference frame. Coupling of FE and MD has also been done with linear [3] and non-linear [4] FE methods. Contact has also been introduced using unilateral constraints [5] or continuous contact forces [6]. The availability of state variables in multibodies allows for different control paradigms in the framework of vehicle dynamics, biomechanics or robots [7]. The coupling of fluid and structural dynamics allows for solid fluid interactions in which there are large rotations of system components [8].

2.3. Distributed Computing for Multi-Phase Mechanics

The previous methods used to solve multi-phase problems do not deploy existing distribution methodologies. These methods do solve certain coupled problems: solid/fluid or flexible multi-bodies, however, they do not exploit the technologies of the CI. They are not scaleable and lack the ability to be readily extended to encompass new physics modules. Once they are running, new physics modules cannot join process groups, nor can modules leave groups once they are no longer needed. The systems are not fault tolerant: certain physics algorithms cannot be restarted with new parameters without having to restart the entire system. Computational modules cannot be easily targeted for the most efficient platforms. The results cannot be easily delivered to clients that might need them, including, for example, near-real time,

fault tolerant, immersive visualization environments. It is still difficult to port data from one physics process group to another and when this does happen, it relies on large-scale network packages.

To begin to address these issues, it is first necessary to categorize computations. Computational software can be broken down into two categories: process driven vs. data driven. Software packages for seismology or bioinformatics must manipulate data (object oriented coding: C++, Ada). Software packages for physics simulations should manipulate processing (processing oriented coding: FORTRAN, C).

3. Background and Proof-of-Concepts

As segue to the effort reported here, two prototypes are first discussed.

3.1. Phase 1

The preliminary platform reproduced a real stress-testing machine as a 3D model in a simulated computational environment. And then it created an interface to drive both the real stress testing machine and the virtual stress-testing machine simultaneously [9]. The functionality of the system is described herein.

First, in the client/server paradigm of inter-process communication, a server is simply a code that waits for remote communication. A client is a process that requests help from a server.

In this test-bed, then, a client requests a FE analysis of an object. The client connects to a server. As indicated in **Figure 1**, the server parses the control client request and **fork()/execs()** a process manager (PM). (In the parlance of inter-process communication, when a process **forks()** itself, it basically spawns a duplicate copy of itself—much like the replication of a strand of DNA. Such a dual process has access to the same memory, files and peripheral devices s the parent.) The child (or fork'd) process then **execs()** another process. (**Exec()** is a method in which a process replaces itself with a second process such that this new child process retains access to external sources and devices.) This exec'd child process will be called a process manager (PM).

The PM extracts all pertinent field information received from the remote client process. It then creates the shared memory in which to store the geometry of the specimen under deformation, and then creates the semaphores to ensure process regulation (see traffic lights in **Figure 2**). The PM then **fork/execs** two processes: the finite element process (FE) to perform the analysis of the deformation and a reader process (RD) whose role is to

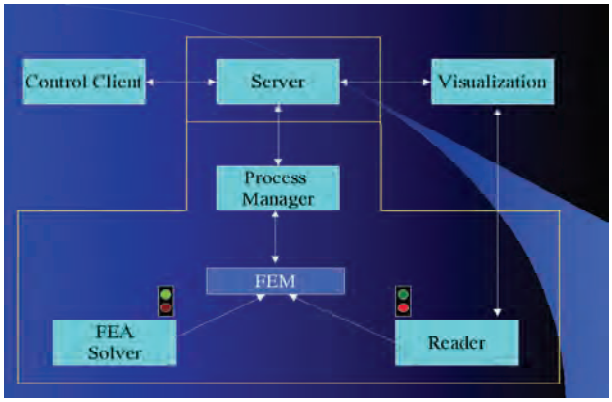


Figure 1. Network schematic for virtual stress testing machine.

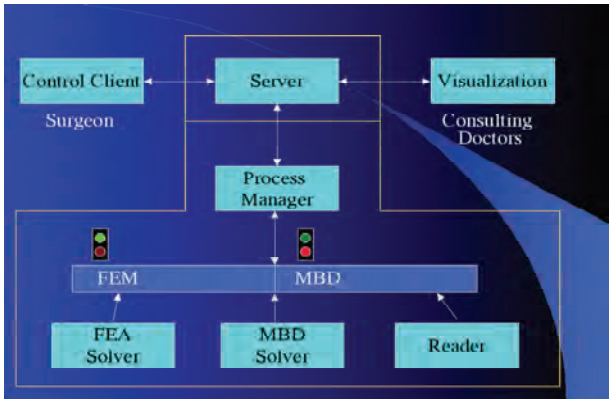


Figure 2. Network schematic for phase 1 design.

relay data back to any client that wishes to view the results. The semaphores (semaphores are simply integer flags that facilitate control and access to data when two competing codes are both trying to modify the data) control access to the shared memory ensuring that all clients view the same updated configuration of the specimen. In this scenario an unlimited number of clients can request visualization of the physical phenomenon and the reader process manages this.

Some preliminary data is needed to set up the "virtual" experiment: material properties, geometric properties, numerical mesh data, location of boundary conditions. This process will first read the initial input data to describe the nature of the virtual experiment, and then continuously read the secondary input data to drive the virtual experiment. As the experiment runs, data is used to drive it: in this case, the value of the applied forces or displacements. The analysis produces several output parameters—such as stress or strain—which are then visualized. Thus, there will be a visualization client (computer program to view the scene), a control client (to interact with the scene), a general server that will orchestrate network connections, and, finally, a physics server to run the virtual experiment. This approach, using shared

memory was the first step taken in the next phase.

3.2. Phase 2

The goal of this platform was to expand the aforementioned test-bed to solve for two coupled phenomena: (multi-body dynamics) MD and (finite element) FE (labeled, respectively, as MBD and FEA in **Figure 2**.

Consider the solution scenario as indicated in **Figure 2**. A control client creates a data set to describe the parameters of a multi-phase problem. Then it connects to a server capable of moderating a solution. The server receives the request from the control client. The server **forks()** and **execs()** a process manager (PM). The PM creates a shared memory arena for processing the geometry and physics data, and deposits descriptive geometries pertinent to the FE and MD methods, in appropriate segments of the shared memory. The PM also creates the semaphores that regulate access to the shared memory. The PM then **forks()** and **execs()** the FE and MD processes to analyze the coupled problem of 2D flexible linkages. The PM now orchestrates the solution using the FE and MD methods, incrementally—first Newton-Raphson and then Runge-Kutta. At each time step, one method produces a result that is passed to the next method. Iteration between the two methods brokered by the PM provides the final solution.

This project had its deficiencies. First, while the use of shared memory proved fast and efficient, it did require all processes to reside on the same computer. Each process—MD and FE—has its own implementation issues (convergence, stability, etc.) and if more processes were to be added (each with their own issues), there would still only be one CPU to handle all numerical methods. Further, if the server node faulted, the entire system would fault. Finally, this approach is not easily distributed and does not readily allow for worldwide remote participation, which is the objective of this research.

4. Current Prototype

The operation of the current system is presented in **Figure 3** (with the addition of a macroscopic fluids mechanics process to demonstrate an *extended* vision of the system).

First, a user creates (using a text editor) a data frame, which defines the problem: mesh data, material properties, and so on. The frame is delivered to a server (connection 1 in the **Figure 3**) from a client control process.

Upon receipt of "the problem description", the server "**fork/execs"** a process manager to broker the solution (connection 2).

The PM reads the data frame (to extract what mechanics processes are requested for the coupled problem) and then instantiates the appropriate processes (finite element and multi-body dynamics, in this case) on targeted hardware (a dual CPU for the FE's solver and a single CPU machine for the dynamics solver) (connection 3). Those processes then each become a server (inverting the client/server paradigm), and the PM then connects as a client to each of these physics servers: its role now is to broker the solution and relay data between various modules.

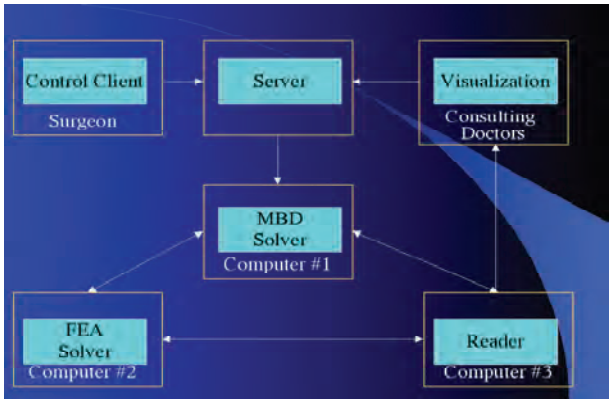


Figure 3. Revised network schematic for multi-phase mechanics.

The PM delivers the results to a visualization workstation for viewing (connection 4). Once the paradigm is inverted, the visualization process is a client of the PM and the PM is a client of the server.

More specifically, the user requests a multi-body dynamics analysis of the deformable linkage system. The FE process conducts an analysis at a given time step, producing displacement, stresses, strains and other data that is delivered to the PM. The PM extracts the data requested by the visualization client. It also extracts and parses the data that is needed for the MD process to function. The MD code updates the time step and conducts the Runge-Kutta method to find the next configuration of the links: position, velocities, accelerations and forces. Upon completion it delivers the data to the PM and then halts. The PM delivers the data that is requested by the analyst and then it delivers the data requested by the FE coded. At this point the FE code commences again and the system continues.

While the system is running, the user can request a deformation analysis to be added. For example, the PM can instantiate an FE code on a new target (one target for each mesh) and set up the network communication to allow the FE and MD codes to communicate. At any point, the user can decide that an FE analysis of any link is no longer necessary. At any point, other processes can join the group.

5. Assessment

The system was assessed and verified using two numeri-

cal process multi-body dynamics and finite element analysis; both codes were two-dimensional systems. The data set was a model of a human lower leg consisting of, femur, tibia, fibia, and homogenized foot (in which there was, in this case, one deformable body for the foot to avoid complexity at this stage). The multi-body dynamics system consisted of three linkages; femur, tibia-fibia, foot and three revolute joints; hip, knee, ankle. In the initial configuration, the femur was horizontal to the ground plane and the tibia-fibia perpendicular to the femur, the leg was then allowed to free swing under gravity.

There were three data sets for the finite element code—femur, tibia-fibia, foot—each consisting of approximately 1000 nodes and 1000 elements. The intent of these models was not to examine the intricate working of human motion and the resulting forces; rather, it was to demonstrate the feasibility of a distributed system. Qualitative evidence of the analysis is offered in **Figures 4-6** along with analyses obtained using large scale finite element package: MSC Marc/Mentat and Adams.

The PM accepted data from the user interface started the selected numerical processes on the targeted machines and successfully passed the necessary data between numerical processes and the user display. Initial experimentation was completed in a UNIX environment using the IRIX operating system.

This system was expanded to function on Linux operating systems. Linux test beds were HP Compaq nx9420 laptops. The transition from UNIX to Linux operating systems including a transition from wired LAN to a WiFi network.

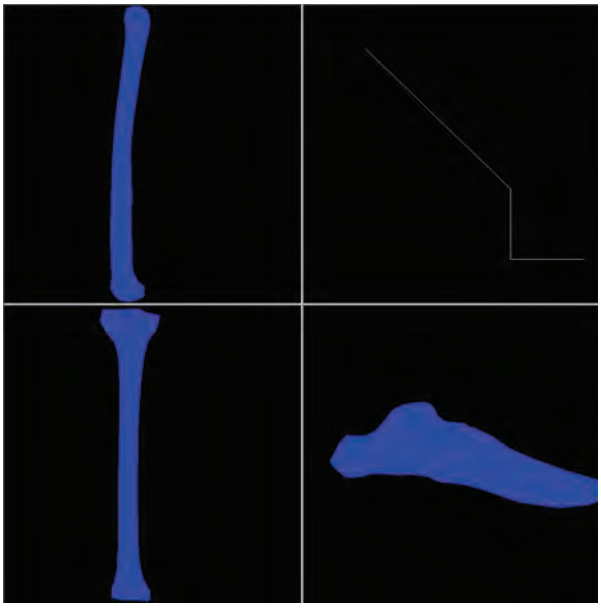


Figure 4. Visualization of three FE, and one MBD process at time 0.

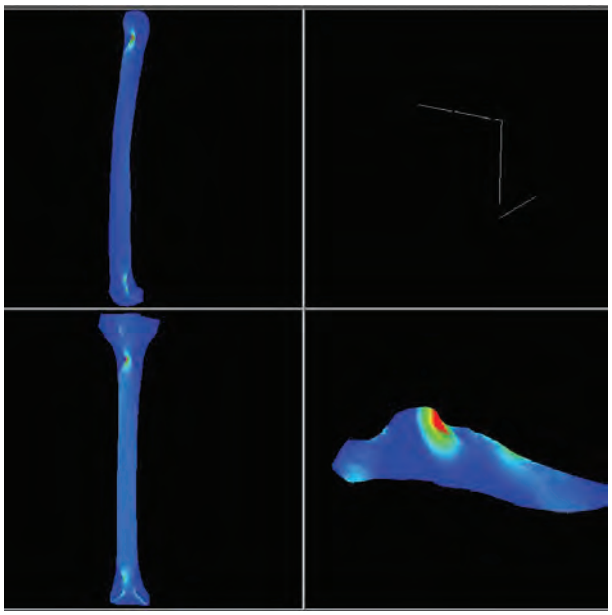


Figure 5. Visualization of three FE, and one MBD process at time 5.0.

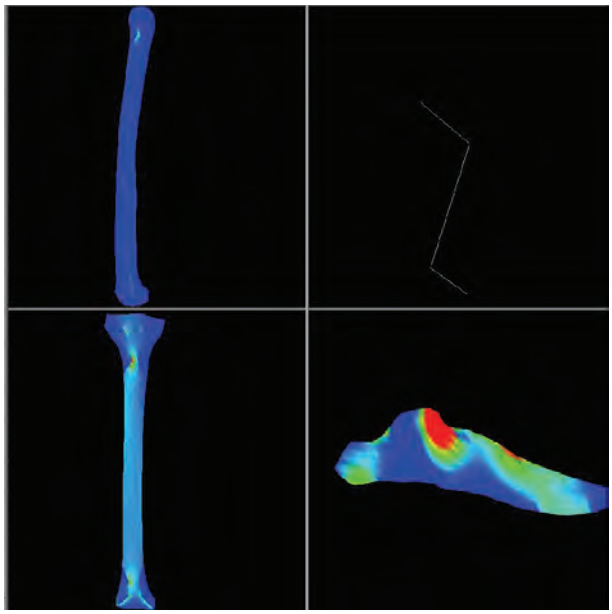


Figure 6. Visualization of three FE, and one MBD process at time 10.0.

Quantitative system analysis was performed by timing the period the PM was brokering data. In the coupled MD FE system the MD process provides the initial data as such the PM will start the MD code first. The PM was timed from the point the MD process was instantiated to the point the process was terminated.

Sixty simulations were performed using three HP nx9420 laptops running; Fedora Core III, Ubuntu, and Debian flavors of the Linux operating system. One ma-

chine served as the PM, GUI, and Visualization client, the other two machines performed the numerical analysis. Each machine served as the GUI, Visualization, and PM for twenty simulations. The PM randomly selected which remaining machines for each numerical process. The fastest and slowest recorded processing period was removed from the dataset. When the computers were networked on a wired LAN the remaining fifty-eight data points were averaged, yielding 15.182 seconds with a standard deviation of 0.023 seconds. When the same experiment was performed on an 802.1 G wireless network the average time was 15.201 with a standard deviation of 0.019 seconds.

This minor difference in processing time can be attributed to the relatively small amount of data that was passed from each of the finite element processes to the PM. Datasets were on the order of 30 Kbytes, further experimentation with larger datasets should be completed to determine where the time discrepancy originates, and if the relationship between transmission time and data size were linear as it would be expected.

The use of LINUX platform demonstrated that an open source operating system was robust and compliant to POSIX procedures to allow a system that was developed on a UNIX platform to be easily ported to less expensive machines. The low cost and abundance of hardware options makes PC's the ideal machines to operate a distributed numerical network.

6. Future Plan

During the past two decades, advances in modeling (FE, MD, and CFD) were made through the use of sophisticated graphical user interfaces. These interfaces allowed for rapid model creation, mesh generation, and installation of boundary and initial conditions. However, these same interfaces now pose a problem in an era evolving research needs. The use of such legacy software results in the creation of lock files, history files, trace files, and all sorts of other peripheral data files that are required to run certain codes. As a result, it is cumbersome to re-run a large analysis a year later, let alone feed the data from an FE code into an MD code or into a CF code. It is exceedingly difficult to use these large-scale codes for multi phase problems such as the analysis of flexible linkages. Just as difficult, also, is the use of such codes to model effects across length scale gaps, e.g., combined use of molecular dynamics and continuum mechanics. An alternative must be sought—and Globus attempts to address these needs by being a platform to facilitate inter-process communication at a very high level, and, generally, for legacy packages. While it can also be used to integrate lower, level, simpler codes, such a legacy system might be inimical to the needs of computational mechanicians; further, there might simply be too much overhead with such a system.

Consider the spirit of the open source operating system, wherein an operating system command is a data converter e.g., an operating system command can covert a *.doc file into a *.pdf file. A physics command should operate on a data set the same way. The only command line options for a physics code hold be the names of the input and output files. The input.dat and output.dat file formats should be conceptually identical, while all fields need not be filled in; the format should be identical, enabling an FE code to read both input and output files. As a result of adhering to strict formatting rules the FE code can also be viewed as a data converter: it should be able to read a file and ensure it represents the equilibrium state. A user, for example, should be able to edit an output file (modify a stress), and feed it back into the same FE code.

In fact, as FE solvers have now moved toward iterative methods for solutions, abandoning direct solvers, this makes much more sense. In such solvers a 'state' can be read, concomitant with all field variables: stress, strain, displacements, forces, velocities and so on. The FE solver then simply inspects the data set, modifies it to ensure equilibrium, and then writes it out again for the next physics server to read.

This coding philosophy now promotes a new view of physics coding wherein an iterative FE code is not just a large-scale analytical tool, but a simple equilibrium enforcer on a data set. In such cases, one should consider the physics processes (FE, MD and others) to be processes that descend upon a data set, inspect it to see if the process has rights to act on the data, and then modify the data to ensure physical equilibrium.

The same philosophy then also holds for multi-body dynamics wherein such a code should be viewed as not just solving the entire motion path, but ensuring that at each times step, equilibrium is assured; *i.e.*: that the Newton-Raphson implementation has converged for a time-step of Runge-Kutta.

This view quickly enables computational mechanics algorithms to be deployed as simple computational blocks from which a multi-phase and multi-scale server can be assembled—worldwide researchers, without access to legacy hardware or software, can contribute to domain knowledge readily.

While contact processing (CX) has not yet been implemented, it remains a critical issue which will make the described software system more compelling.

Contact between finite element meshes has historically been resolved by calling a subroutine within an FE code. In the spirit of the physics coding described in Section 4.1, a new approach is advocated to solve for contact between deformable objects. Rather than relegating contact analyses to a subroutine within a, contact (CX) should be elevated to become a process of its own, enabling those in the field of geometric visualization to work with those

in mechanics computation.

Suppose one mesh in a process group has millions of elastic finite elements, while another mesh has several thousand inelastic elements. Each FE mesh can be analyzed on optimal hardware for that algorithm, while the CX algorithm runs on yet another. This can enhance optimization and load balancing as meshes with specific properties can be targeted for certain architectures. Functionally, after an equilibrium update, data from each FE process is delivered to CX process by the PM. The CX process will inspect the data and provide contact forces to the PM. The PM then delivers the appropriate contact force that each mesh (FE process) needs to prevent penetration.

7. Summary

While the interfaces provided by commercial software have facilitated analysis of specific single-phase or single-scale problems, they are not open to the lightweight, distributed, fault-tolerant needs that the CI can enable. Further, the emerging systems to allow for software integration seems predicated on the continuing value of large-scale legacy systems: they are data-driven. As the appendix shows, the low level techniques of the CI are not that complicated. This paper advocates a return to simplicity in mechanics software design for multi-phase problems. This simplicity is finally realized with the suggested software design.

Many researchers are conducting analyses of multi-phase and multi-scale problems. Packages such as Globus enable this research by enabling existing large-scale legacy systems to be stitched together. This proposed effort builds the system from the bottom. This system is topologically flat and scaleable with the potential for high fault tolerance. It resists legacy in one software "house" and enables a distributed contribution to the solution of such problems for once the interfaces are defined, any one module can be written by any researcher and a server can be notified that a given machine, anywhere in the world, can contribute to an analysis. If, however, the climate moves toward the use of Globus, the results of this work will enable computational scientists to quickly assimilate the technologies of CI and contribute to a new dialogue.

Once again, one day, a package such as Globus might evolve to satisfy such needs. But this author advocates that it is in the interest of computational mechanicians to take the time to understand the intricacies of software needed to integrate the modules of mechanics.

8. References

[1] Atkins, NSF Report. http://www.communitytechnology.org/nsf_ci_report/

[2] M. Geradi and A. Cardona, "Flexible Multibody Dynamics: A Finite Element Approach," John Wiley and Sons, England, 2001.

[3] J. Goncalves and J. Ambrósio, "Complex Flexible Multibody Systems with Application to Vehicle Dynamics," *Multibody System Dynamics*, Vol. 6, No. 2, 2001, pp. 163-182.

[4] J. Ambrósio and P. Nikravesh, "Elastic-Plastic Deformation in Multibody Dynamics," *Nonlinear Dynamics*, Vol. 3, No. 2, 1992, pp. 85-104.

[5] F. Pfeiffer and C. Glocker, "Multibody Dynamics with Unilateral Contacts," John Wiley and Sons, New York, 1996.

[6] H. M. Lankarani and P. E. Nikravesh, "Continous Contact Force Models for Impact Analysis in Multibody Systems," *Nonlinear Dynamics*, Vol. 5, No. 2, 1994, pp. 193-207.

[7] M. Z. Valesek and Z. Sika, "Evaluation of Dynamic Capabilities of Machines and Robots," *Multibody System Dynamics*, Vol. 6, No. 2, 2001, pp. 183-202.

[8] H. Moller and E. Lund, "Shape Sensitivity Analysis of Strongly Coupled Fluid-Structure Interaction Problems," *Proceedings of the* 8th *AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, Long Beach, CA, 2000, pp. 4823-4833.

[9] R. Harris and T. Impelluso, "Virtual Stress Testing Machine and the Cyber-Infrastructure," *Engineering with Computers*, Vol. 24, No. 2, 2008, pp. 107-117.

Intelligent Information Management, **2010, 2, 398-405**
doi:10.4236/iim.2010.27049 Published Online July 2010 (http://www.SciRP.org/journal/iim)

Scientific Research

A Unified Monitoring Framework for Distributed Environment

Xiaoguang Wang[1], Hui Wang[2],Yongbin Wang[1]
[1]*Computer School, Communication University of China, Beijing, China*
[2]*Information Engineering School, Communication University of China, Beijing, China*
E-mail: *xgwong*0815*@gmail.com*
Received April 12, 2010; *revised May* 27, 2010; *accepted June* 28, 2010

Abstract

Distributed computing is a field of computer science that studies distributed systems. With the increasing computing capacity of computers it is widely used to solve large problems. Monitoring system is one of the key components in distributed computing. Although there have been varieties of monitoring systems developed by different organizations, it is still a great challenge to monitor a heterogeneous distributed environment in a unified and transparent way. In this paper, we present a unified monitoring framework for distributed environment (UMFDE) with heterogeneous monitoring systems, and then propose a comprehensive method based on the Enterprise Service Bus (ESB) to integrate the monitoring systems in the environment as a unified monitoring system. A representative case study is given to show the feasibility of this framework.

Keywords: Distributed Computing, Unified Monitoring, Enterprise Service Bus

1. Introduction

Distributed computing is a field of computer science that studies distributed systems that consist of multiple autonomous computers or clusters that communicate through a network.

Distributed systems have been in the mainstream of high performance computing with the increasing computing power of computers. They have been primarily used for solving comprehensive application problems such as parallel rendering, weather modeling, nuclear simulations, and data mining.

A scalable and unified resource monitoring system is one of the key components for the effective utilization of the massive computing power in distributed environment. Due to the considerable diversity of resources in the distributed environment, the monitoring of the environment is much more difficult than that of a group of homogenous computers. There have been a variety of monitoring systems developed by different organizations, such as ganglia, Relational Grid Monitoring Architecture (R-GMA). Although some of them can be used in distributed environment, there are still some constraints. For example, we need to replace the existing monitoring systems and deploy the same monitoring system in all the clusters or computer sites. To address such challenges, [1] proposes an integrated framework (UGMF) which is compatible with other monitoring systems in the grid environment, however, there are still some problems not addressed, such as message transformation, content-based routing and flexibility.

In this paper, we present a loosely-coupled, scalable and non-intrusive monitoring framework for distributed environment with heterogeneous monitoring systems. Here we use an integrative method based on Enterprise Service Bus (ESB) to integrate the existing monitoring systems into a unified monitoring framework without redeploying or modifying the systems themselves. The proposed architecture is capable of providing clients with standard-based interfaces, and implementing uniform access to different monitoring data sources through ESB in a transparent fashion. Besides that, we implement a representative case study in a distributed environment to verify the feasibility and scalability of the proposed framework.

The rest of the paper is organized as follows. In Section 2, we introduce several monitoring systems and present an overview of the Enterprise Service Bus. In Section 3, the proposed unified monitoring architecture is discussed in detail. A representative case study based on the proposed architecture is introduced in Section 4. The final conclusion is presented in Section 5.

This research is supported by China National High Technology Program 863 (No.2008AA01A318-0)

2. Related Work

2.1. Monitoring Systems

Resource monitoring is one of the fundamental components in distributed systems. Monitoring data plays a key role in various tasks of distributed computing such as fault detection, performance analysis, performance tuning, performance prediction, and task scheduling. Resources monitored in distributed environment include the CPU load, the memory usage, the network status, the disk usage, and etc.

In this section we present a brief introduction of several existing monitoring systems for distributed systems.

2.1.1. Grid Monitoring Architecture (GMA)

The Grid Monitoring Architecture (GMA) [2] is developed by the Global Grid Forum Performance Working Group. The goal of the architecture is to provide a minimal specification that will support required functionality and allow interoperability.

The Grid Monitoring Architecture consists of three types of components, as shown in **Figure 1**:

- Directory Service: The directory service stores the information about producers and consumers, and accepts requests, supports information publication and discovery.
- Producer: A producer is any component that uses the producer interface to send monitoring data to a consumer.
- Consumer: A consumer is any component that uses the consumer interface to receive monitoring data from a producer.

The GMA architecture supports three interactions for transferring data between producers and consumers: publication/subscription, query/response, and notification. In any case, the communication of control messages and transfer of performance data occur directly between each consumer/producer pair without further involvement of the directory service.

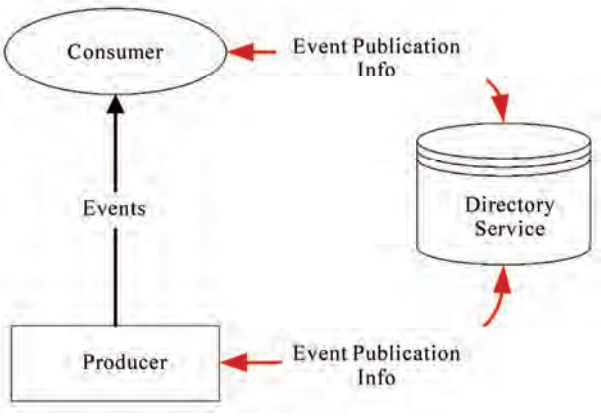


Figure 1. Components of the grid monitoring architecture.

2.1.2. Ganglia

The Ganglia [3] is a scalable distributed monitoring system for high-performance computing systems such as clusters and Grids. It is based on a hierarchical design targeted at federations of clusters, and relied on a multicast-based listen/announce protocol to monitor state within clusters and uses a hierarchy of point-to-point connections amongst representative cluster nodes to federate clusters and aggregate their state. It has been used to interconnect clusters across university campuses and around the world and can scale to handle clusters with 2000 nodes.

Compared with the Grid Monitoring Architecture, there is not a directory service for Ganglia to store and discover the information of producers and consumers.

Literature [4] presents a structure for monitoring a large set of computational clusters over wide-area networks using Ganglia. It illustrates methods for scaling a monitor network comprised of many clusters while keeping processing requirements low.

2.1.3. Windows HPC Server

Windows HPC Server 2008 [5] (HPCS) is the Microsoft high performance computing (HPC) platform built on Windows Server 2008 64-bit technology. HPCS can efficiently scale to a large number of processing cores and computers. HPCS includes a new, integrated management console that integrates the new network configuration wizard, template-based provisioning based on the Windows Server 2008 Windows Deployment Services technology, a new scheduler, cluster health monitoring at a glance along with built-in diagnostics, and a faster Microsoft Message Passing Interface (MS-MPI) that includes new NetworkDirect support.

HPC Server 2008 provides administrators and users with several tools to effectively monitor IT resources in the cluster, including the Node Management section in the Administration Console, System Center Operations Manager and Microsoft HPC Class Library.

2.2. Enterprise Service Bus

Enterprise Service Bus [6] (ESB) is a software infrastructure that can be used to connect heterogeneous applications and IT resources. It brings flow-related concepts such as transformation and routing to a Service-Oriented Architecture (SOA). It mediates incompatibilities among various applications, coordinates their interactions, and makes them broadly available as services for additional uses.

Despite various definitions used by different users, there are common components in the architecture of an ESB as below:

- Message transformation: To transform messages from one format to another based on open standards like XSLT and XPath.
- Message routing: To determine the destination of an

incoming message based on user-defined rules and logic.

- Security: Mechanisms such as authentication, authorization, and encryption should be provided by an ESB to ensure the integrity and privacy of both the ESB runtime and the messages.
- Management: A management environment is necessary for the configuration of the ESB to be reliable and also to monitor the runtime execution of the message flows in the ESB.
- Transport Management: To convert incoming transport protocols to different outgoing transport protocols.
- Message Broker: ESB acts as a broker between service consumers and service providers.

3. Overview of the Architecture

An overview of the proposed architecture (UMFDE) is shown in **Figure 2**. It is a four-tier framework that consists of the Service Provision Layer, the Service Interface Layer, the Integration Layer and the Application Layer.

Generally, there are two ways to monitor a distributed environment with a number of heterogeneous computers and clusters in the internet. One way is to deploy a new monitoring system to substitute all of the existing systems. This way is simple but impossible in most cases, because owners of the existing systems are in general not the same one. The other way is to integrate the existing systems by their exposed service interfaces without replacing the existing systems. Developers can reuse the monitoring services or develop new services.

The proposed architecture uses the integration method to achieve the goal of monitoring a distributed system.

3.1. Service Provision Layer

Service Provision Layer is comprised of a group of clusters or sites with their monitoring systems. Most of existing monitoring systems provide some kind of services that can be used by the Service Interface Layer.

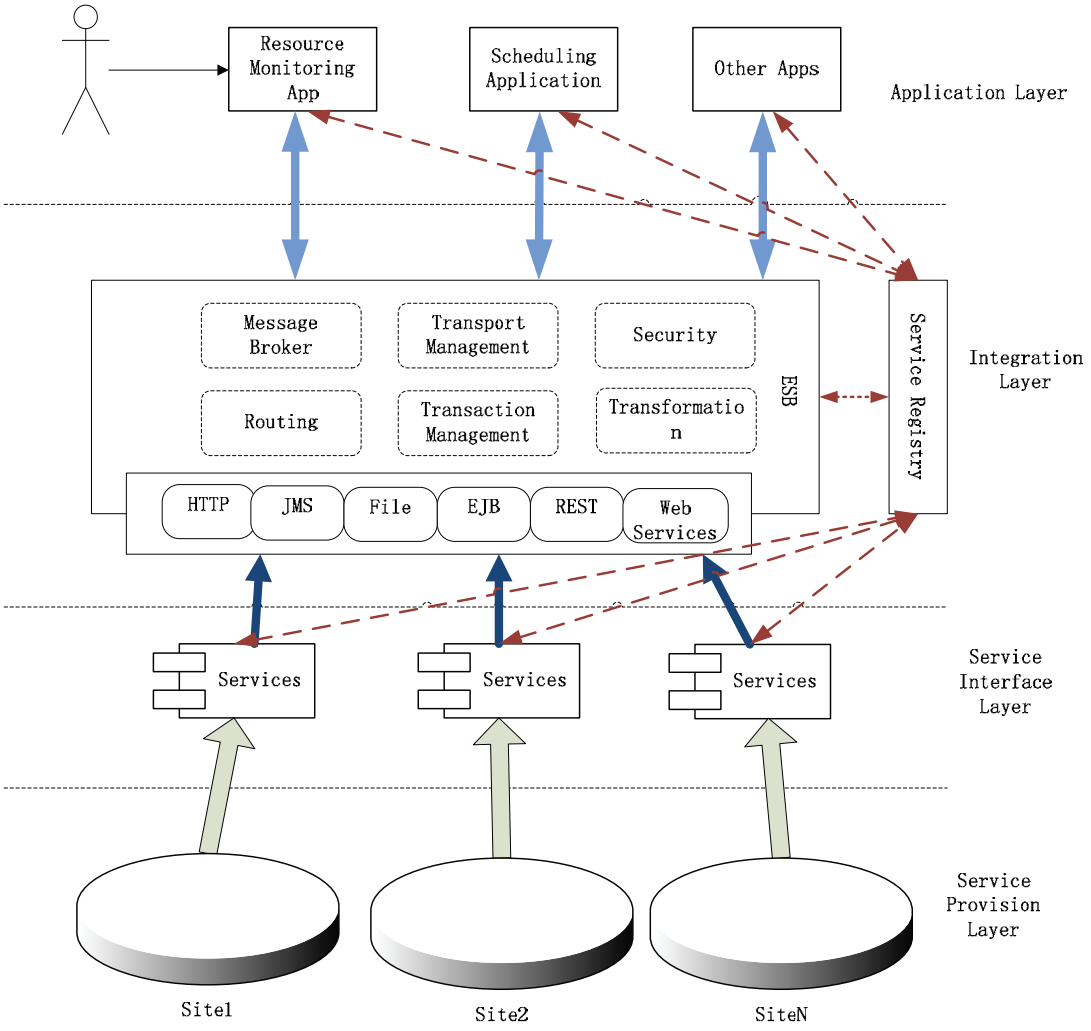


Figure 2. Architecture of the proposed monitoring framework.

3.2. Service Interface Layer

This layer is in charge of the encapsulation of the functionalities of existing monitoring systems into services, which can be connected to components and used in the Integration Layer. In this layer we may develop new monitoring services for clusters or sites without monitoring system. Since services may be offered by different systems, they may be developed based upon various protocols, such as HTTP, JMS, TCP, and SMTP.

3.3. Integration Layer

The aim of this layer is to integrate and thus provide reusability of original monitoring services defined in the Service Interface Layer, and to provide consumers with composite monitoring services. The Integration Layer is made up of two components: a Service Registry, an ESB.

Service Registry is a platform for publishing and discovering information about producers and consumers of the services, including the encapsulated services in the Service Interface Layer, the composite services in the Integration Layer, and consumer services in the Application Layer. However, it will not be further involved in the interactions between the publishers and the consumers.

ESB provides a reliable and scalable infrastructure that connects services of heterogeneous monitoring systems, mediates their incompatibilities, transforms their responses, and makes them broadly available as uniform services for different users. It is responsible for routing requests, invoking original services in Service Integration Layer, integrating and returning results and other fundamental functionalities including content-based routing, message transformation and location transparency. It also enables disparate applications and services to communicate using different protocols through transport protocol conversion.

The main features of the UMFDE are provided by the Integration Layer.

3.4. Application Layer

This layer consists of a variety of applications utilizing the monitoring data provided by the services in the Integration Layer instead of gathering raw data directly from the existing monitoring systems. These applications contain resource management applications, scheduling applications and so on.

The proposed framework supports two interaction types between producers and consumers, namely publication/subscription, and query/response.

4. Implementation

In this section, we discuss the implementation of a case study using the architecture proposed in last section. We will begin with the description of the scenario, and then continue into the details of implementation.

4.1. Overview

The context of the case study is that the client application sends a request to the UMFDE for the response of "the status of computers the client has access to in this environment".

The distributed environment used in this case study is comprised of two clusters, and a group of computers, *site A*. One cluster is Rocks cluster with monitoring system *ganglia*, the other one is Windows HPC Cluster. Site A has no monitoring system.

Here we choose the open source Mule ESB which is widely used by many enterprises to implement the case study. Key elements of Mule ESB are explained below [7-9]:

- Service component: A service component is nothing more than a Java object that is hosted and contained by Mule. These components become services within the Mule instance. Components contain the business logic. Each service is configured using an inbound router collection, a component, and an outbound router collection.
- Endpoint: An endpoint is responsible for the receiving or sending of messages via associated transport.
- Router: Based on the Message Router pattern [8], routers exist to be able to decouple individual processing steps when messages are received from, or are dispatched to, endpoints.
- Transformer: Transformers are used to convert data from one format to another, such as type transformation, message transformation, and so on.

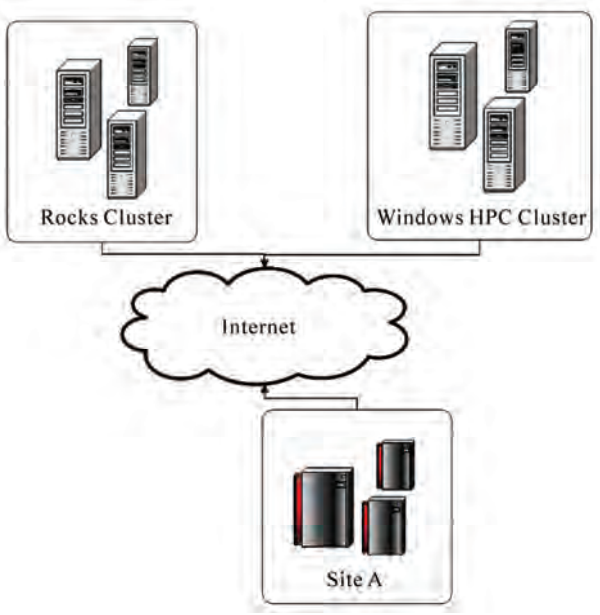


Figure 3. The distributed environment in the case study.

Besides that, we use Mule Service Registry to provide a full complement of registry and repository features, including service and artifact management, governance, lifecycle and dependency management.

The main services used in the case study include:

- *Monitor Broker Service:* Receives requests from and returns responses to client applications. It is hosted by the mule ESB.
- *Authorization Service:* As an external web service, it provides an interface for the Monitor Broker Service to obtain a list of clusters or sites whose monitoring data the current user is authorized to access.
- *Original Monitor Services*: Three services are developed to act as brokers to monitor two clusters and site A.

4.2. Implementation Details

4.2.1. Service Interface Layer

We develop web services for the monitoring systems of the two clusters. We design and develop a monitoring web service for Site A from scratch because there is no monitoring system in Site A. These services will return the status of its machines.

All of the three monitoring services will be published to the Mule Service Registry.

The response of web service of ganglia contains a list of machine's status which includes Id, Hostname, Ip, Mem_free, Cpu_num, cpu_speed, cpu_idle.

The response of web service of Windows HPC Cluster also contains a list of machine's status, but the status has different format and information. The status mainly includes Id, Name, NumberOfCores, CpuSpeed, State, Memory-Size, OnlineTime, NodeGroups, and Reachable. And the web service for Site A has similar response format as these computers are also built on windows platform.

It is obviously that the formats of responses are different, so we propose a unified response format to unify and integrate all of the response messages. We list key properties of the unified format as follows:

OsName: Name of the operating system.
Cluster name: Name of cluster the node belongs to.
ClusterType: Rocks or Windows HPC, Site A.
CpuNum: Number of cpu.
CpuSpeed: Speed of cpu.
CpuLoad: The current cpu load.
MemorySize: The total size of memory.
MemoryFreePercent: The percent of free memory.

4.2.2. Integration Layer

The message flow of the case study is divided into two parts: a request flow for handling the request message, and a response flow for dealing with the response message.

4.2.2.1. Request Flow

As shown in **Figure 4**, at first, client application calls the *MonitorBroker* Service in Mule ESB.

The *MonitorBroker* Service uses a component *msglogger* to log the request message into log files. Here we define a class to implement the log service. Then, the request message is forwarded to *AuthService*.

In the service *AuthService*, we define a Reflection-MessageBuilder component which will try and set the response of the authorization service as a bean property on the request message using reflection. The remaining part of *AuthService* is a filtering router which contains two outbound endpoints.

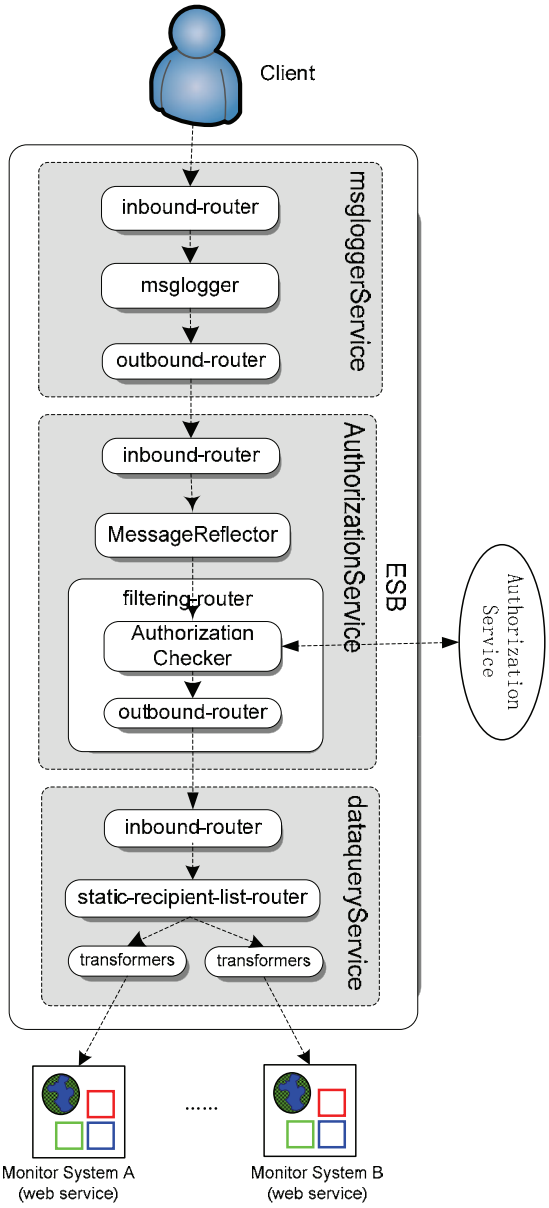


Figure 4. An overview of the Mule configuration of the request flow.

The first outbound endpoint invokes the external authorization service. After receiving the response, the *ReflectionMessageBuilder* will set the list of authorized clusters as a property called *recipients* on the request message. Then the second outbound endpoint will send the updated request message to the *DataQueryService*.

The *static-recipient-list-router* in the *DataQueryService* will apply a transformer to the request message, and then invoke external monitor services whose inbound endpoint is defined in the property *recipients* of the request message.

4.2.2.2. Response Flow

In the response flow, the monitor broker service will aggregate the response messages from different monitor services. And because the response messages are based on different formats, the broker service will transform them with the unified format.

The procedure of the response flow is described as following:

- The monitoring services process requests from the Mule ESB broker service, and return responses to the broker service.
- In the broker service, *transferService* serivce component will record the reponse activity, and then transform reponse messages using the unified format. After that, reponse messages will be forwarded to *aggregationService* component.
- *AggregationService* Service mainly contains a custom asynchronous reply router component, *result Aggregator*. The *resultAggregator* will retrieve the reponse message, and aggregate them into one reponse.
- Finnaly, the broker service will return the reponse to the client application.

4.2.2.3. Application Layer

We build a web application based on ASP.NET for users to monitor the status of the distributed environment. This client application is responsible for controlling user access and users' requests. It also provides multiple views of the status of the environment for users, such as the status of a computer and some cluster.

For this case study, we build a web form to collect users' requests, call the broker service in Mule ESB and display the results.

4.2.2.4. Sequence Graph

The sequence graph as shown in **Figure 6** is described in detail below:

1) The client sends a request to the monitor broker service hosted by the ESB to get the status of computers in the distributed environment;

2) The monitor broker service receives the request, and retrieves the authorized clusters of the client through the third party service *authorizationService*;

3) After receiving the response from *authorization-Service*, the broker service individually calls the monitoring services of the authorized sites;

4) The broker service asynchronously receives the responses from monitoring services, and then integrates them into one response message;

5) ESB sends the integrated message to the client.

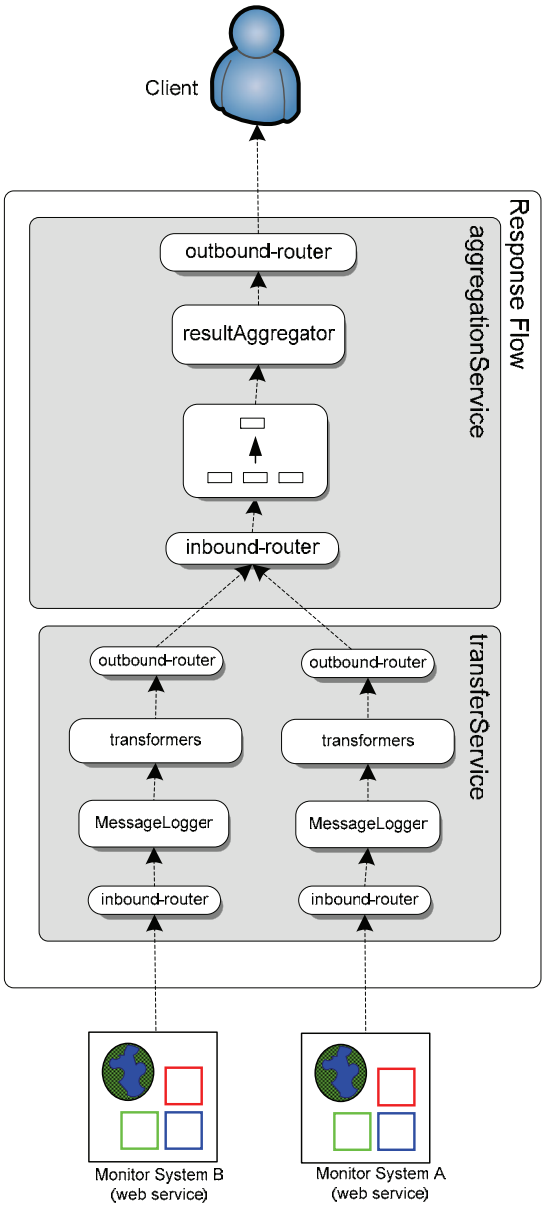


Figure 5. An overview of the Mule configuration of the response flow.

4.3. Summary

As shown in the case study, the proposed architecture is loose-coupled and scalable. It is advantageous in senses of:

- Compared with traditional unified monitoring tech-

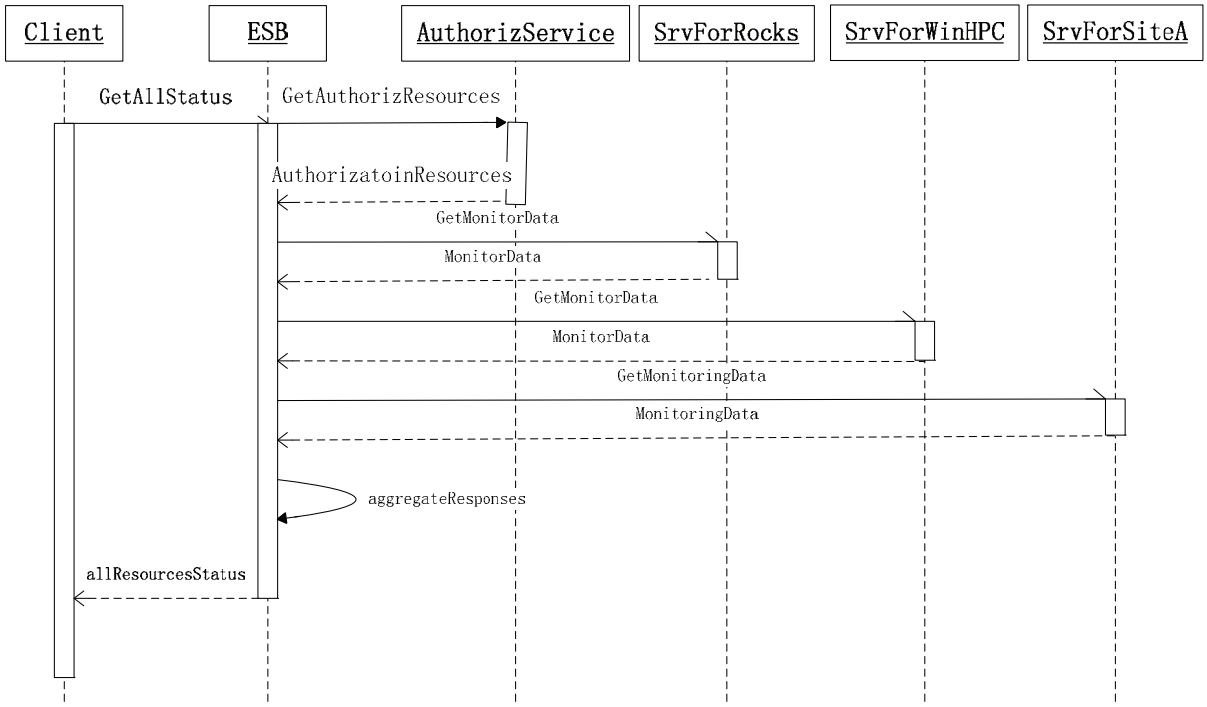


Figure 6. The sequence graph.

nologies, the integration method based on enterprise service bus makes it easier to integrate the existing monitoring systems.

- Reducing the cost of software development and maintenance.
- Rapid response to changing requirements. For example, if requirement changes, we just need modify the broker service hosted by ESB.
- Improving system flexibility, scalability, availability and robustness.

5. Conclusions

This paper proposes a unified monitoring framework for distributed environment that contains clusters or computer sites with heterogeneous monitoring systems. This framework provides a loosely-coupled, scalable and non-intrusive way to monitor a largely distributed environment.

We use an integration method based on Enterprise Service Bus to monitor computers which belong to different clusters in a unified and transparent way. For clusters with monitoring systems, we can reuse or develop services based on the legacy systems. For clusters without monitoring systems, we need develop new monitoring systems and services. By this approach, it is not required to redeploy or modify the legacy monitoring systems. The proposed framework provides standard-based interfaces to clients, and uniform access to different monitoring data sources through ESB in a transparent fashion.

The implemented case study has shown the feasibility and scalability of the proposed framework.

6. References

[1] K. Shen, S. Yang, M. Tian and P. Liu, "Towards a Uniform Monitoring Framework Supporting Interoperability in Grid," *Proceedings of the Fifth International Conference on Grid and Cooperative Computing*, Changsha, China, 2006, pp. 50-53.

[2] B. Tierney, R. Aydt, D. Gunter, W. Smith, V. Taylor, R. Wolski and M. Swany, "A Grid Monitoring Architecture," Technical Report GWD-Perf-16-1, GGF, 2002.

[3] M. Massie, B. Chun and D. Culler, "The Ganglia Distributed Monitoring System: Design, Implementation, and Experience," *Parallel Computing*, Vol. 30, No. 7, 2004, pp. 817-840.

[4] F. Sacerdoti, M. Katz, M. Massie and D. Culler, "Wide Area Cluster Monitoring with Ganglia," *IEEE International Conference on Cluster Computing*, Hong Kong, 2003, pp. 289-298.

[5] C. Russel and S. Crawford, "Windows Server 2008 Administrator's Companion," Microsoft Press, 2008.

[6] X. G. Wang, H. Wang and Y. B. Wang, "A Monitoring

Framework for Multi-Cluster Environment Using Enterprise Service Bus," *International Conference on Management and Service Science*, Beijing, China, 2009, pp. 1-4.

[7] T. Rademakers and J. Dirksen, "Open-Source ESBs in Action," Manning Publications, 2008.

[8] G. Hohpe and B. Woolf, "Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions," Addison-Wesley Longman Publishing Co. Inc., Boston, MA, USA, 2003.

[9] P. Delia and A. Borg, "Mule 2: A Developers Guide," 1st Edition, Apress, 2008.

Intelligent Information Management, **2010, 2, 406-416**
doi:10.4236/iim.2010.27050 Published Online July 2010 (http://www.SciRP.org/journal/iim)

Design and Comparison of Simulated Annealing Algorithm and GRASP to Minimize Makespan in Single Machine Scheduling with Unrelated Parallel Machines

Panneerselvam Sivasankaran[1], Thambu Sornakumar[2], Ramasamy Panneerselvam[3]
[1]*Department of Mechanical Engineering, VIT University, Vellore, India*
[2]*Department of Mechanical Engineering, Thiagarajar College of Engineering, Madurai, India*
[3]*Department of Management Studies, School of Management, Pondicherry University, Pondicherry, India*
E-mail: {*Sivasankaran.panneerselvam, sornakumar*2000}*@yahoo.com, panneer_dms@yahoo.co.in*
Received April 11, 2010; *revised May* 15, 2010; *accepted June* 20, 2010

Abstract

This paper discusses design and comparison of Simulated Annealing Algorithm and Greedy Randomized Adaptive Search Procedure (GRASP) to minimize the makespan in scheduling *n* single operation independent jobs on *m* unrelated parallel machines. This problem of minimizing the makespan in single machine scheduling problem with uniform parallel machines is NP hard. Hence, heuristic development for such problem is highly inevitable. In this paper, two different Meta-heuristics to minimize the makespan of the assumed problem are designed and they are compared in terms of their solutions. In the first phase, the simulated annealing algorithm is presented and then GRASP (Greedy Randomized Adaptive Search procedure) is presented to minimize the makespan in the single machine scheduling problem with unrelated parallel machines. It is found that the simulated annealing algorithm performs better than GRASP.

Keywords: Makespan, Simulated Annealing Algorithm, GRASP, Unrelated Parallel Machines, Mathematical Model

1. Introduction

The production scheduling is classified into single machine scheduling, flow shop scheduling, job shop scheduling, open shop scheduling and batch scheduling. The single machine scheduling is classified into single machine scheduling with single machine and with parallel machines.

The single machine scheduling problem with parallel machines is further classified into single machine scheduling with identical parallel machines and with non-identical parallel machines. The single machine scheduling problem with non-identical parallel machines is further classified into single machine scheduling problem with uniform parallel machines and with unrelated parallel machines.

Let, t_{ij} be the processing time of the job j on the machine i, for i = 1, 2, 3,…, m and j = 1, 2, 3,…, n. The three types of single machine scheduling problems with parallel machines are defined as follows.

1) If $t_{ij} = t_{1j}$ for all i and j, then the problem is called as *identical parallel machines scheduling problem*.

This means that all the parallel machines are identical in terms of their speed. Each and every job will take the same amount of processing time on each of the parallel machines.

2) If $t_{ij} = t_{1j}/s_i$ for all i and j, then the problem is termed as *uniform* (*proportional*) *parallel machines scheduling problem*. Here, s_i is the speed of the machine i and t_{1j} is the processing time of the job j on the machine 1.

This means that the parallel machines will have different speeds. Generally, we assume s_1, s_2, s_3,…, and s_m for the parallel machines 1, 2, 3,…, and m, respectively such that $s_1 < s_2 < s_3 < \ldots < s_m$. That is the machine 1 is the slowest machine and the machine m is the fastest machine. For a given job, its processing times on the parallel machines will be as per the following relation: $1/s_1 > 1/s_2 > 1/s_3 > \ldots > 1/s_m$

3) If t_{ij} is arbitrary for all i and j, then the problem is known as *unrelated parallel machines scheduling problem*.

In this type of scheduling, there will not be any relation amongst the processing times of a job on the parallel machines. This may be due to technological differences

of the machines, different features of the jobs, etc.

In this paper, the single machine scheduling problem with unrelated parallel machines is considered. The essential characteristics of the single machine scheduling problem with unrelated parallel machines are as listed below.

- It has n single operation jobs.
- It has m *unrelated* parallel machines.
- m machines are continuously available and they are never kept idle while work is waiting.
- t_{ij} is the processing time of the job j on the machine i, for $i = 1, 2, 3,..., m$ and $j = 1, 2, 3,..., n$.
- t_{ij} is arbitrary for different combinations of i and j, for $i = 1, 2, 3,..., m$ and $j = 1, 2, 3,\ldots, n$. This means that there is no relationship between the processing times of a job on different parallel machines.
- The ready time of each job is assumed to be zero ($r_j = 0, j = 1, 2, 3,\ldots, n$)
- Once processing begins on a job, it is processed to its completion without interruption. This means that the preemption of the jobs is not permitted.

There are many measures in single machine scheduling problem [1]. In this paper, the minimization of the makespan of scheduling n independent jobs on m unrelated parallel machines is considered, because it is considered to be an integrated measure of performance, which represents the earliest completion time of the given batch of jobs.

2. Literature Review

This section presents review of literature of scheduling n independent jobs on m unrelated parallel machines to minimize the makespan. As already stated, this problem comes under NP-hard category. Hence, any attempt to obtain the optimal solution through exact algorithm may end with failure for most of the instances, because of exponential computational time for such problem [2]. Hence, researchers are focusing on the development of heuristics.

Lawler and Labetoulle [3] developed a linear programming model to minimize the makespan of scheduling n jobs on m unrelated parallel machines with preemption of jobs.

Potts [4] developed a heuristic coupled with linear programming to schedule n jobs on m unrelated parallel machines for minimizing the makespan. This heuristic uses linear programming in the first phase to form a partial solution at most with $m - 1$ jobs unscheduled. In the second stage, these unscheduled jobs are scheduled using an enumeration method. Van De Velde [5] considered the single machine scheduling problem with unrelated parallel machines. The author aimed to minimize the maximum job completion time which means the minimization of makespan. He presented an optimization and an approximation algorithm that are both based on surrogate relaxation and duality to solve this NP hard problem. The idea behind surrogate relaxation is to replace a set of nasty (complex) constraints with a single constraint that is a weighted aggregate of these constraints. Glass, Potts and Shade [6] have applied meta-heuristics to the problem of scheduling jobs on unrelated parallel machines to minimize the makespan and reported that genetic algorithm gives poor results. Also, they reported that a hybrid method in which a descent is incorporated into the genetic algorithm is comparable in performance with simulated annealing. Hariri and Potts [7] developed heuristic, which consists of two phases to minimize the makespan of scheduling n independent jobs on m unrelated parallel machines. In the first phase, a linear programming model is used to schedule some of the jobs and then a heuristic is used in the second phase to schedule the remaining jobs. They have stated that some improvement procedure is necessary to have good solutions.

Piersma and Van Dijk [8] have proposed new local search algorithms to minimize the makespan of scheduling jobs in unrelated parallel machines. In these algorithms, the neighbourhood search of a solution uses the efficiency of the machines for each job. They claimed that this approach gives better results when compared to general local search algorithms. Martello, Soumis and Toth [9] have considered the scheduling of jobs on unrelated parallel machines in which the makespan is minimized. They proposed lower bounds based on Lagrangian relaxations and additive techniques. They then introduced new cuts which eliminate infeasible disjunctions on the cost function value, and prove that the bounds obtained through such cuts dominate the previous bounds. These results are used to obtain exact and approximation algorithms.

Klaus and Lorant [10] have presented polynomial-approximation schemes for preemptive and non-preemptive schemes with polynomial time complexity functions to schedule jobs on unrelated parallel machines for minimizing the makepsan. Sourd [11] has developed two approximation algorithms for minimizing the makespan of independent tasks assigned on unrelated parallel machines. The first one is based on a partial and heuristic exploration of a search tree. The second implements a new large neighbourhood improvement procedure to an already existing algorithm. He reported that the computational efficiency is equivalent to the best local search heuristics.

Serna and Xhafa [12] have developed an approach to schedule jobs on unrelated parallel machines to minimize the makespan. There approach shows how to relate the linear program obtained by relaxing the integer programming formulation of the problem with a linear program formulation that is positive and in the packing/covering form. They also demonstrated the application of

the same technique to the general assignment problem.

Mokotoff and Chretienne [13] have developed a cutting plane algorithm for the unrelated parallel machine scheduling problem in which the objective is to minimize the makespan. This algorithm deals with the polyhedral structure of the scheduling problem stated above. In their work, strong inequalities are identified for fixed values of the maximum completion time and are used to build a cutting plane scheme from which exact algorithm and an approximation algorithm are developed. Mokotoff and Jimeno [14] developed heuristics based on partial enumeration for the unrelated parallel machines scheduling problem. Given the mixed integer linear model with binary decision variables, they presented heuristics based on partial enumeration. Computational experiments are reported for a collection of test problems, showing that some of the proposed algorithms achieve better solutions than other relevant approximation algorithms published up to that *time*.

Pfund, Fowler and Gupta [15] have carried out a survey of algorithms for single and multi-objective unrelated parallel-machine deterministic scheduling problems. Under single objective case, they have considered the following.

1) Minimization of makespan

2) Minimization of the sum of the weighted completion times

3) Minimization of maximum tardiness

4) Minimization of total tardiness

5) Minimization of the sum of the weighted total earliness and weighted total tardiness

Under multi-objective case, they have discussed some select combinations of the above measures.

Ghirardi and Potts [16] have considered the problem of scheduling jobs on unrelated parallel machines to minimize the makespan. They developed recovering beam search method to minimize the makespan in the unrelated parallel machines. The traditional beam search method is a truncated version of branch and bound method. The recovering beam search allows the possibility of correcting wrong decisions by replacing partial solutions with others. It has polynomial time complexity function for the *NP* hard problem of this research. Further, they reported the computational results of this method.

Shchepin and Vakhania [17] presented a polynomial-time algorithm for non-preemptive scheduling of *n*-independent jobs on *m* unrelated parallel machines to minimize the makespan. The algorithm employs rounding approach. Monien and Woclaw [18] have presented an experimental study on the unspittable-Truemper algorithm to minimize the makespan of scheduling jobs on unrelated parallel machines. This computes 2-approximate solutions in the best worst-case running time known so far. The goal of their simulations was to prove its efficiency in practice. They compared their technique with algorithms and heuristics in practice, especially with those based on *two-step approach.*

Efraimidis and Spirakis [19] have given a new rounding approach that yields approximation schemes for multi-objective minimum makespan scheduling with a fixed number of linear cost constraints in unrelated parallel machines. The same approach can be used to maximize the minimum load on any machine and for assigning specific or equal loads to the machines. Gairing, Monien and Woclaw [20] presented a combinatorial approximation algorithm that matches an integral 2-approximation quality. It is generic minimum cost flow algorithm, without any complex enhancements, tailored to handle unsplittable flow. In their approach, they replaced the classical technique of solving LP-relaxation and rounding afterwards by a completely integral approach. Christodoulou, Koutsoupias and Vidali [21] gave an improved lower bound for the approximation ratio of truthful mechanisms for the unrelated parallel machines scheduling. The objective of the mechanism is to schedule tasks on the machines to minimize the makespan.

From these literatures, it is clear that the development of an efficient heuristic to minimize the makespan of scheduling n independent jobs on m unrelated parallel machines is a challenging task. Various authors have proposed different methodologies. Since, this problem comes under combinatorial category, development of an efficient heuristic for this problem is highly essential. Hence, in this paper, an attempt has been made to develop a simulated annealing algorithm and GRASP to minimize the makespan of scheduling jobs on unrelated parallel machines.

3. Mathematical Model to Minimize Makespan

In this section, a mathematical model is presented to minimize the makespan of the single machine scheduling problem with unrelated parallel machines.

Let, n be the number of independent jobs with single operation

m be the number of unrelated parallel machines

t_{ij} be the processing time of the job j on the machine i and it is arbitrary for different combinations of i and j.

Minimize $Z = M$

$$M - \sum_{j=1}^{n} t_{ij} x_{ij} \geq 0,\ i = 1, 2, 3, \ldots, m$$

$$\sum_{i=1}^{m} x_{ij} = 1,\ j = 1, 2, 3, \ldots, n$$

where, $x_{ij} = 1$, if the job j is assigned to the machine $i = 0$, otherwise, for $i = 1, 2, 3, \ldots, m$ and $j = 1, 2, 3, \ldots, n$

$M \geq 0$ and it is the makespan of the schedule.

The number of variables in this model is $mn + 1$ and the total number of constraints is $m + n$.

4. Simulated Annealing Algorithm to Minimize Makespan

This section presents a simulated annealing algorithm to minimize the makespan in the single machine scheduling problem with unrelated parallel machines.

In a shop floor, to give shape and size for a component, either hammering or any other equivalent operation will be carried out, which will increase the internal energy of the component. This amounts to increasing the internal stress of the component. This will misalign the internal grains of the component, which will offer resistance to work further on it. Hence, the component is to be heat treated using a process called annealing. The component will be heated to a high temperature and the temperature will be kept constant at that value for sometime. Then, it will be reduced to a next lower temperature and it will be kept constant at that value for sometime. This process will continue until the temperature is reduced to the room temperature. This heat treatment process is called annealing, which will release the internal stress of the component, there by changing the misaligned grain structure to its original structure.

The above process of annealing is mapped to solve optimization problems, especially combinatorial problems and it is termed as "simulated annealing algorithm" [22]. The parameters of the simulated annealing algorithm are given as:

T—a temperature

r—a range from 0 to 1 which is used to reduce the temperature

δ—a small positive number provided by the user (terminating criterion)

In simulated annealing algorithm, a feasible solution S_1 in the neighbourhood of S_o is generated. Such generation of a feasible schedule is obtained using perturbation. The initial seed consists of groups of jobs and each group consists of the jobs which are scheduled on one of the m unrelated parallel machines.

In simulated annealing, there are three schemes of perturbation as listed below.

1) Exchanging jobs between two machines.

2) Shifting a job from one machine to another machine.

3) Transferring a job from one of the existing machines to a new machine (In this type of scheduling, it is an infeasible scheme).

Under the perturbation schemes, the third scheme is to form a new group of jobs by transferring a job from any one of the existing machines. Under such case, a new machine will have to be included to accommodate the job which is transferred from any of the existing machines. But, the given problem of scheduling has a fixed number of machines (m). This prevents the usage of the third scheme of perturbation. So, in this paper, only the first two schemes of perturbation are used.

Caution for Perturbation: While using the second scheme of perturbation, that is shifting a job from any one of the machines to another machine, care should be taken such that the number of jobs on the machine from which a job will be shifted is at least one.

4.1. Seed Generation Heuristic

The quality of the solution of the simulated annealing algorithm depends on the effectiveness of the seed generation algorithm. The steps of the seed generation algorithm used in this paper to obtain the initial solution, S_o are presented below.

Step 1: Input the data:

Number of independent jobs, n

Number of unrelated parallel machines, m

Processing time T_{IJ}, $I = 1$ to m and $J = 1$ to n.

Step 2: Assign each of the jobs to the machine on which it takes the least processing time.

Step 3: Find the machine whose last job completion time is the maximum, *CMAX*.

Step 4: *Shift the jobs on the machine with CMAX to some other machines which give maximum reduction in makespan.*

Set *STATUS_INDEX* = 0

For each job K on the machine with CMAX, do the following:

1) a) Find the machine other than the machine with *CMAX* which requires least processing time for the current job (*K*).

b) Find the maximum of the completion times of the machines after the current job (*K*) temporarily scheduled on that machine. Let it be *MAXMCT*1.

2) a) Find the machine other than the machine with *CMAX*, on which the completion time of the last job is the minimum.

b) Find the maximum of the completion times of the machines after the current job (*K*) temporarily scheduled on that machine. Let it be *MAXMCT*2.

3) Find the minimum of *MAXMCT*1 and *MAXMCT*2 [*MIN_MAXMCT*).

4) If *MIN_MAXMCT* < CMAX, then

{Transfer the job *K* to the corresponding identified machine and update the results.

Set *STATUS_INDEX* = 1}

Step 5: If *STATUS_INDEX* = 1, then go to *Step* 4;

otherwise go to *Step* 6.

Step 6*:* Print the results: *Machine Completion Time, Assignments of the jobs on machines and Minimized makespan*.

4.2. Steps of Simulated Annealing Algorithm to Minimize Makespan

In this section, the steps of simulated annealing algorithm are presented. After an extensive experimentation, the parameters of the simulated annealing algorithm applied to the minimization of the makespan in the single machine scheduling problem with unrelated parallel machines are as follows.

$T = 60$, $r = 0.85$ and $\delta = 0.01$

Step 1: Obtain an initial schedule using the seed generation algorithm given in *Section* 4.1. Let this initial schedule be initial feasible solution S_o and the makespan of the schedule of the initial feasible solution be $f(S_o)$.

Step 2: Set the initial temperature, $T = 60$.

Step 3: Generation of feasible solution S_1 in the neighbourhood of S_o and computation of corresponding makespan, $f(S_1)$.

Step 3.1: Generate a random number, R in between 0 to 0.99.

Step 3.2: If $R \leq 0.49$ then go to *Step* 3.3; else go to *Step* 3.4.

Step 3.3: Exchanging jobs between two machines.

Step 3.3.1: Randomly select a machine (M_1), which is assigned with at least one job for transferring a job from that machine.

Step 3.3.2: Randomly select a job from the machine M_1 and let it be J_1.

Step 3.3.3: Randomly select another machine (M_2), which is assigned with at least one job for transferring a job from that machine.

Step 3.3.4: Randomly select a job from the machine M_2 and let it be J_2.

Step 3.3.5: Exchange the jobs J_1 and J_2 between the machines M_1 and M_2.

Step 3.3.6: Compute the makespan of this schedule, $f(S_1)$.

Go to *Step* 4.

Step 3.4: Transferring a job from one machine to another machine.

Step 3.4.1: Randomly select a machine (M_1), which is assigned with at least one job for transferring a job from that machine.

Step 3.4.2*:* Randomly select a job from the machine M_1 and let it be J_1.

Step 3.4.3: Randomly select another machine (M_2) to which the job J_1 is to be transferred.

Step 3.4.4*:* Transfer the job J_1 from the machine M_1 to the machine M_2.

Step 3.4.5: Compute the makespan of this schedule, $f(S_1)$.

Go to *Step* 4.

Step 4: Compute $d = f(S_0) - f(S_1)$.

Step 5: Updating S_o.

Step 5.1: If $d > 0$, set $S_o = S_1$ and go to *Step* 6; else go to *Step* 5.2.

Step 5.2: Generate uniformly distributed random number (R) in the range 0 to 1.

Step 5.3: If $R < e^{(d/T)}$, then set $S_o = S_1$ and go to Step 6; else go to *Step* 6.

Step 6: Set $T = r \times T$

Step 7: If $T > \delta$, then go to *Step 3*; otherwise go to *Step* 8.

Step 8: Use the seed generation algorithm (local optimum procedure) to reach a local optimum starting from the last S_0 value and print the final schedule along with the corresponding makespan.

Step 9: Stop.

5. Greedy Randomized Adaptive Search Procedure (GRASP) to Minimize Makespan

In this section, another heuristic based on Greedy Randomized Adaptive Search Procedure (GRASP) for the single machine scheduling problem with unrelated parallel machines is presented. The GRASP is an iterative procedure, which has two phases, namely construction phase and local search phase. In the construction phase, a feasible solution is constructed by choosing the next element randomly from the Candidate List (CL). The candidate list contains only the best elements selected by greedy function.

The Candidate List technique makes it possible to obtain different solution in each of the iterations. Since the solutions generated by the GRASP construction phase are not guaranteed to be the local optimum, it is recommended to apply the local search phase, which is the second phase of the GRASP. In this paper, a greedy heuristic is applied for the local search phase. At the end of each GRASP-iteration, the better solution is updated. The latest best solution becomes the final solution, when the given termination criterion is reached [23].

5.1. Steps of GRASP to Minimize Makespan

The steps of GRASP to minimize the makespan of the single machine scheduling problem with unrelated parallel machines are presented below.

Step 1*:* Input the data:

Number of independent jobs, n

Number of unrelated parallel machines, m

Processing time $T(I, J)$, $I = 1$ to m and $J = 1$ to n.

Greedy parameter used in *Step 9*, γ (In this case, it is assumed as 10)

Step 2: Set the Candidate List, LC = Null Set.

Step 3: Use the following ***Greedy heuristic*** to find the feasible solutions and add it to the *Candidate List* (*CL*).

Step 3.1: Assign each of the jobs to the machine on which it takes the least processing time.

Step 3.2: Find the machine whose last job completion

time is the maximum, *CMAX*. Update the Best Makespan, *BMS* = *CMAX*

Step 4: Select the lone member of the *Candidate List* (*CL*),

Let it be *M*.

Step 5: Generation of *Current Pool of Solution* (*CPS*).

Step 5.1: $I = 1$

Step 5.2: Set the termination criterion index, $k = 0$.

Step 5.3: Randomly select any one of the following:

1) Exchange of jobs between machines

2) Transfer of jobs from one machine to another machine.

If the selected option is "Exchange of jobs between machines", then go to *Step* 5.4; otherwise, go to *Step* 5.5.

Step 5.4: Exchanging jobs between two machines.

Step 5.4.1: Randomly select a machine (M_1), which is assigned with at least one job for transferring a job from that machine.

Step 5.4.2: Randomly select a job from the machine M_1 and let it be J_1.

Step 5.4.3: Randomly select another machine (M_2), which is assigned with at least one job for transferring a job from that machine.

Step 5.4.4: Randomly select a job from the machine M_2 and let it be J_2.

Step 5.4.5: Whether the exchange is feasible? If yes, go to *Step* 5.4.6; otherwise, go to *Step* 5.6.

Step 5.4.6: Exchange the jobs J_1 and J_2 between the machines M_1 and M_2.

Step 5.4.7: Compute the makespan of this schedule, $f(S_1)$ and add the solution to the C*urrent Pool of Solution* (*CPS*).

Update $k = 1$ *and* go to *Step* 5.6.

Step 5.5: Transferring a job from one machine to another machine.

Step 5.5.1: Randomly select a machine (M_1), which is assigned with at least one job for transferring a job from that machine.

Step 5.5.2: Randomly select a job from the machine M_1 and let it be J_1.

Step 5.5.3: Randomly select another machine (M_2) to which the job J_1 is to be transferred.

Step 5.5.4: Whether the transfer is feasible? If yes, go to *Step* 5.5.5; otherwise, go to *Step* 5.6.

Step 5.5.5*:* Transfer the job J_1 from the machine M_1 to the machine M_2.

Step 5.5.6: Compute the makespan of this schedule, $f(S_1)$ and add the solution to *Current Pool of Solution* (*CPS).*

Set $k = 1$ *and* go to *Step* 5.6.

Step 5.6: $I = I + 1$

Step 5.7: If $I \leq \gamma$ then go to *Step* 5.3, or else go to *Step* 6.

Step 6: If $k = 0$ then go to *Step* 10, or else go to *Step* 7.

Step 7: Find best solution from the *Current Pool of Solution* (*CPS*) and add it to the *Candidate List* (*CL*).

Step 8: Find the best solution from the *Candidate List* (*CL*) and let it be *M*.

Step 9: Go to *Step* 5.

Step 10: Print the lastly used best solution (*M*) in *Step* 5.

Step 11*:* Stop.

6. Comparison of Solutions of Simulated Annealing Algorithm and GRASP

In this section, the solutions obtained through the simulated annealing algorithm are compared with the solutions obtained through GRASP using a complete factorial experiment. In the factorial experiment, two factors are assumed, viz., "*Method* (*M*)" and "*Problem Size* (*P*)". The number of levels for "Method" is 2, viz., "Simulated Annealing Algorithm" and "*GRASP"*. The number of levels for "Problem Size" is 90 which are 3 × 11, 3 × 12, 3 × 13,..., 3 × 25, 4 × 11, 4 × 12, 4 × 13,..., 4 × 25, 5 × 11, 5 × 12, 5 × 13,..., 5 × 25,…, 8 × 11, 8 × 12, 8 × 13,…, *and* 8 × 25. For each of 180 experimental combinations, the data for three replications have been randomly generated. The values of the makespan of these problems using the simulated annealing algorithm (SA algorithm) and GRASP are presented in **Table 1**.

The respective ANOVA model [24] is presented below.

$y_{ijk} = \mu + M_i + P_j + MP_{ij} + e_{ijk}$

where, μ is the overall mean of the makespan

y_{ijk} is the response in terms of the makespan for the k^{th} replication under the i^{th} level of the factor *M* and the j^{th} level of the factor *P*.

M_i is the effect of the i^{th} level of the factor *M* on the response y_{ijk}

P_j is the effect of the j^{th} level of the factor *P* on the response y_{ijk}

MP_{ij} is the effect of the i^{th} level of the factor *M* and the j^{th} level of the factor *P* on the response y_{ijk}

e_{ijk} is the error in the k^{th} replication under the i^{th} level of the factor *M* and the j^{th} level of the factor *P*.

The results of the corresponding ANOVA model are shown in **Table 2**.

The hypotheses of this ANOVA model are as listed below.

6.1. Factor "Method (*M*)"

H_0: There is no significant difference between the methods (*Simulated Annealing Algorithm* and *GRASP*) in terms of makespan.

H_1: There is significant difference between the methods (*Simulated Annealing Algorithm* and *GRASP*) in terms of makespan.

In the **Table 2**, the calculated F ratio of the factor

Table 1. Makespan results of SA algorithm and GRASP.

Problem Size	Replication	Method		Problem Size	Replication	Method	
		SA	GRASP			SA	GRASP
	1	27	31		1	20	44
3 × 11	2	26	42	4 × 11	2	13	22
	3	26	47		3	20	27
	1	33	35		1	24	29
3 × 12	2	27	43	4 × 12	2	22	34
	3	28	31		3	18	22
	1	31	54		1	20	23
3 × 13	2	44	52	4 × 13	2	14	14
	3	20	20		3	22	30
	1	37	37		1	18	23
3 × 14	2	30	38	4 × 14	2	23	26
	3	24	28		3	19	27
	1	27	41		1	25	34
3 × 15	2	43	61	4 × 15	2	23	31
	3	42	47		3	22	23
	1	44	44		1	29	36
3 × 16	2	43	57	4 × 16	2	28	44
	3	43	57		3	19	29
	1	35	42		1	32	42
3 × 17	2	48	68	4 × 17	2	25	29
	3	38	38		3	22	31
	1	32	35		1	34	50
3 × 18	2	45	50	4 × 18	2	21	40
	3	34	47		3	30	44
	1	33	61		1	36	51
3 × 19	2	49	53	4 × 19	2	26	30
	3	60	87		3	33	51
	1	51	73		1	31	43
3 × 20	2	40	45	4 × 20	2	34	41
	3	41	46		3	29	33
	1	53	88		1	32	52
3 × 21	2	57	76	4 × 21	2	28	44
	3	43	64		3	22	28
	1	49	63		1	24	34
3 × 22	2	47	65	4 × 22	2	36	50
	3	50	60		3	37	50
	1	57	55		1	34	53
3 × 23	2	45	71	4 × 23	2	37	63
	3	57	76		3	36	55
	1	46	56		1	37	52
3 × 24	2	51	81	4 × 24	2	39	45
	3	62	68		3	31	48
	1	64	84		1	44	53
3 × 25	2	58	83	4 × 25	2	38	64
	3	60	110		3	37	41

Problem Size	Replication	Method	
		SA	GRASP
5 × 11	1	16	16
	2	16	17
	3	12	20
5 × 12	1	18	22
	2	19	26
	3	15	15
5 × 13	1	12	13
	2	13	16
	3	7	10
5 × 14	1	19	22
	2	18	20
	3	16	24
5 × 15	1	15	21
	2	22	27
	3	21	26
5 × 16	1	15	17
	2	19	26
	3	18	23
5 × 17	1	24	42
	2	19	32
	3	16	18
5 × 18	1	21	21
	2	18	25
	3	23	28
5 × 19	1	27	27
	2	18	28
	3	21	31
5 × 20	1	27	30
	2	22	24
	3	21	28
5 × 21	1	25	31
	2	16	26
	3	28	41
5 × 22	1	22	32
	2	24	28
	3	22	24
5 × 23	1	31	35
	2	21	29
	3	23	33
5 × 24	1	32	50
	2	38	38
	3	27	42
5 × 25	1	31	35
	2	28	43
	3	27	31

Problem Size	Replication	Method	
		SA	GRASP
6 × 11	1	21	28
	2	13	20
	3	18	19
6 × 12	1	11	17
	2	18	30
	3	15	26
6 × 13	1	20	33
	2	14	17
	3	19	25
6 × 14	1	15	29
	2	11	15
	3	12	18
6 × 15	1	12	14
	2	16	24
	3	21	29
6 × 16	1	16	25
	2	21	28
	3	15	20
6 × 17	1	13	13
	2	16	24
	3	14	27
6 × 18	1	20	26
	2	19	28
	3	14	21
6 × 19	1	23	23
	2	22	33
	3	17	23
6 × 20	1	21	32
	2	19	27
	3	18	28
6 × 21	1	17	20
	2	19	22
	3	20	23
6 × 22	1	13	18
	2	20	25
	3	18	22
6 × 23	1	18	35
	2	30	40
	3	24	36
6 × 24	1	21	34
	2	19	23
	3	26	32
6 × 25	1	25	32
	2	20	29
	3	20	34

Problem Size	Replication	Method	
		SA	GRASP
7 × 11	1	7	9
	2	6	7
	3	11	25
7 × 12	1	12	16
	2	9	11
	3	9	12
7 × 13	1	8	13
	2	7	9
	3	9	13
7 × 14	1	12	14
	2	14	22
	3	14	29
7 × 15	1	14	14
	2	13	21
	3	13	26
7 × 16	1	10	12
	2	14	20
	3	14	14
7 × 17	1	13	22
	2	10	11
	3	16	19
7 × 18	1	13	18
	2	9	16
	3	15	21
7 × 19	1	11	13
	2	19	26
	3	10	14
7 × 20	1	12	14
	2	16	21
	3	11	21
7 × 21	1	15	18
	2	14	34
	3	12	19
7 × 22	1	17	25
	2	12	13
	3	12	15
7 × 23	1	12	16
	2	23	29
	3	15	20
7 × 24	1	15	25
	2	19	25
	3	19	23
7 × 25	1	14	14
	2	18	23
	3	18	34

Problem Size	Replication	Method	
		SA	GRASP
8 × 11	1	8	8
	2	11	20
	3	8	8
8 × 12	1	5	7
	2	5	8
	3	7	9
8 × 13	1	12	17
	2	9	13
	3	10	10
8 × 14	1	11	13
	2	7	8
	3	15	17
8 × 15	1	10	12
	2	11	13
	3	12	12
8 × 16	1	16	8
	2	18	24
	3	16	16
8 × 17	1	15	15
	2	10	11
	3	10	10
8 × 18	1	7	7
	2	13	14
	3	10	17
8 × 19	1	11	13
	2	12	16
	3	11	15
8 × 20	1	9	13
	2	12	14
	3	12	16
8 × 21	1	15	26
	2	11	16
	3	14	21
8 × 22	1	14	20
	2	20	20
	3	13	19
8 × 23	1	10	19
	2	11	17
	3	13	16
8 × 24	1	16	32
	2	14	16
	3	14	16
8 × 25	1	15	22
	2	15	28
	3	18	23

Table 2. ANOVA results for comparison of simulated annealing algorithm and GRASP.

Source of variation	Degrees of freedom	Sum of squares	Mean Sum of Squares	F_{Ratio}
Method(*M*)	1	8252.469	8252.469	225.471
Problem Size(*P*)	89	103964.300	1168.138	31.926
Method × Problem Size(*M* × *P*)	89	3660.500	41.129	1.124
Error	360	13176.380	36.601	
Total	539	129053.640		

"Method (*M*)" is 225.471, which is more than the corresponding table F value (3.84) for (1,360) degrees of freedom at a significance level of 0.05. Hence, the null hypothesis is to be rejected. This means that t*here is significant difference between the methods* (*Simulated Annealing Algorithm and GRASP*) in terms of makespan.

The mean of the makespan values of all 270 problems is 22.30741 using the simulated annealing algorithm and that is 30.12593 using GRASP. By combining the above two facts, it is clear that the Simulated Annealing Algorithm performs better than the Greedy Randomized Adaptive Search Procedure (GRASP).

6.2. Factor "Problem Size" (*P*)

H_0: There is no significant difference between the problems in terms of makespan [*Problem size* (*P*)].

H_1: There is significant difference between the problems in terms of makespan [*Problem size* (*P*)].

In the **Table 2**, the calculated F ratio of the factor, "*Problem Size* (*P*)" is 31.916, which is more than the table F value (1.27) for (89,360) degrees of freedom at a significance level of 0.05. Hence, the corresponding null hypothesis is to be rejected. This means that there is significant difference between the problems in terms of makespan.

6.3. Interaction "Method × Problem Size" (*M* × *P*)

H_0: There is no significant difference between the interaction terms in terms of makespan.

H_1: There is significant difference between at least one pair of the interaction terms in terms of makespan.

In the **Table 2**, the calculated F ratio of the interaction "Method × Problem Size" is 1.124, which is less than the table F value (1.27) for (89,360) degrees of freedom at a significance level of 0.05. Hence, the corresponding null hypothesis is to be accepted. This means that there is no significant difference between the interaction terms in terms of makespan.

7. Conclusions

Production scheduling paves ways for effective calendar of production for day to day requirements in industries. The single machine scheduling problem with unrelated parallel machines which is considered in this paper is a challenging problem because it is a combinatorial problem. The design of two different meta-heuristics, viz., simulated annealing algorithm and greedy randomized adaptive search procedure (GRASP) are presented. Then, their performances are compared through a complete factorial experiment with 270 randomly generated problems with different sizes (small to big sizes: 3 × 11, 3 × 12,…, 3 × 25, 4 × 11, 4 × 12,…, 4 × 25,..., 8 × 11, 8 × 12,..., 8 × 25, each size with three replications). Based on the ANOVA results, it is found that there is significant difference between the simulated annealing algorithm and GRASP, in terms of their performance. The mean of the makespan values of all 270 problems is 22.30741 using the simulated annealing algorithm and that is 30.12593 using GRASP. By combining the above two facts, it is clear that the simulated annealing performs better than the Greedy Randomized Adaptive Search Procedure (GRASP).

8. References

[1] R. Panneerselvam, "Production and Operations Management," 2nd Edition, PHI Learning Private Limited, New Delhi, 2005.

[2] R. Panneerselvam, "Design and Analysis of Algorithms," PHI Learning Private Limited, New Delhi, 2007.

[3] E. L. Lawler and J. Labetoulle, "On Preemptive Scheduling on Unrelated Parallel Processors by Linear Programming," *Journal of the ACM*, Vol. 25, No. 4, 1978, pp. 612-619.

[4] C. N. Potts, "Analysis of a Linear Programming Heuristic for Scheduling Unrelated Parallel Machines," *Discrete Applied Mathematics*, Vol. 10, No. 2, 1985, pp. 155-164.

[5] S. L. Van De Velde, "Duality-Based Algorithms for Scheduling Unrelated Parallel Machines," *ORSA Journal of Computing*, Vol. 5, No. 2, 1993, pp. 182-205.

[6] C. A. Glass, C. N. Potts and P. Shade, "Unrelated Parallel Machine Scheduling Using Local Search," *Mathematical and Computer Modelling*, Vol. 20, No. 2, 1994, pp. 41-52.

[7] A. M. A. Hariri and C. N. Potts, "Heuristics for Scheduling Unrelated Parallel Machines," *Computers and Operations Research*, Vol. 18, No. 3, 1991, pp. 323-331.

[8] N. Piersman and W. Van Dijk, "A Local Search Heuristic for Unrelated Parallel Machine Scheduling with Efficient Neighbourhood Search," *Mathematical and Computer Modelling*, Vol. 24, No. 9, 1996, pp. 11-19.

[9] S. Martello, F. Soumis and P. Toth, "Exact and Approximation Algorithms for Makepsan Minimization on

Unrelated Parallel Machines," *Discrete Applied Mathematics*, Vol. 75, No. 2, 1997, pp. 169-188.

[10] J. Klaus and P. Lorant, "Improved Approximation Schemes for Scheduling Unrelated Parallel Machines," *Mathematics of Operations Research*, Vol. 26, No. 2, 2001, pp. 324-338.

[11] F. Sourd, "Scheduling Tasks on Unrelated Machines: Large Neighbourhood Improvement Procedures," *Journal of Heuristics*, Vol. 7, No. 6, 2001, pp. 519-531.

[12] M. Serna and F. Xhafa, "Approximating Scheduling Unrelated Parallel Machines in Parallel," *Computational Optimization and Applications*, Vol. 21, No. 3, 2002, pp. 325-338.

[13] E. Mokotoff and P. Chretienne, "A Cutting Plane Algorithm for the Unrelated Parallel Machine Scheduling Problem," *European Journal of Operational Research*, Vol. 141, No. 3, 2002, pp. 515-525.

[14] E. Mokotoff and J. L. Jimeno, "Heuristics Based on Partial Enumeration for the Unrelated Parallel Processor Scheduling Problem," *Annals of Operations Research*, Vol. 117, No. 1-4, 2002, pp. 133-150.

[15] M. Pfund, J. W. Fowlwr and J. N. D. Gupta, "A Survey of Algorithms for Single Machine and Multi-Objective Unrelated Parallel-Machine Deterministic Scheduling Problems," *Journal of the Chinese Institute of Industrial Engineers*, Vol. 21, No. 3, 2004, pp. 230-241.

[16] M. Ghirardi and C. N. Potts, "Makespan Minimization for Scheduling Unrelated Parallel Machines: A Recovering Beam Search Approach," *European Journal of Operational Research*, Vol. 165, No. 2, 2005, pp. 457-467.

[17] E. V. Shchepin and N. Vakhania, "An Optimal Rounding Gives a Better Approximation for Scheduling Unrelated Machines," *Operations Research Letters*, Vol. 33, No. 2, 2005, pp. 127-133.

[18] B. Monien and A. Woclaw, "Scheduling Unrelated Parallel Machines Computational Results," *Proceedings of the* 5*th International Workshop on Experimental Algorithms*, Menorca, Spain, 2006, pp. 195-206.

[19] P. S. Efraimidis and P. G. Spirakis, "Approximation Schemes for Scheduling and Covering on Unrelated Machines," *Theoretical Computer Science*, Vol. 359, No. 1, 2006, pp. 400-417.

[20] M. Gairing, B. Monien and A. Woclaw, "A Faster Combinatorial Approximation Algorithm for Scheduling Unrelated Machines," *Theoretical Computer Science*, Vol. 380, No. 1-2, 2007, pp. 87-99.

[21] G. Christodoulou, E. Koutsoupias and A. Vidali, "A Lower Bound for Scheduling Mechanisms," *Algorithmica*, Vol. 55, No. 4, 2009, pp. 729-740.

[22] S. Kirkpatrick, Jr. C. D. Gelatt and M. P. Vecchi, "Optimization by Simulated Annealing," *Science*, Vol. 220, No. 4598, 1983, pp. 671-680.

[23] T. A. Feo and M. G. C. Resende, "Greedy Randomized Adaptive Search Procedures," *Journal of Global Optimization*, Vol. 6, No. 2, 1995, pp. 109-133.

[24] R. Panneerselvam, "Research Methodology," PHI Learning Private Limited, New Delhi, 2004.

Intelligent Information Management, **2010, 2, 417-421**
doi:10.4236/iim.2010.27051 Published Online July 2010 (http://www.SciRP.org/journal/iim)

Particle Filter Data Fusion Enhancements for MEMS-IMU/GPS

Yafei Ren, Xizhen Ke
Faculty of Automation and Information Engineering, Xi'an University of Technology, Xi'an, China
E-mail: *doctor_liu@126.com*
Received April 14, 2010; *revised May* 18, 2010; *accepted June* 22, 2010

Abstract

This research aims at enhancing the accuracy of navigation systems by integrating GPS and Micro-Electro-Mechanical-System (MEMS) based inertial measurement units (IMU). Because of the conditions required by the large number of restrictions on empirical data, a conventional Extended Kalman Filtering (EKF) is limited to apply in navigation systems by integrating MEMS-IMU/GPS. In response to non-linear non-Gaussian dynamic models of the inertial sensors, the methods rely on a particle cloud representation of the filtering distribution which evolves through time using importance sampling and resampling ideas. Then Particle Filtering (PF) can be used to data fusion of the inertial information and real-time updates from the GPS location and speed of information accurately. The experiments show that PF as opposed to EKF is more effective in raising MEMS-IMU/GPS navigation system's data integration accuracy.

Keywords: Micro-Electro-Mechanical-System, Particle Filter, Data Fusion, Extended Kalman Filtering

1. Introduction

Inertial sensors are widely used for navigation systems [1]. Compared to GPS tracking result, inertial tracking offers attractive complementary features. Given perfect sensors, no external information other than initial pose estimation is required. Lang *et al*. [1,2] show that inertial sensors can provide a good signal-to-noise ratio, especially in cases of rapid directional change (acceleration/deceleration) and for high rotational speed. However, since inertial sensors only measure the variation rate or accelerations, the output signals have to be integrated to obtain the position and orientation data. As a result, longer integrated time produces significant accumulated drift because of noise or bias.

In MEMS-IMU/GPS integration, there are nonlinear models that should be properly handled, for example: 1) nonlinear state equations describing the MEMS-IMU error propagation; 2) nonlinear measurement equations that are related to pseudo ranges, carrier phases and Doppler shifts measured in the GPS receiver [3]. In recent years, to overcome the problems with the nonlinearity, other nonlinear filters are also considered for use in the MEMS-IMU/GPS integration, for example: 1) Particle Filter(PF), 2) Unscented Kalman Filter(UKF), 3) SIR Particle Filter(SPF) [4,5]. It is reported [5,6] that the integrated systems with these nonlinear filters show the similar performances, producing almost the same accuracies in horizontal position and velocity while the accuracy of the heading angle can be improved.

This paper combined GPS and inertial sensor with a two-channel complementary PF, which can take advantage of the low-frequency stability of GPS sensors and the high-frequency tracking of gyro sensors.

2. System Overview

Hybrid solutions attempt to overcome the drawbacks of any single sensing solution by combining the measurements of at least two tracking methods. The fusion of complementary sensors should be used to build better tracking systems. Synergies can be exploited to gain robustness, tracking speed and accuracy, and to reduce jitter and noise. Nowadays, a hybrid tracking system is the best solution to achieve a better object pose estimation and is widely applied in recent research works.

A Particle Filter (PF) is used to estimate motion by fusion of inertial and GPS data. The Extended Kalman Filter (EKF) is used for data fusion and error compensation.

Most of the hybrid approaches use the EKF to estimate the object state by fusion of inertial and GPS data. However, the EKF algorithm provides only an approximation

to optimal nonlinear estimation. This can introduce large errors in the true posterior mean and covariance of the transformed Gaussian random variables, which may lead to suboptimal performance and sometimes divergence of the filter.

Our GPS system consists of a robust pose and an Inertial Measurement Unit (IMU) providing 3-D linear acceleration and 3-D rate of turn (rate gyro) [7]. Moreover, the Particle Filtering algorithm is suggested and assessed to model the variations of the MEMS sensors' performance characteristics. Initial results show the efficiency and precision of the proposed PF modeling algorithm.

3. Three-Dimensional Error Estimation

3.1. Motion Model and System Dynamics

Any navigation systems tracking approach requires some kind of motion model, even if it is constant motion. Since we have no a priori knowledge about the forces changing the motion of GPS system or the objects, we assume no forces (accelerations) and hence constant velocities. Augmented reality (AR) systems [1] have the goal of enhancing a person's perception of the surrounding world. We used the motion model proposed by Ref [7], where the objects motion is represented by a 15 × 1 vector:

$$X_{gps} = (\theta, \omega, x, \dot{x}, \ddot{x}) \tag{1}$$

where θ is the orientation of GPS with respect to the world (we use $Z-Y-X$ Euler angles), ω is the angular velocity, $x, \dot{x}, \ddot{x}$ are the position, velocity and acceleration of GPS with respect to the world. With these states, the discretized system dynamics are given by:

$$\begin{bmatrix} \theta_{k+1} \\ \omega_{k+1} \\ x_{k+1} \\ \dot{x}_{k+1} \\ \ddot{x}_{k+1} \end{bmatrix} = \begin{bmatrix} \theta_k + \Delta T \cdot W(\theta) \cdot \omega_k + w_k^1 \\ \omega_k + w_k^2 \\ x_k + \Delta T \cdot \dot{x} + \frac{1}{2}\Delta T^2 \cdot \ddot{x}_k + w_k^3 \\ \dot{x} + \Delta T \cdot \ddot{x}_k + w_k^4 \\ \ddot{x}_k + w_k^5 \end{bmatrix} \tag{2}$$

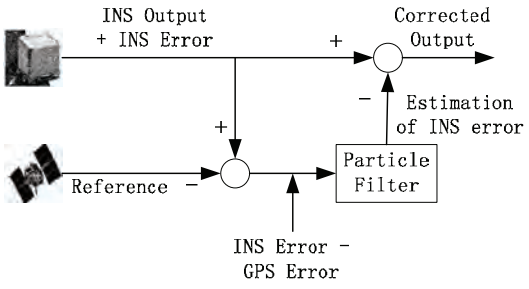


Figure 1. System dataflow.

where ΔT is the sampling period, $W(\theta)$ is the Jacobian matrix that relates the absolute rotation angle to the angular rate and w_k^i is the system random distribution noise. At each time, the 3-D GPS pose is given by the values of the x and θ parameters.

3.2. Inertial Sensor Measurements

Our inertial sensor consists of three orthogonal rate gyroscopes to sense angular rates of rotation along three perpendicular axes and three accelerometers which produce three acceleration measurements. GPS orientation changes are reported by the inertial sensor, so the transformation between the $\{G\}$ and $\{I\}$ is needed to relate inertial and GPS motion. The rotation motion relationship between the two coordinates can be derived by:

$$\omega_G = R_I^G \cdot \omega_I \tag{3}$$

where ω_G and ω_I denote the angular velocity relative to the GPS coordinate frame and the inertial coordinate frame, respectively. In order to determine the transformation matrix R_I^G we have developed an efficient calibration method which is described in more detail in Ref [8]. The accelerometers produce three acceleration measurements, one for each axis (units: mm/s^2). The accelerometers provide linear acceleration measurements in their own coordinate frame $\{I\}$. However, the acceleration term in our state vector is the linear acceleration from $\{C\}$ to $\{W\}$. The function relating the state vector X_{GPS} and the linear acceleration measurements is given by:

$$a_c = R_W^I \times (\ddot{x} + \omega \times \frac{d}{dt}(R_C^W(\theta) \cdot P_I^C) + g) \tag{4}$$

where the rotation from $\{W\}$ to $\{I\}$ is R_I^C, P_I^C is the position of the origin of the $\{I\}$ frame with respect to the $\{C\}$ frame and g is gravity. R_I^C and P_I^C are determined by the calibration procedure.

3.3. Fusion Filter

The goal of the fusion filtering is to estimate object pose parameters of (1) from the measurements of the vision and inertial sensors [9]. The basis of our fusion algorithm is a SIR particle filter [8,10]. In this section we will explain how to use such a filter to estimate the camera pose. For more details on particle filter theory, see Refs [11-13]. The motion tracking can be considered as a single-target non-linear discrete time system whose process model can be expressed as follows:

$$X_{k+1} = f(X_k, w_k) \tag{5}$$

where X_k denotes the state vector of the system at time *k*, it is defined by (1). $f(X_k, w_k)$ represents the deterministic process which is given by (2). The measurement model is given by:

$$y_k = h(X_k) + n_k \tag{6}$$

where n_k represent the measurements noise. The nonlinear function h relates the state vector X to the measurements y. In our design, the input measurements to the fusion filter come from three different sensors, *i.e.*, GPS, gyroscope and accelerometers, each with its own output equation h and an uncertainty in the output space:

3.3.1. For the Gyroscope

The gyroscope produces three angular velocity measurements, one for each axis (units: rad/s). This information will be associated only with the angular velocity term in the state vector. The gyroscope measurement model is then given by:

$$y_k^{gyro} = h_{gyro}(X_k) + n_k^{gyro} = H_{gyro} \times X_k + n_k^{gyro} \tag{7}$$

relating the state vector with the measurement vector using an identity matrix, so:

$$H_{gyro} = [0_{3\times3} \quad I_{3\times3} \quad 0_{3\times3} \quad 0_{3\times3} \quad 0_{3\times3}] \tag{8}$$

where $0_{3\times3}$ represents a 3 × 3 zero matrix and $I_{3\times3}$ a 3 × 3 identity matrix.

3.3.2. For the Accelerometers

The accelerometers produce three acceleration measurements, one for each axis (units: mm/s^2). The accelerometer measurement model is defined by:

$$y_k^{acc} = h_{acc}(X_k) + n_k^{acc} \tag{9}$$

where h_{acc} is given by (4) and n_k^{acc} is the accelerometer's measurements noise.

3.3.3. Particle Filtering

After having formulated the process and measurement models for both of the inertial and GPS sensors, we can now give an iteration of the SIR particle filter:

1) Hypotheses.

Let: N be the number of particles.

$p(X_0) \equiv p(X_0 / y_0)$ be the prior distribution (at $k = 0$).

2) Initialization.

For $k = 0$: initialize the N particles and generate $\{X_0^{(i)}\}_{i=1}^N$ from the initial distribution $p(X_0)$ initialize the weights $\Omega_0^{(i)} = \frac{1}{N}$.

3) Evolution.

Predict new particles $X_k^{(i)}$ using different noise realization and the process model:

$$X_k^{(i)} = f(X_{k-1}^{(i)}, w_k), \qquad i = 1,..., N. \tag{10}$$

4) Weighting.

In our application, the measurement noises are considered Gaussian whose means are zero, and whose error covariance matrices are, respectively R_k^{gyro}, R_k^{acc}, and R_k^{GPS}. In this case, the weights $\Omega_k^{(i)}$ are computed as follows:

$$\Omega_k^{(i)} = \frac{\exp(-\frac{1}{2}\left\|y_k - h(X_k^{(i)})\right\|_{R_k}^2)}{\sum_{j=1}^N \exp(-\frac{1}{2}\left\|y_k - h(X_k^{(j)})\right\|_{R_k}^2)} \Omega_{k-1}^{(i)} \tag{11}$$

$$(\|\cdot\|_{R_k}^2 = (\cdot)^T R_k^{-1}(\cdot))$$

where y_k is the observed measurements from GPS or inertial sensors and h is the measurement model, corresponding either to h_{GPS}, h_{gyro} or h_{acc} according to the availability of the measurement data.

5) Estimation.

Compute the output of the SIR filter by:

$$\hat{X}_k^N = \sum_{i=1}^N \Omega_k^{(i)} \cdot X_k^{(i)} \tag{12}$$

Increase *k* and iterate to item (3).

Since GPS data are obtained at a slower rate than the inertial sensor data, the filter will perform object pose estimation when gyroscope data, accelerometer data or GPS data is available. Thus, we implement a complementary filter as shown in **Figure 2**. There are two particles weighting channels sharing a common estimation module: one is for GPS measurements and the other is for inertial sensor measurements. Independent channel processing handles incomplete information measurements. For example, when no GPS measurement is available (e.g., due to occlusions), the overall system maintains object pose tracking by only using the inertial weighting channel vice versa, when GPS measurement is available, only the GPS weighting channel is used to estimate object pose to overcome the problems of inertial sensor drift due to longer integrated time.

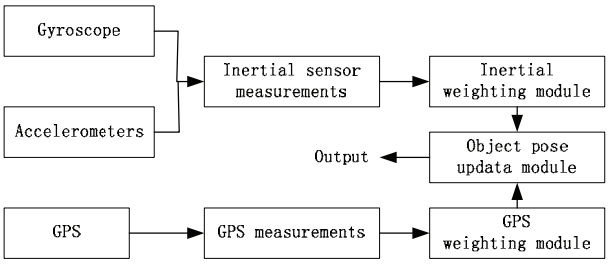


Figure 2. Block diagram of the fusion filter.

4. Experiments

This experiment evaluates the accuracy and the robustness of our fiducially detection and recognition method. We propose to use the particle filter framework for the sensor fusion system on MEMS-IMU/GPS. Particle filters are sequential Monte-Carlo methods based upon a point mass (or 'particle') representation of probability densities, which can be applied to any state space model and which generalize the traditional Kalman filtering methods. We have tested our algorithm to evaluate its performance and have compared the results obtained by the particle filter with those given by a classical extended Kalman filter.

Experimental data are presented in **Figure 3**.

First, the experiments research based on Matlab simulation soft, which is the language of technical computing. The original data of a pilot study based on the inertia output data of the inertial measurement unit and real-time GPS location and speed, which is used to the particle filtering data fusion arithmetic. The first result is taken on one-dimensional position above the error data analysis as **Figure 3**. Then, and respectively, used Kalman filter、the traditional extended Kalman filter、the unscented Kalman filter and particle filter to data fusion experiment. Three filters arithmetic of the first are not to introduce, particle filter specific reference to the course of the last section, the formula is derived, and available Experimental results are presented in **Figure 4**.

Finally, a new technique augmenting the powerful PF predictor with the traditional KF for improving the integrated MEMS-IMU/GPS integration system performance is presented. Initial test results show the significance of the proposed PF augmentation in reducing position and velocity drifts during GPS outages.

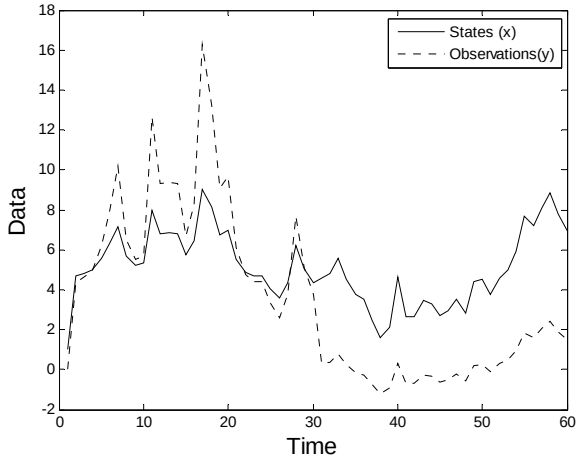


Figure 3. Emulational data of experiment.

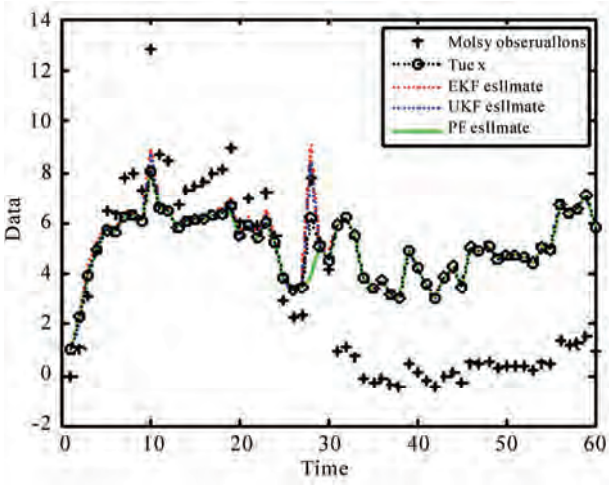


Figure 4. Result of the fusion filter.

Table 1. The RMSE of the fusion filter.

	Origin	EKF	UKF	PF
RMSE	3.5026	1.4749	1.4443	0.0201

5. Conclusions

In this paper we presented a hybrid AR approach which uses a particle filter to combine GPS and inertial technologies in order to improve the stability and the accuracy of the registration between the virtual and real world objects when enhancing the user perception.

An overview of the developed navigation system was described, and experiments demonstrated the feasibility and reliability of the system under various situations. Otherwise, we have implemented a SIR particle filter to fuse inertial and GPS data and, thus, to estimate the object poses. We have used the RMSE analysis to describe the performances of the filter. The results have been very satisfactory compared to those of classical AR techniques; they showed that the fusion method using the particle filter achieves high tracking accuracy, stability and robustness.

It is possible to apply more than three distributions to the PF and there are more error states to be investigated in real situations. Therefore the relations between the number of distributions and the filter performances will be investigated in the future work.

6. References

[1] P. Lang, A. Kusej, A. Pinz and G. Brasseur, "Inertial Tracking for Mobile Augmented Reality," *IEEE Instrumentation and Measurement Technology Conference*, Anchorage, AK, USA, 2002, pp. 1583-1587.

[2] E. Marchand, P. Bouthemy and F. Chaumette, "A 2D-3D Model-Based Approach to Real-Time Visual Tracking," *Image Vision Computer*, Vol. 19, No. 13, 2001, pp. 941-955.

[3] M. S. Grewal, L. R. Weill and A. P. Andrews, "Global Positioning Systems, Inertial Navigation, and Integration," 2nd Edition, John Wiley & Sons, Hoboken, 2007.

[4] Y. Yi and D. A. Grejner-Brzezinska, "Nonlinear Bayesian Filter: Alternative to the Extended Kalman Filter in the GPS/INS Fusion Systems," *Proceedings of the Institute of Navigation*, *ION GNSS* 2005, Long Beach, CA, 2005, pp. 1391-1400.

[5] Y. Kubo, T. Sato and S. Sugimoto, "Modified Gaussian Sum Filtering Methods for INS/GPS Integration," *Journal of Global Positioning Systems*, Vol. 6, No. 1, 2007, pp. 65-73.

[6] M. Nishiyama, S. Fujioka, Y. Kubo, T. Sato and S. Sugimoto, "Performance Studies of Nonlinear Filtering Methods in INS/GPS In-Motion Alignment," *Proceedings of the Institute of Navigation*, *ION GNSS* 2006, Fort Worth, TX, 2006, pp. 2733-2742.

[7] L. Chai, W. Hoff and T. Vincent, "Three-Dimensional Motion and Structure Estimation Using Inertial Sensors and Computer Vision for Augmented Reality," *Presence*: *Teleoperators and Virtual Environments*, Vol. 11, No. 5, 2002, pp. 474-492.

[8] M. Maidi, F. Ababsa and M. Mallem, "Vision-Inertial System Calibration for Tracking in Augmented Reality," *Proceedings of the* 2*nd International Conference on Informatics in Control*, *Automation and Robotics*, Barcelona, 2005, pp. 156-162.

[9] S. You and U. Neumann, "Fusion of Vision and Gyro Tracking for Robust Augmented Reality Registration," *Proceedings of IEEE Conference on Virtual Reality*, Washington, DC, 2001, pp. 71-77.

[10] A. Doucet, N. J. Gordon and V. Krishnamurthy, "Particle Filters for State Estimation of Jump Markov Linear Systems," *IEEE Transactions on Signal Processing*, Vol. 49, No. 3, 2001, pp. 613-624.

[11] N. Bergman and A. Doucet, "Markov Chain Monte Carlo data Association for Target Tracking," *Proceedings of IEEE International Conference on Acoustics*, *Speech*, *and Signal Processing*, Istanbul, 2000, pp. 705-708.

[12] N. J. Gordon, "A Hybrid Bootstrap Filter for Target Tracking in Clutter," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 33, No. 1, 1997, pp. 353-358.

[13] A. Doucet, N. de Freitas and N. Gordon, "An Introduction to Sequential Monte Carlo Methods," In: A. Doucet, N. de Freitas and N. Gordon, Eds., *Sequential Monte Carlo Methods in Practice*, Springer, New York, 2001, pp. 3-14.

Intelligent Information Management, **2010, 2, 422-430**
doi:10.4236/iim.2010.27052 Published Online July 2010 (http://www.SciRP.org/journal/iim)

Predicting ERP User Satisfaction—an Adaptive Neuro Fuzzy Inference System (ANFIS) Approach

Chengaleth Venugopal[1], Siva Prasanna Devi[2], Kavuri Suryaprakasa Rao[2]
[1]*Faculty of Management Studies, Anna University, Chennai, India*
[2]*Department of Industrial Engineering, Anna University, Chennai, India*
E-mail: {*cvenugopal*, *prasannasiva*11}*@gmail.com*, *ksprao@annauniv.edu*
Received April 7, 2010; *revised May* 12, 2010; *accepted June* 21, 2010

Abstract

ERP projects' failing to meet user expectations is a serious problem. This research develops an Adaptive Neuro Fuzzy Inference System (ANFIS) model, to predict the key ERP outcome "User Satisfaction" using causal factors present during an implementation as predictors. Data for training and testing the models was from a cross section of firms that had implemented ERPs. ANFIS is compared with other prediction techniques, ANN and MLRA. The results establish that ANFIS is able to predict outcome well with an error (RMSE) of 0.277 and outperforms ANN and MLRA with errors of 0.85 and 0.86 respectively. This study is expected to provide guidelines to managers and academia to predict ERP outcomes ex ante, and thereby enable corrective actions to redirect ailing projects.

Keywords: ANFIS, ERP Implementation Outcome, Prediction, Failure Detection, CSFs, Causal Factors

1. Introduction

The track record of successful IT projects of which Enterprise Resource Planning (ERP) is a subset projects remains poor. The latest CHAOS study of the Standish Group reports a marked decrease in IT project success rates, with only 32% succeeding in "on time" and "on budget" delivery with required features and functions. 44% were delivered late or over budget, and/or with less than the required features and functions. 24% were cancelled prior to completion or delivered and never used [1]. This is worse than the figures of about decade back as observed by Robey in 2002: "About half of ERP projects fail to achieve anticipated benefits" [2].

Information systems (IS) project failures often encounter project "escalation" defined as a continued commitment to a failing course of action despite "uncertainty surrounding the likelihood of goal attainment" [3]. Escalation research lists issues that cause escalation and suggests strategies for de-escalation which includes abandoning or "redirecting" the project [4,5]. While these are acceptable as reactive steps a proactive approach of predicting impending failures, would be invaluable as one could then attempt to forestall or at least redirect the project far better.

The essence of proactive control is having predictive capabilities. The challenge is to move from the diagnosis of the source of past problems to the prediction and forecasting of potential problems in new projects [6]. Can a robust, easy to use and reliable predictor be developed that would "red flag" impending failures in ERP implementations? This is the research question we seek to answer in this paper. This research has developed a method of predicting *User Satisfaction*, a key measure of ERP project success using *ex ante* causal factors as predictors.

This study consolidates and extends an earlier study which gathered data from a cross section of business organizations that had implemented ERP systems in the last three years and developed and tested a measurement model for causal factors for success [7]. Data was collected, using a structured questionnaire, on Critical Success Factors (CSFs), identified in literature as being causal for the success of an ERP implementation [8] and overall *User Satisfaction* , a key indicator of the success [9,10]. Respondents to our questionnaire represented different user cohorts: *Strategic Users*, *Technical Users* and *Operational Users*. The validity and reliability of the measurement model and its innate value as a predictor of ERP success was established using Structural Equation Modeling (SEM) with LISREL 8.7.

In the present study the data from the earlier study was used to develop predictive models for ERP implementation outcomes measured in terms of *User Satisfaction*.

Three prediction techniques, Multiple Linear Regression Analysis (MLRA), Artificial Neural Networks (ANN) and Adaptive Neuro Fuzzy Inference System (ANFIS) were tested. Of the three ANFIS was found to be significantly better in predicting *User Satisfaction* of an ERP project.

This paper is organized as follows: Section 2 presents the literature review and establishes the need and relevance of this research work. Section 3 outlines the method used in the research. This section also explains different prediction techniques with specific emphasis on ANFIS. Section 4 presents the results of the modeling and compares the results of the various techniques used. Section 5 concludes the paper with the direction for continuing research.

2. Literature Review

We looked at the several studies on the Critical Success Factors (CSFs) of an ERP implementation. CSFs can best be defined in the words of Somers and Nelson [8] as "key players and activities...playing a pivotal role in (determining) an organization's experience with the ERP implementation". By their very definition then, CSFs should be good predictors or causal factors as key determinants of outcome—"the organization's experience" or *User Satisfaction*. In this study, therefore, we have used *CSFs* and *Causal Factors* interchangeably.

One of the earliest researchers on CSFs, Davenport [11] identifies six factors mostly relating to the directional or strategic aspects of an ERP project. These include factors related to top management support, use of a cross functional steering committee, communication, cross functional implementation teams etc.

Parr and Shanks [12] focus on the process management aspects of an implementation. The key factors identified include management support, an organizational commitment to change and appropriate definition of scope. Hong and Kim [13] stress on the organization preparedness aspects of the implementation namely organizational fit, system adaptation levels, resistance to change, etc.

Somers and Nelson [8] work considered the most comprehensive [14] identifies a set of 22 CSFs. They used the Cooper and Zmud's [15] six stage model of IT implementation to track the importance of different factors across the six stages of: *initiation, adoption, adaptation, acceptance, routinization* & *infusion.*

CSFs have been listed extensively but not too many attempts are evident to group and measure CSFs present in an implementation. Attempts at grouping in literature are: Parr *et al*. who suggested into four categories [12]: 1) *Management* 2) *Software* 3) *Project* & 4) *People*. Wixom and Watson [16] suggest three groups: *Organizational, Project* & *Technical*. Holland and Light [17] two groups: *Strategic & Tactical.*

The other research gap we found is the lack of studies that systematically addressed the issue of ERP project risk detection. In his doctoral desertaion Marbach [18] suggests the following nomological network to categorise IT risk literature: *describe, identifiy, detect, assess* & *address*. While there has been a fair amount of work relating to the first two, not too much work is evident in the areas of *detect* and *assess* and much less in the area of *address*. These areas seem to be dominated by practitioner approaches and case studies with not too many empirical studies. Finally, we could not find research that combined empirical data with a dynamic modeling approach which would address both issues of detection and assessing and provide through simulation, a means to address the third—"*addressing the risk*".

In addition, there have been several studies over the last two and a half decades on measuring IS success with one of the earliest studies being the work of Davis [19] who cites "Lack of user acceptance" as the main impediment of success. Other researchers who studied enterprise systems success and corrective measures include Markus and Tannis [20], Shang and Seddon [21] and Myers *et al*. [22]. The often cited work on IS systems success of DeLone and McLean [9,10] posit six major dimensions of Information success and identify *User Satisfaction* as "probably the most widely used single measure of I/S success" [9]. In conclusion, study of literature establishes the validity of causal factors in impacting I/S project success and confirms the use of overall user satisfaction as a good and acceptable omnibus measure for determining success or failure of an I/S implementation project.

3. The Method

The conceptual model underlying the present study is given in **Figure 1**.

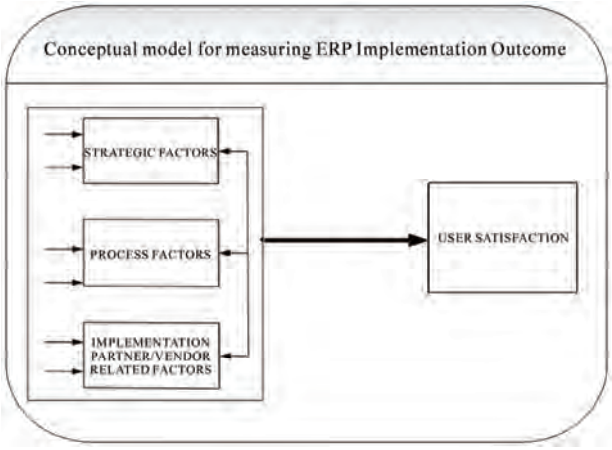


Figure 1. Conceptual model.

Through an earlier empirical study [7], data was collected from a cross section of around 60 organizations and 156 respondents representing three user cohorts: *Strategic, Technical & Operational* responded to a pretested and validated (for content validity) structured questionnaire. Respondents rated the CSFs present in their organizations during ERP implementation. The CSFs list used for this research was drawn from prior research, and confirmed by an expert panel as relevant for the current context. The CSFs were also validated as relevant as per Structuration, Expectations-Confirmation, Lewins Change and Agency theories. Responses were captured on a Likert scale with end values of 5 = *Completely Agree* and 1 = *Completely Disagree*. From the same set of respondents their overall satisfaction, a measure of *Success* of the ERP project was also captured on a seven point Likert Scale with end values of 7 = *Completely Satisfied* and 1 = *Completely Dissatisfied*.

The data collected was checked and 142 responses were found to be complete and used for further analysis. Exploratory Factor Analysis (EFA) with SPSS 7.0 using Principle Component Analysis and Varimax rotation, revealed the presence of three distinct constructs, that logically mapped to *Process Factors*, *Strategic Factors*, and *Vendor* (related) *factors*. Confirmatory Factor Analysis (CFA) using Structural Equation Modeling (SEM) using LISREL 8.7 confirmed the construct validity, and discriminant validity of the constructs in addition to confirming good overall model fit. **Table 1** gives the complete set of measures and the constructs that they map to.

Scale reliability tests were conducted—**Table 2**. High factor loadings in excess of 0.7 and high Croanbach Alpha values in excess of 0.8 confirmed the additivity of the measures [23]. This allowed for the creation of summated factors scores using the Bartlet method which was found to give better reliability when compared to the Andersen—Rubin method. These factors scores for *Strategic, Process* and *Vendor* factors were used as the predictors and *User Satisfaction* as the dependent variable.

The overall dataset consisted of 142 responses. This was used for the model building and testing exercise. A sample of the data set showing the independent variables as well as the dependent variable for the model is given in **Table 3**.

Table 1. Construct and measures.

Predictors used for Modelling			
CRITICAL SUCCES FACTORS(MEASURES)[1]	**Label**	**CONSTRUCT[2]**	**FACTOR SCORES[3]**
Scope (clarity of)	P1	PROCESS	PF
Cooperation between departments	P2		
Legacy data (quality of)	P3		
Contingency Planning	P4		
Package Selection	P5		
Dedicated team	S1	STRATEGIC	SF
Team training	S2		
Champion (presence of)	S3		
Steering Committee (regular functioning of)	S4		
Top management Commitment	S5		
BPR	S6		
Consultants(skill and competence)	V1	VENDOR	VF
ERP Vendor(cooperation)	V2		
Tools and techniques(of implementation partner)	V3		

Notes:

1. Survey instrument got responses on a scale of 5-1

2.Construct confirmed through EFA followed by CFA

3. Summated scale created using Factor scores by Bartlet method

Table 2. Scale reliability test.

CAUSAL FACTORS—MEASURES	CONSTRUCT	Cronbach's Alpha Score
Scope (clarity of)		
Cooperation between departments		
Legacy data (quality of)	PROCESS	0.871
Contingency (planned for)		
Package Selection		
Dedicated team		
Team training		
Champion (presence of)		
Steering Committee (regular functioning of)	STRATEGIC	0.908
BPR (carried out)		
Top management Commitment		
Consultants(skill and competence)		
ERP Vendor (cooperation)	VENDOR	0.808
Tools and techniques (of implementation partner)		

Table 3. Sample dataset—training and test.

	Predictors—Factor Scores (*Bartlett method-with mean* = 0)			**Dependent Variable (7 to 1)**
Sl.No.	Process	Strategic	Vendor	**User Satisfaction**
1	-1.38304	-0.77426	0.69434	4
2	-0.49719	-0.62034	2.14214	5
3	0.33652	-0.71615	0.50457	5
4	-0.56822	-0.08176	1.35287	5
5	-0.19034	-0.36148	0.69369	6
6	0.99333	-0.72094	-0.60319	6
7	-2.21601	-0.81693	-0.8998	1
8	1.12723	0.32234	1.22514	6
9	0.86559	-1.52226	-0.70272	4
10	-0.28925	-1.53267	-1.49165	4

4. Modeling—Results and Discussions

Three different prediction techniques were used: 1) Mutiple Linear Regression Analysis (MLRA), 2) Artificial Neural Networks (ANN) and 3) Adaptive Neuro-Fuzzy Inference System (ANFIS). In all cases about 70% of the data was used to build/train the model. The balance 30% of the data was used for testing the model. Each of these is discussed in the following paragraphs.

4.1. Multiple Linear Regression Analysis (MLRA)

Linear least squares regression analysis is still the most common technique used, as observed in the literature [24]. Being a pure statistical technique MLRA has a few important underlying assumptions. These are 1) "linearity"—the assumption that the predictor variable is linearly related to the dependent variable, 2) no "multicolloinearity"—the individual predictors are not correlated to each other, 3) no "heteroscadacity"—the error variances of the predictor variables are constant across the range of data. These conditions make the use of MLRA restrictive especially when modeling issues related to human jugdement where multicollinearity and heteroscadacity are sometimes unavoidable [25]. However, despite its limitations MLRA is an established technique and this study compares the results of MLRA with results obtained from other prediction techniques.

4.2. Artificial Neural Networks (ANN)

The most common model-building technique identified in the literature as an alternative to MLRA is back-propagation trained feed-forward neural networks [24] often referred to simply as back-propagation networks. ANNs are complex and flexible nonlinear systems with the ability to deal with noisy or incomplete input patterns, high fault tolerance, and the ability to generalize from the input data. [26]

Neural networks excel at applications where pattern recognition is important, and precise computational answers are not required [24]. ANN works on the principle of an adaptive learning algorithm and uses an information processing system composed of a large number of interconnected processing elements (neurons) working in tandem.

Neural networks are made of basic units arranged in layers. A unit collects information provided by other units (or by the external world) to which it is connected with weighted connections called synapses. These weights, called synaptic weights multiply (*i.e.*, amplify or attenuate) the input information. A positive weight is considered excitatory, a negative weight inhibitory. One of the most popular architectures in neural networks is the multi-layer perceptron which is illustrated in **Figure 2**.

Learning happens through a methodology of continuously altering the weights to achieve closer and closer values to the desired outputs. One algorithm that performs this is known as the back propogation algorithm. The back propagation algorithm is a generalization of the

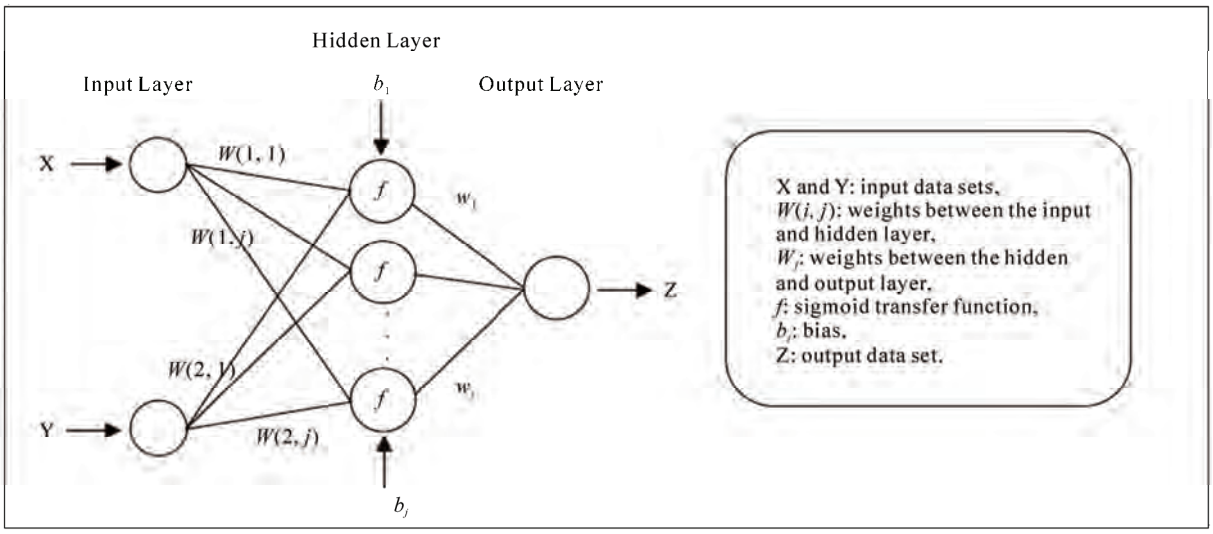


Figure 2. ANN-multilayered perceptron.

least mean square algorithm. The network weights are modified to minimize the mean squared error between the desired and actual outputs of the network. The network is trained using a training data set where the input and output values are known. After the training is completed, the weights are frozen and the model can be used for prediction of outputs for new sets of input values.

4.3. Adaptive Neuro-Fuzzy Inference Systems (ANFIS)

While ANN is a good technique that emulates the way a human brain makes a judgement, a limitation is the way it handles the input data. In the case of human reasoning, input data need not always be crisp but could have linguistic labels like "small", "high"etc. Also, the response to the data need not always follow a strict "yes-no" rule but could have a range of responses across a continuum. Such a pattern of responses is referred to as the membership function and such reasoning is called "fuzzy" reasoning. A fuzzy inference system using fuzzy rules can model qualitative aspects of human behavior. This was first explored by Takagi and Sugeno [27] and has since been used in numerous applications involving predictions [28].

Fuzzy inference systems are composed of five functional blocks as given in **Figure 3**. These are 1) a rule base containing a number of if-then rules 2) a database which defines the membership function, 3) a decision making interface that operates the given rules 4) a fuzzification interface that converts the crisp inputs into "degree of match "with the linguistic values like high or low etc., and 5) a de fuzzification interface that reconverts to a crisp output [28].

Adaptive Neuro Fuzzy Inference system (ANFIS) is a hybrid technique which combines the adaptive learning capability of ANN along with the intuitive fuzzy logic of human reasoning formulated as a feed-forward neural network. Hence, the advantages of a fuzzy system can be combined with a learning algorithm [28]. Fusion of Artificial Neural Networks (ANN) and Fuzzy Inference Systems (FIS) is used by researchers in various scientific and engineering areas due to the growing need of adaptive intelligent systems to solve the real world problems. ANN learns by adjusting the weights of interconnections between layers. FIS uses fuzzy set theory, fuzzy if-then rules, and fuzzy reasoning. Given in the following section are details of the model developed.

A typical ANFIS consists of five layers, which perform different actions in the ANFIS are detailed below. For simplicity, we have illustrated a system that has two inputs *x* and *y* and one output Z. The rule base, for illustrative purposes consists of two if-then rules of the Takagi-Sugeno type.

Layer 1: All the nodes in this layer are adaptive nodes. They generate membership grades of the inputs. The node function is given by:

$$O^1_{A_i} = \mu_{A_i}(x) \quad , \quad i = 1,\ 2$$
$$O^1_{B_j} = \mu_{B_j}(y) \quad , \quad j = 1, 2 \tag{1}$$

where *x* and *y* are inputs and A_i and B_j are appropriate membership functions (MF's), which can be triangular, trapezoidal, Gaussian functions or other shapes. In our study, the Gaussian MF's has been utilized and three input parameters are: *Process*, *Strategic* and *Vendor*.

Layer 2: The nodes in this layer are fixed nodes which multiply the inputs and send the product out. The outputs of this layer are represented as:

$$W_i = \mu_{A_i}(x)\ x\ \mu_{B_j}(y) \quad , \quad i, j = 1, 2 \tag{2}$$

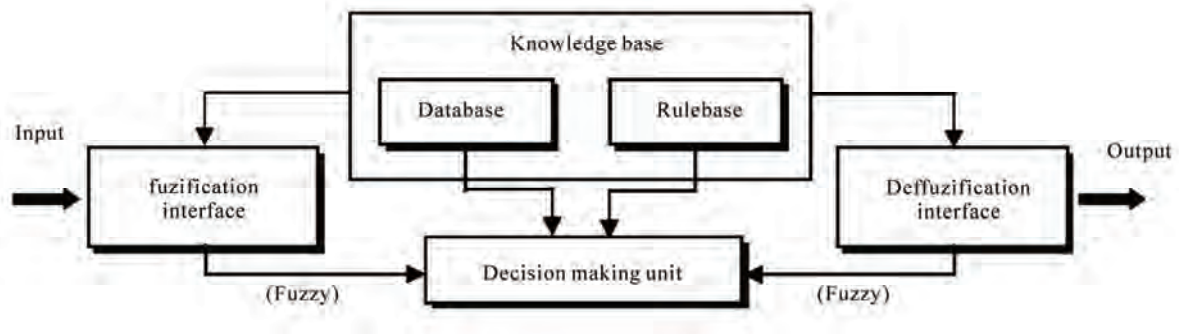


Figure 3. Fuzzy inference system.

Layer 3: The nodes in this layer are also fixed nodes. It calculates the ratio of a rule's firing strength to sum of the firing strengths of all the rules. This action is represented as follws:

$$\overline{W}_{ij} = \frac{W_{ij}}{\sum_{i=1}^{2}\sum_{j=1}^{2} W_{ij}} \quad , \quad i, j = 1, 2 \tag{3}$$

This is called normalized firing strength.

Layer 4: Each node in this layer is an adaptive node, whose output is simply the product of the normalized firing strength and a first-order polynomial (for a first order Sugeno model).

Thus, the outputs of this layer are given by:

$$O_{ij}^{4} = \overline{W}_{ij} f_{ij} = \overline{W}_{ij}(p_{ij}x + q_{ij}y + r_{ij}) \quad , \quad i, j = 1, 2 \tag{4}$$

Parameters in this layer are referred to as consequent parameters.

Layer 5: The single node in this layer computes the overall output as the summation of all incoming signals, *i.e.*

$$\begin{aligned} Out = O^{5} &= \sum_{i=1}^{2}\sum_{j=1}^{2} \overline{W}_{ij} f_{ij} = \sum_{i=1}^{2}\sum_{j=1}^{2} \overline{W}_{ij}(p_{ij}x + q_{ij}y + r_{ij}) \\ &= \sum_{i=1}^{2}\sum_{j=1}^{2} [(\overline{W}_{ij}p_{ij})x + (\overline{W}_{ij}q_{ij})y + (\overline{W}_{ij}r_{ij})] \end{aligned} \tag{5}$$

where the overall output *Out* is a linear combination of the consequent parameters when the values of the premise parameters are fixed. It uses Sugeno type fuzzy inference systems and Gaussian membership function is used to train the given data set. The ANFIS structure used is illustrated in **Figure 4**.

This model uses eight rules which are as given below:

1) (Process = in1mf1) & (Strategic = in2mf1) & (Vendor = in3mf1) => (UserSatisfaction = out1mf1)

2) (Process = in1mf1) & (Strategic = in2mf1) & (Vendor = in3mf2) => (UserSatisfaction = out1mf2)

3) (Process = in1mf1) & (Strategic = in2mf2) & (Vendor = in3mf1) => (UserSatisfaction = out1mf3)

4) (Process = in1mf1) & (Strategic = in2mf2) & (Vendor = in3mf2) => (UserSatisfaction = out1mf4)

5) (Process = in1mf2) & (Strategic = in2mf1) & (Vendor = in3mf1) => (UserSatisfaction = out1mf5)

6) (Process = in1mf2) & (Strategic = in2mf1) & (Vendor = in3mf2) => (UserSatisfaction = out1mf6)

7) (Process = in1mf2) & (Strategic = in2mf2) & (Vendor = in3mf1) => (UserSatisfaction = out1mf7) (1)

8) (Process = in1mf2) & (Strategic = in2mf2) & (Vendor = in3mf2) => (UserSatisfaction = out1mf8) (1)

The graphical representation of the eight rules is given in **Figure 5**.

After training (using 99 data sets), the model was tested with the balace 44 data sets. The results of the

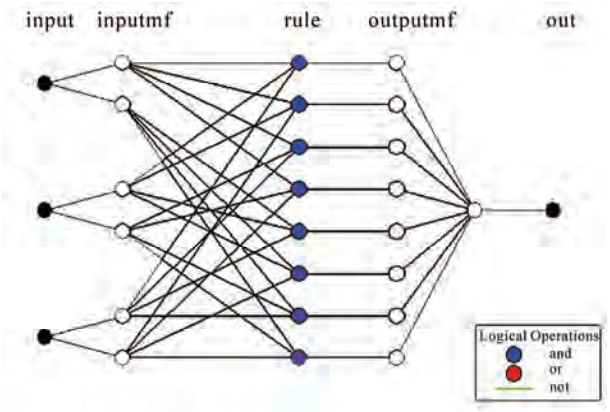


Figure 4. ANFIS structure.

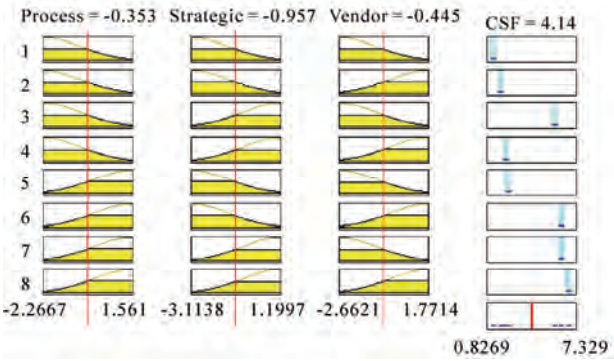


Figure 5. ANFIS rules.

predicted data were compared with the actual data are given in **Figure 6**.

The contour diagrams of the output and its predictor variables are given in **Figures 7(a)**, **7(b)**, **7(c).**

4.3.1. Prediction with ANN

The prediction was carried out using ANN. The ANN structure used in our study is given in **Figure 8**. The same training set of 99 used for the ANFIS model was used for the ANN modelling as well. The testing set consisted of the same 44 that were used to test the ANFIS model.

The model developed is able to predict User Satisfaction with a Root Mean Square Error (RMSE) of 0.85 and a Mean Average Percentage Error (MAPE) of 0.195. The prediction results of the ANN model for the test data is as given in **Figure 9**.

4.3.2. Prediction with MLRA

The predictors given in **Table 4** above were regressed against *User Satisfaction* which is a measure of ERP success. The resulting regression equation is given in Equation (6). **Table 4** lists the coefficients and the results of the significance tests.

User Satisfaction = 4.86 + 0.812 Process + 0.496 Strategic – 0.010 Vendor (6)

Process and *Strategic* predictors have a very strong influence on the prediction of *User Satisfaction* with P < 0.05, but the *Vendor* predictor (with P > 0.05), does not contribute much to the prediction of *User Satisfaction*.

The model is tested for assumption violations of multiple regressions. Plotting the residuals versus the predicted variable is a basic method of identifying such violations. An assumption often encountered violation in non-normality. A diagnostic for the same is a histogram of the residuals as well as a normality probability plot of residuals. A normal distribution makes a straight diagonal line and the residual line closely following the diagonal indicates that the distribution is normal [10]. **Figure 10** gives the residual plots and as can be seen the specified model does not violate the core assumptions of multiple regression.

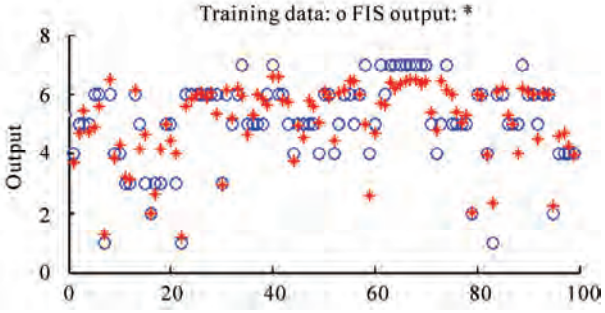
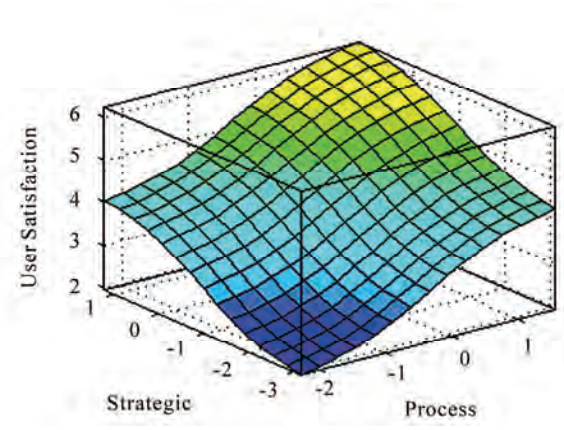
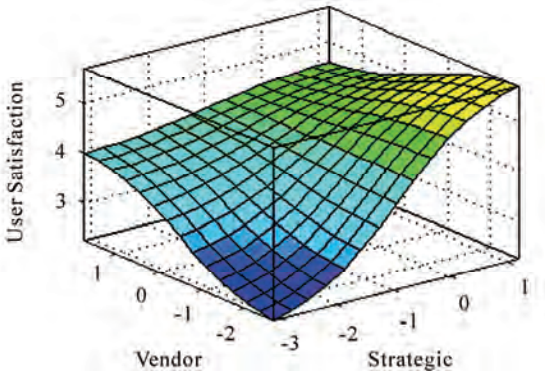


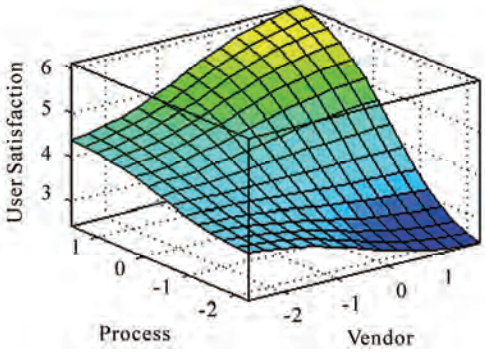
Figure 6. Prediction results.



(a)



(b)



(c)

Figure 7. (a) Contour diagram; (b) contour diagram; (c) contour diagram.

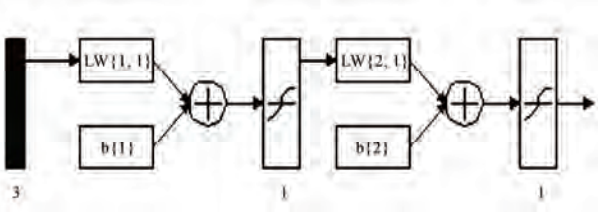


Figure 8. ANN structure.

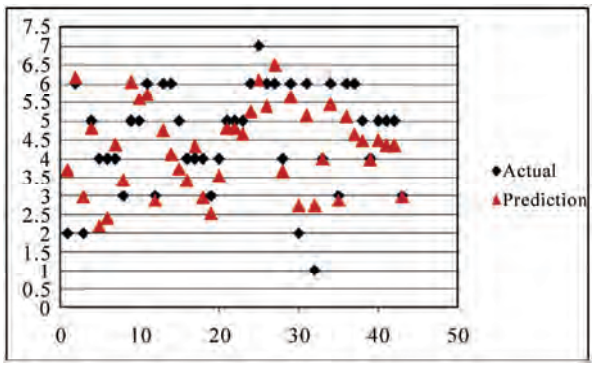


Figure 9. ANN prediction results.

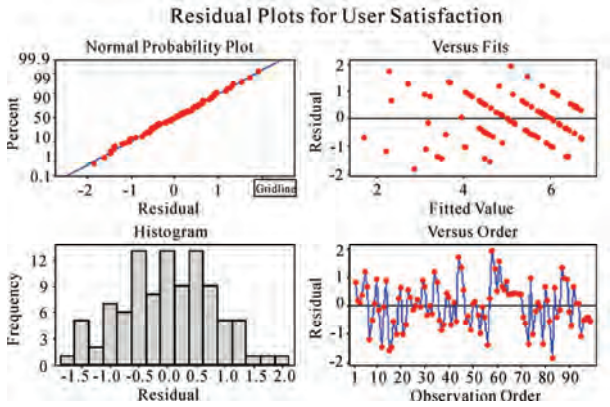


Figure 10. Residual plots for user satisfaction.

Table 4. Regression results.

Predictors	Coef	SE Coef	T	P
Constant	4.86012	0.08375	58.03	0
Process	0.8118	0.1335	6.08	0
Strategic	0.4959	0.128	3.87	0
Vendor	-0.0098	0.107	-0.09	0.927

S = 0.803159, R-Sq = 69.5%, R-Sq(adj) = 68.5%

4.3.3. Models Comparision

Figure 10 compares the results of prediction using MLRA, ANN and ANFIS with the actual outcomes. The Mean Average Percentage Error (MAPE) and Root Mean Square Error (RMSE) are used to test the efficacy of the prediction techniques. They are calculated using Equations (7) & (8). Lower values are indicative of better fit. As can be seen from **Table 5** ANFIS outperforms both ANN and Regression significantly.

$$MAPE = \frac{1}{n_t}\sum_{i=1}^{n_t} 100 \times \left|\frac{a_i - p_i}{a_i}\right| \quad (7)$$

$$RMSE = \sqrt{\frac{1}{n_t}\sum_{i=1}^{n_t}(a_i - p_i)^2} \quad (8)$$

Table 5. Comparitive results.

	ANFIS	ANN	MLRA
RMSE	0.277411	0.852738	0.865293
MAPE	0.04407	0.195142	0.20079

where a is actual value, p is predicted value and n_t is the number of testing samples. The results show that ANFIS has a significantly better prediction capability. The RMSE of ANFIS is almost three times better than the other two methods. The MAPE results are also much better for ANFIS. This establishes that ANFIS as a modelling tool is an excellent predictor of ERP implementation outcomes.

5. Conclusions

This study has modeled the ERP Implementation process, using causal factors *Strategic*, *Process* and *Vendor* as predictor variables and *User Satisfaction* as the dependent variable. These factors represent the relevant causal factors that impact the success or failure of an ERP implementation in terms of *User Satiscaftion.*

Data for the modeling is from a prior study that developed and tested a measurement model for assessing the causal factors for ERP implementation outcomes using Structural Equation Modeling (SEM). About 150 respondents from about 60 business organizations representing different user cohorts, Strategic, Operational and Technical, responded to a structured questionnaire. This generated the data on the causal factors and also *User Satisfaction*. This data was used for further analysis in this study.

We developed/trained Multiple Linear Regression Analysis (MLRA), Artificial Neural Network (ANN) and Adaptive Neuro Fuzzy Inference System (ANFIS) prediction models using part of the dataset (99 responses) and tested using the balance (43 responses). Of the three techniques ANFIS outperformed ANN and MLRA in terms RMSE and MAPE.

The study established the efficacy of ANFIS as a good predictor of project risk of ERP implementations measured in terms of *User Satisfacion* considered as a good measure to evaluate overall IS success. It has important significance to practitiones who can use the survey instrument developed along with the model to asses the risk of their ERP project very early in the implementation cycle. The model developed can be used by the management to asses the predicted *User Satisfaction* levels well in advance and thereby take appropriate corrective measures. The utility would be further enhanced if the model developed could be supplemented with a decision support system (DSS) that would help practioners simulate the outcome of the implementations dynamically by

altering the value of the three predictor factors and assessing the response of the model in terms of user satisfaction. This is the continuing work that is being carried out by the authors and would be reported in a subsequent paper.

6. References

[1] New Standish Research Report, "Roadmap to the Megapples," 2009. http://www.standishgroup.com

[2] D. Robey, J. Ross and M. Boudreau, "Learning to Implement Enterprise Systems: An Exploratory Study of the Dialectics of Change," *Journal of Management Information Systems*, Vol. 19, No. 1, 2002, pp. 17-46.

[3] J. Brockner, "The Escalation of Commitment to a Failing Course of Action: Towards Theoretical Progress," *Academy of Management Review*, Vol. 17, No. 1, 1992, pp. 39-61.

[4] M. Keil, "Pulling the Plug: Software Project Management and the Problem of Project Escalation," *MIS Quarterly*, Vol. 19, No. 4, 1995, pp. 421-447.

[5] M. Keil, "Why Software Projects Escalate—an Empirical Study and Analysis of Four Theoretical Models," *MIS Quarterly*, Vol. 24, No. 4, 2000, pp. 631-664.

[6] A. J. Al-Shehab, R. T. Hughes and G. Winstanley, "Modelling Risks in IS/IT Projects through Causal and Cognitive Mapping," *The Electronic Journal of Information Systems Evaluation*, Vol. 8, No. 1, 2005, pp. 1-10.

[7] C. Venugopal and S. Rao, "Detecting Project Risks in ERP Projects Measurement Models for Critical Success Factors and Success of ERP Implementations," *Proccedings of International Conference on Advances in Industrial Engineering Applications*, Chennai, India, 2010.

[8] T. M. Somers and K. Nelson, "The Impact of Critical Success Factors across the Stages of Enterprise Resource Planning Implementations," *Proceedings of the 34th Annual Hawaii International Conference on System Sciences*, Hawaii, 2001, pp. 8016-8025.

[9] W. H. DeLone and E. R. McLean, "Information Systems Success: The Quest for the Dependent Variable," *Information Systems Research*, Vol. 3, No. 1, 1992, pp. 60-95.

[10] W. H. DeLone and E. R. McLean, "The Delone and Mclean Model of Information Systems Success—a Ten Year Update," *Journal of Information Systems*, Vol. 19, No. 4, 2003, pp. 9-30.

[11] T. H. Davenport, "Putting the Enterprise into the Enterprise System," *Harvard Business Review*, Vol. 76, No. 4, 1998, pp. 121-131.

[12] A. Parr and G. Shanks, "A Model of ERP Project Implementation," *Journal of Information Technology*, Vol. 15, No. 4, 2000, pp. 289-303.

[13] K. K. Hong and Y. G. Kim, "The Critical Success Factors for ERP Implementations: An Organizational Fit Perspective," *Information and Management*, Vol. 40, No. 1, 2002, pp. 25-40.

[14] J. Hedman, "Enterprise Resource Planning Systems: Critical Factors in Theory and Practice," Lund University, 2004.

[15] R. B. Cooper and R. W. Zmud, "Implementation Technology Implementation Research: A Technological Diffusion Approach," *Management Science*, Vol. 36, No. 2, 1990, pp. 123-139.

[16] B. H. Wixom and H. J. Watson, "An Empirical Investigation of the Factors Affecting Data Warehousing Success," *MIS Quarterly*, Vol. 25, No. 1, 2001, pp. 16-41.

[17] P. Holland, B. Light and N. Gibson, "A Critical Success Factors Model for Enterprise Resource Planning Implementation," *Proceedings of the 7th European Conference on Information Systems*, Vol. 1, 1999, pp. 273-297.

[18] A. T. Marbach, "Detecting Risk in Information Technology Projects," Doctoral Thesis, University of Texas, Arlington, 2003.

[19] F. D. Davis, "User Acceptance of Information Technology: System Characteristics, User Perceptions and Behavioral Impacts," *International Journal of Man Machine Studies*, Vol. 38, No. 3, 1993, pp. 475-487.

[20] M. L. Markus and C. Tanis, "The Enterprise System Experience-from Adoption to Success," In: R. W. Zmud, Ed., *Framing the Domains of IT Management*: *Projecting the Future through the Past*, *Chapter* 10, Pinnaflex Educational Resources Inc., Cincinnati, 2000, pp. 173-207.

[21] S. Shang and P. B. Seddon, "Assessing and Managing the Benefits of Enterprise Systems: The Business Manager's Perspective," *Information Systems Journal*, Vol. 12, No. 4, 2002, pp. 271-299.

[22] B. L. Myers, L. A. Kappelman and V. R. Prybutok, "A Comprehensive Model for Assessing the Quality and Productivity of the Information Systems Function," *Information Resources Management Journal*, Vol. 10, No. 1, 1997, pp. 6-25.

[23] J. F. Hair, W. Black, R. E. Anderson and R. L. Tatham, "Multivariate Data Analysis (6/E)," Pearson Education, 2008.

[24] B. Eftekhar, K. Mohammad, H. E. Ardebili, G. Mohammad and E. Ketabchi, "Comparison of Artificial Neural Network and Logistic Regression Models for Prediction of Mortality in Head Trauma Based on Initial Clinical Data," *BMC Medical Informatics and Decision Making*, Vol. 5, No. 3, 2005, pp. 1-8.

[25] A. R. Gray and S. G. MacDonell, "A Comparison of Techniques for Developing Predictive Models of Software Metrics," *Information and Software Technology*, Vol. 39, No. 6, 1997, pp. 425-437.

[26] D. W. Patterson, "Artificial Neural Networks: Theory and Applications," Prentice Hall, Englewood Cliffs, 1996.

[27] T. Takagi and M. Sugeno, "Fuzzy Identification of Systems and its Applications to Modeling and Control," *IEEE Transactions Systems, Man, Cybernetics*, Vol. 15, No. 1, 1985, pp. 116-132.

[28] J. S. R. Jang, "Adaptive-Network-Based Fuzzy Inference system," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 23, No. 3, 1993, pp. 665-685.

Intelligent Information Management, **2010, 2, 431-436**
doi:10.4236/iim.2010.27053 Published Online July 2010 (http://www.SciRP.org/journal/iim)

New Investigative Findings from the Debiased Converted-Measurement Kalman Filter

John N. Spitzmiller[1], Reza R. Adhami[2]
[1]*Cobham Analytic Solutions*, 401 *Diamond Drive, Huntsville, USA*
[2]*Department of Electrical and Computer Engineering, University of Alabama in Huntsville, Huntsville, USA*
E-mail: *john.spitzmiller@cobham.com, adhami@ece.uah.edu*
Received March 5, 2010; *revised April* 19, 2010; *accepted May* 27, 2010

Abstract

The original algorithm for the 2-D debiased converted-measurement Kalman filter (CMKF) specified, with incorrect mathematical justification, a requirement for evaluating the average true bias and covariance with the best available polar estimate, rather than exclusively with the polar measurement. Even though this original algorithm yields better tracking performance than the debiased-CMKF algorithm which evaluates the average true bias and covariance exclusively with the polar measurement, this paper shows the specified requirement compromises the statistical consistency between the debiased converted measurement's error and the average true covariance. To resolve this apparent contradiction, this paper provides the correct empirical explanation for the tracking-performance improvement obtained by the specified requirement.

Keywords: Tracking, Converted Measurements, Kalman Filter, Debiased CMKF, Polar-To-Cartesian Conversion

1. Introduction

In the original paper describing the 2-D debiased converted-measurement Kalman filter (CMKF) [1], Lerro and Bar-Shalom derived approximate but practical expressions for the converted measurement's error bias and covariance that depend on only the polar target-position measurement and the polar measurement's error statistics. They termed these quantities the "average true bias" and the "average true covariance," respectively. In order to improve the practical debiased CMKF's dynamic tracking performance, Lerro and Bar-Shalom further specified the additional requirement of evaluating the average true bias [2] and the average true covariance [1,2] using the best available polar estimate rather than evaluating them exclusively with the polar measurement. To provide a practical means of meeting this additional requirement, Lerro and Bar-Shalom presented a simple test which chooses the more accurate of the polar measurement and predicted polar estimate (obtained via nonlinear transformation of the CMKF's predicted Cartesian position components) based on the sizes of the respective error covariances in Cartesian coordinates. (Note that, strictly speaking, [1] only calls for the average true covariance to be evaluated with the best available polar estimate. However, [2] confirms that *both* the average true bias *and* the average true covariance should be evaluated with the best available polar estimate; the omitted call for evaluating the average true bias with the best available polar estimate was a publishing oversight. Furthermore, the simulation results of [1] were obtained by evaluating both the average true bias and the average true covariance with the best available polar estimate [2].)

Lerro and Bar-Shalom demonstrated good statistical consistency between the debiased converted measurement's error and the average true covariance for the "static case" *when both the average true bias and covariance were evaluated exclusively with the polar measurement*. They also demonstrated that, when compared with the previously dominant 2-D extended Kalman filter, the fully specified 2-D debiased CMKF's algorithm yields improved tracking performance and statistical consistency between the actual state-estimate error and the state-estimate-error covariance. However, they did not quantify the additional specification's impact on the statistical consistency between the debiased converted measurement's error and the average true covariance for the "dynamic case" when both the average true bias and covariance are usually [1] evaluated with the polar prediction—having error statistics significantly

different from those of the polar measurement [3]—during tracking.

The stated reason in [1] for the additional requirement of using the best available polar estimate was that the average true covariance had been shown to be "a function of the [true] target range and bearing as well as the error in their respective measurements." The average true covariance is in fact a function of the polar measurement whose components are respectively given by (1) of [1] as the sum of the true polar quantities and the polar-measurement errors. However, the average true bias and covariance expressions were derived to properly account only for the fact that the polar measurement's error components are zero-mean, Gaussian, and uncorrelated; the expressions do not properly account for the polar prediction's error whose statistics are non-Gaussian and correlated [3]. From another point of view, whereas using the best available polar estimate in a function of the *true* polar position would be a mathematically justified approximation technique, substituting a more accurate polar estimate for the polar measurement of which the average true bias and covariance are functions is not mathematically justified. From either point of view, the only mathematically justified polar estimate to use in the evaluation of the average true bias and covariance is the polar measurement itself. Thus, the stated reason for the debiased CMKF's additional requirement cannot be correct.

This paper provides the correct empirical explanation (rather than a mathematical justification) for the improved tracking performance in the simulated tracking scenarios of [1] obtained by evaluating the average true bias and covariance with a polar estimate less uncertain than the polar measurement. Specifically, we show that evaluating the average true bias and covariance with a polar estimate less uncertain than the polar measurement results in bias and covariance expressions which more closely approximate the ideal bias and covariance than does exclusively evaluating the average true bias and covariance with the polar measurement.

Section 2 provides a concise review of the 2-D debiased CMKF's algorithm. In Section 3 we show three important, empirical performance characteristics resulting from evaluating the average true bias and covariance with polar estimates of varying quality. First, we confirm that, as claimed by [1], the tracking performance improves with the quality of the polar estimate used to evaluate the average true bias and covariance. Second, we demonstrate that statistical *inconsistency* between the debiased converted measurement's error and the average true covariance results when the average true bias and covariance are evaluated with polar estimates having error statistics different from those of the polar measurement's error. Third, we resolve the apparent contradiction between the tracking improvement and the statistical inconsistency which result when the average true bias and covariance are evaluated with polar estimates more accurate than the polar measurement by showing the average true bias and covariance actually become respectively closer, on average, to the true bias and covariance. This third characteristic underlies the superior tracking performance of the debiased CMKF of [1,2] over the debiased CMKF which evaluates the average true bias and covariance exclusively with polar measurement.

2. Technical Background

A sensor remotely measures a target's position and produces the polar position measurement

$$\mathbf{z} = \begin{bmatrix} r_m \\ \theta_m \end{bmatrix} = \begin{bmatrix} r \\ \theta \end{bmatrix} + \begin{bmatrix} \tilde{r}_m \\ \tilde{\theta}_m \end{bmatrix}. \tag{1}$$

In (1), $\begin{bmatrix} r & \theta \end{bmatrix}'$ is the target's true position in polar coordinates (range and bearing), and $\begin{bmatrix} \tilde{r}_m & \tilde{\theta}_m \end{bmatrix}'$ is [1] a white, zero-mean, Gaussian measurement noise with covariance

$$\mathbf{R}_m = \mathrm{cov}\left\{ \begin{bmatrix} \tilde{r}_m & \tilde{\theta}_m \end{bmatrix}' \right\} = \begin{bmatrix} \sigma^2_{\tilde{r}_m} & 0 \\ 0 & \sigma^2_{\tilde{\theta}_m} \end{bmatrix}. \tag{2}$$

Since the target's true Cartesian position is

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} r\cos(\theta) \\ r\sin(\theta) \end{bmatrix}, \tag{3}$$

the error $\begin{bmatrix} \tilde{x}_m & \tilde{y}_m \end{bmatrix}'$ of the raw converted measurement

$$\mathbf{z}_m = \begin{bmatrix} x_m \\ y_m \end{bmatrix} = \begin{bmatrix} r_m\cos(\theta_m) \\ r_m\sin(\theta_m) \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} \tilde{x}_m \\ \tilde{y}_m \end{bmatrix} \tag{4}$$

is

$$\begin{aligned} \tilde{\mathbf{z}}_m &= \begin{bmatrix} \tilde{x}_m \\ \tilde{y}_m \end{bmatrix} = \begin{bmatrix} x_m \\ y_m \end{bmatrix} - \begin{bmatrix} x \\ y \end{bmatrix} \\ &= \begin{bmatrix} r_m\cos(\theta_m) - r\cos(\theta) \\ r_m\sin(\theta_m) - r\sin(\theta) \end{bmatrix}. \end{aligned} \tag{5}$$

Lerro and Bar-Shalom [1] derived a closed-form expression for the raw converted measurement's true error bias

$$\boldsymbol{\mu}_t = E\left(\tilde{\mathbf{z}}_m \middle| r,\theta\right), \tag{6}$$

with which (4) can be debiased to produce an unbiased converted measurement. They also derived a closed-form expression for the corresponding debiased converted measurement's true error covariance

$$\begin{aligned}\mathbf{R}_t &= \operatorname{cov}\left\{\mathbf{z}_m - \boldsymbol{\mu}_t - \begin{bmatrix} x \\ \\ y \end{bmatrix} \middle| r,\theta\right\} \\ &= \operatorname{cov}\left(\tilde{\mathbf{z}}_m - \boldsymbol{\mu}_t \middle| r,\theta\right) \\ &= \operatorname{cov}\left(\tilde{\mathbf{z}}_m \middle| r,\theta\right)\end{aligned} \tag{7}$$

which is necessary for the standard Kalman-filter algorithm. However, since the closed-form expressions for (6) and (7) require the target's *true* range and bearing [1], realizable CMKFs cannot use these expressions.

In response to the impracticality of using (6) and (7), Lerro and Bar-Shalom [1] proposed what is now known as the 2-D debiased CMKF. The 2-D debiased CMKF approximates the raw converted measurement's true error bias with the "average true bias" of the converted-measurement error [1]

$$\boldsymbol{\mu}_a = E\left(\boldsymbol{\mu}_t \middle| \mathbf{z}\right) \tag{8}$$

which requires the polar measurement rather than the target's true position. Thus, the debiased converted measurement that is actually input to the 2-D debiased CMKF's tracking algorithm is

$$\mathbf{z}_m^{CMKF-D} = \mathbf{z}_m - \boldsymbol{\mu}_a . \tag{9}$$

Similarly, the 2-D debiased CMKF approximates the debiased converted measurement's true error covariance with the "average true covariance" of the converted-measurement error [1]

$$\mathbf{R}_a = E\left(\mathbf{R}_t \middle| \mathbf{z}\right) \tag{10}$$

which, like the average true bias, requires the polar measurement rather than the target's true position. Note that the procedure of first conditioning the bias and covariance on the target's true position and then conditioning the resulting expressions' means on the polar measurement is known as "nested conditioning" [4].

As a final specification for the 2-D debiased CMKF, Lerro and Bar-Shalom called for using the best available polar estimate, $\left[\hat{r} \quad \hat{\theta}\right]'$, for the evaluation of $\boldsymbol{\mu}_a$ [2] and $\mathbf{R}_a$ [1,2] during tracking. Lerro and Bar-Shalom identified the polar measurement itself and the polar position prediction (obtained via nonlinear transformation of the CMKF's Cartesian position prediction) as the only such practically available polar estimates. Lerro and Bar-Shalom specified a simple test that, at each processing index, chooses the polar estimate having the error covariance with the smaller "size" in Cartesian coordinates as measured by the matrix determinant. Mathematically, the employed polar estimate, $\left[\hat{r} \quad \hat{\theta}\right]'$, is selected according to

$$\begin{bmatrix} \hat{r} \\ \\ \hat{\theta} \end{bmatrix} = \begin{cases} \mathbf{z}, & \left|\mathbf{HP}[k \mid k-1]\mathbf{H}'\right| \geq \left|\mathbf{R}_a\right| \\ \left[\sqrt{x_p^2 + y_p^2} \quad \tan^{-1}\left(y_p / x_p\right)\right]', & \left|\mathbf{HP}[k \mid k-1]\mathbf{H}'\right| < \left|\mathbf{R}_a\right| \end{cases} \tag{11}$$

where x_p and y_p are the position components of the CMKF's prediction in the *x* and *y* directions, respectively, $\mathbf{H}$ is the matrix which extracts the CMKF's Cartesian position prediction from its Cartesian state prediction, and $\mathbf{P}[k \mid k-1]$ is the predicted state-estimate-error covariance. In (11), $\mathbf{H}[k]\mathbf{P}[k \mid k-1]\mathbf{H}'[k]$ is used as a practically available approximation for the Cartesian position prediction's error covariance.

3. Debiased CMKF Performance Characteristics

This Section presents the results of a three-part investigation of the debiased CMKF's performance in one simulated tracking scenario of [1]. For comparison purposes, we consider the four debiased-CMKF implementations summarized in **Table 1**. Debiased CMKF 1 is the ideal debiased CMKF which respectively uses $\boldsymbol{\mu}_t$ and $\mathbf{R}_t$ as the debiasing and covariance terms. Debiased CMKF 2 is the practical debiased CMKF which respectively uses $\boldsymbol{\mu}_a$ and $\mathbf{R}_a$—both of which are evaluated exclusively with the polar measurement—as the debiasing and covariance terms. Debiased CMKF 3 is the debiased CMKF of [1,2] which employs the test (11) to determine the polar estimate, $\left[\hat{r} \quad \hat{\theta}\right]'$, with which to evaluate $\boldsymbol{\mu}_a$ and $\mathbf{R}_a$ during tracking. Finally, debiased CMKF 4 is the debiased CMKF which exclusively uses the target's true polar position, $\left[r \quad \theta\right]'$, to evaluate $\boldsymbol{\mu}_a$ and $\mathbf{R}_a$ during tracking. Note that only the second and third considered debiased-CMKF implementations are practically realizable since the first and fourth implementations require the target's true position.

In all testing of the four debiased-CMKF implementations summarized in **Table 1**, we use the target-kinematics

Table 1. Considered debiased-CMKF implementations.

Debiased CMKF	Debiasing Term	Measurement-Error Covariance
1	$\boldsymbol{\mu}_t$	$\mathbf{R}_t$
2	$\boldsymbol{\mu}_a\|_{r_m,\theta_m}$	$\mathbf{R}_a\|_{r_m,\theta_m}$
3	$\boldsymbol{\mu}_a\|_{\hat{r},\hat{\theta}}$	$\mathbf{R}_a\|_{\hat{r},\hat{\theta}}$
4	$\boldsymbol{\mu}_a\|_{r,\theta}$	$\mathbf{R}_a\|_{r,\theta}$

model and the second test scenario (*i.e.*, the case where $\sigma_{\tilde{\theta}_m} = 2.5°$) from the "Simulation Results" section of [1] (since the first test scenario yields qualitatively similar results). The presented results are those averaged over 100000 Monte-Carlo runs.

3.1. Tracking Performance

Figure 1 shows the root-mean-squared (RMS) position-tracking errors of the four debiased-CMKF implementations specified in **Table 1**. With the exception of the first few measurement indices, the ideal debiased CMKF 1 yields the best tracking performance as expected. Consistent with the claim of [1], debiased CMKF 3 exhibits tracking performance superior to that of debiased CMKF 2. Debiased CMKF 4, which uses the best possible polar estimate, $\begin{bmatrix} r & \theta \end{bmatrix}'$, to evaluate $\boldsymbol{\mu}_a$ and $\mathbf{R}_a$, nearly matches the performance of the ideal debiased CMKF 1 and in turn outperforms debiased CMKF 3 (after the first few measurement indices). Thus, these results support the tracking-performance claim made in [1] with regard to the use of better polar estimates for the evaluation of $\boldsymbol{\mu}_a$ and $\mathbf{R}_a$.

3.2. Statistical-Consistency Test

As stated earlier, in [1] Lerro and Bar-Shalom demonstrated that, for the static case in which $\boldsymbol{\mu}_a$ and $\mathbf{R}_a$ are evaluated exclusively with the polar measurement, the resulting $\mathbf{R}_a$ is statistically consistent with the actual debiased converted-measurement error for even fairly large bearing-measurement standard deviations (up to 10°). In addition, Lerro and Bar-Shalom showed the state-estimate error to be statistically consistent with the state-estimate-error covariance during tracking when using (11). However, no mention was made about the impact on the statistical consistency between the actual debiased converted-measurement error and $\mathbf{R}_a$ when both $\boldsymbol{\mu}_a$ and $\mathbf{R}_a$ are evaluated with a polar estimate that is better than the polar measurement.

To quantify this effect, we use the statistical-consistency test outlined in Section III of [1] during tracking for the four previously specified debiased-CMKF implementations. **Figure 2** shows the normalized error squared (NES) and the chi-square 0.99 probability bounds. Clearly, debiased CMKF 1 and debiased CMKF 2 both demonstrate the expected statistical consistency between their employed converted-measurement-error covariances and their respective actual debiased converted-measurement errors. Debiased CMKF 3 quickly loses statistical consistency between its $\mathbf{R}_a$ and its actual debiased converted-measurement error. Note that this loss of statistical consistency coincides exactly with the test (11) dictating the predicted polar estimate rather than the polar measurement be used to evaluate $\boldsymbol{\mu}_a$ and $\mathbf{R}_a$. Since $\boldsymbol{\mu}_a$ and $\mathbf{R}_a$ were originally derived from $\boldsymbol{\mu}_t$ and $\mathbf{R}_t$, respectively, by conditioning on the polar measurement, evaluating $\boldsymbol{\mu}_a$ and $\mathbf{R}_a$ with a polar estimate having error statistics significantly different from those of the polar measurement compromises statistical consistency. For this same reason, debiased CMKF 4 *never* exhibits statistical consistency between its $\mathbf{R}_a$ and its actual debiased converted-measurement error.

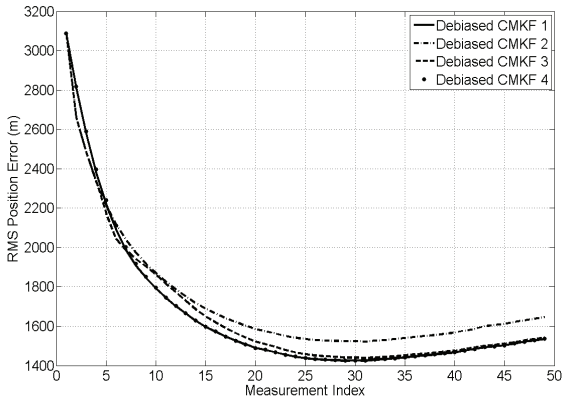


Figure 1. RMS position errors.

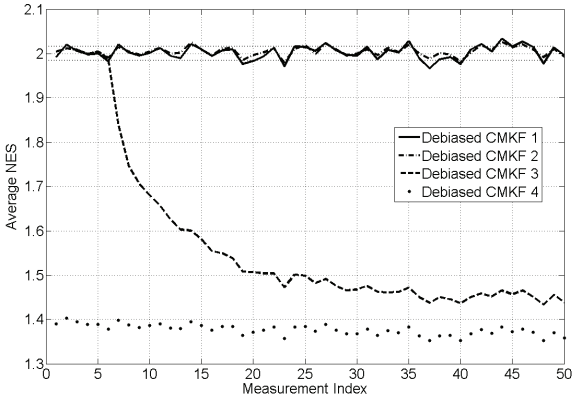


Figure 2. Average NES values.

3.3. Resolving the Apparent Contradiction

The four considered debiased-CMKF implementations of **Table 1** are identical except for the employed debiasing terms and converted-measurement-error covariances. We thus postulate that debiased-CMKF implementations 3 and 4 achieve their superior tracking performance over debiased CMKF 2 because their employed debiasing terms and measurement-error covariances are "closer" to the respective "optimal" values. For the "optimal" debiasing terms and measurement-error covariances, we respectively propose $\boldsymbol{\mu}_t$ and $\mathbf{R}_t$ since using these ideal expressions ultimately yield the best debiased-CMKF tracking performance in simulation. We propose using $E\left\{\left\|\boldsymbol{\mu}_a-\boldsymbol{\mu}_t\right\|_2\right\}$, the average 2-norm distance between each employed debiasing term and $\boldsymbol{\mu}_t$, as the metric for the employed debiasing term's closeness to $\boldsymbol{\mu}_t$. Similarly, we propose using $E\left\{\left\|\mathbf{R}_a-\mathbf{R}_t\right\|_F\right\}$, the average Frobenius-norm distance between each employed measurement-error covariance and $\mathbf{R}_t$, as the metric for the employed measurement-error covariance's closeness to $\mathbf{R}_t$.

Figure 3 shows $E\left\{\left\|\boldsymbol{\mu}_a-\boldsymbol{\mu}_t\right\|_2\right\}$, obtained through simulation using a sample-mean approach, for the three non-ideal debiased-CMKF implementations specified in **Table 1**. While debiased CMKF 3 evaluates $\boldsymbol{\mu}_a$ and $\mathbf{R}_a$ with the polar measurement, its $E\left\{\left\|\boldsymbol{\mu}_a-\boldsymbol{\mu}_t\right\|_2\right\}$ exactly equals the $E\left\{\left\|\boldsymbol{\mu}_a-\boldsymbol{\mu}_t\right\|_2\right\}$ of debiased CMKF 2, as expected. However, as debiased CMKF 3 shifts to evaluating $\boldsymbol{\mu}_a$ and $\mathbf{R}_a$ with the predicted polar estimate, its $E\left\{\left\|\boldsymbol{\mu}_a-\boldsymbol{\mu}_t\right\|_2\right\}$ becomes smaller than the $E\left\{\left\|\boldsymbol{\mu}_a-\boldsymbol{\mu}_t\right\|_2\right\}$ of debiased CMKF 2, indicating the $\boldsymbol{\mu}_a$ of debiased CMKF 3 is closer, on average, to $\boldsymbol{\mu}_t$ than is the $\boldsymbol{\mu}_a$ of debiased CMKF 2. By evaluating $\boldsymbol{\mu}_a$ and $\mathbf{R}_a$ with the true position, debiased CMKF 4 achieves the smallest $E\left\{\left\|\boldsymbol{\mu}_a-\boldsymbol{\mu}_t\right\|_2\right\}$, indicating its debiasing term is the closest, on average, to $\boldsymbol{\mu}_t$ of all the considered non-ideal debiased-CMKF implementations.

Figure 4 shows $E\left\{\left\|\mathbf{R}_a-\mathbf{R}_t\right\|_F\right\}$, obtained through simulation using a sample-mean approach, for the three non-ideal debiased-CMKF implementations specified in **Table 1**. While debiased CMKF 3 evaluates $\boldsymbol{\mu}_a$ and $\mathbf{R}_a$ with the polar measurement, its $E\left\{\left\|\mathbf{R}_a-\mathbf{R}_t\right\|_F\right\}$ exactly equals the $E\left\{\left\|\mathbf{R}_a-\mathbf{R}_t\right\|_F\right\}$ of debiased CMKF 2, as expected. However, as debiased CMKF 3 shifts to evaluating $\boldsymbol{\mu}_a$ and $\mathbf{R}_a$ with the predicted polar estimate, its $E\left\{\left\|\mathbf{R}_a-\mathbf{R}_t\right\|_F\right\}$ becomes smaller than the $E\left\{\left\|\mathbf{R}_a-\mathbf{R}_t\right\|_F\right\}$ of debiased CMKF 2, indicating the $\mathbf{R}_a$ of debiased CMKF 3 is closer, on average, to $\mathbf{R}_t$ than is the $\mathbf{R}_a$ of debiased CMKF 2. By evaluating $\boldsymbol{\mu}_a$ and $\mathbf{R}_a$ with the true polar position, debiased CMKF 4 achieves the smallest $E\left\{\left\|\mathbf{R}_a-\mathbf{R}_t\right\|_F\right\}$, indicating its measurement-error covariance is the closest, on average, to $\mathbf{R}_t$ of all the considered non-ideal debiased-CMKF implementations.

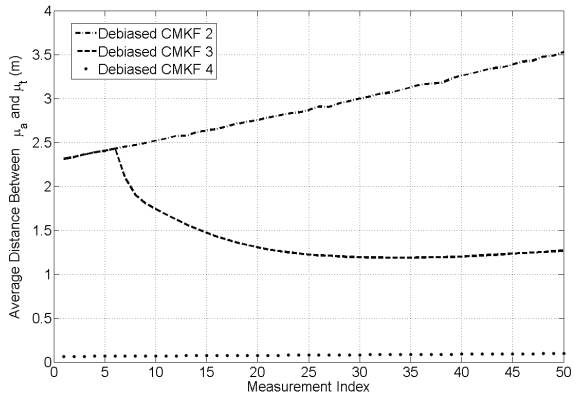


Figure 3. Average distance between the employed $\boldsymbol{\mu}_a$ and $\boldsymbol{\mu}_t$.

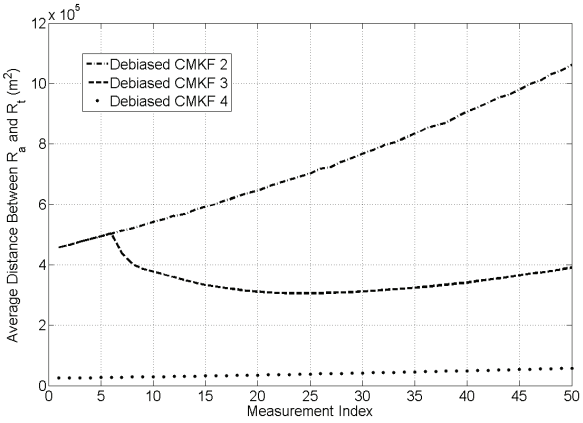


Figure 4. Average distance between the employed $\mathbf{R}_a$ and $\mathbf{R}_t$.

From **Figures 3** and **4** it is apparent that the debiased-CMKF implementations which evaluate $\boldsymbol{\mu}_a$ and $\mathbf{R}_a$ with polar estimates less uncertain than the polar measurements have both debiasing terms and converted-measurement-error covariances which are closer to the ideal $\boldsymbol{\mu}_t$ and $\mathbf{R}_t$, respectively.

4. Acknowledgements

J. N. Spitzmiller thanks Dr. D. Lerro for graciously making his computer programs available for review.

5. References

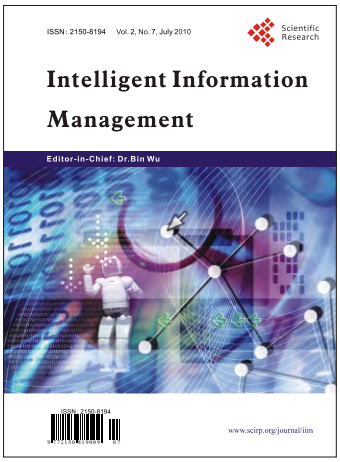
[1] D. Lerro and Y. Bar-Shalom, "Tracking with Debiased Consistent Converted Measurements Versus EKF," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 29, No. 3, 1993, pp. 1015-1022.

[2] D. T. Lerro, Alion Science and Technology, Private Communication, Mystic, CT, June 2008.

[3] D. T. Lerro, Alion Science and Technology, Private Communication, Mystic, CT, October 2008.

[4] X. R. Li and V. P. Jilkov, "A Survey of Maneuvering Target Tracking—Part III: Measurement Models," *Proceedings of SPIE Conference on Signal and Data Processing of Small Targets*, San Diego, CA, USA, 2001, pp. 423-446.

Call for Papers



Intelligent Information Management

ISSN 2150-8194 (print) ISSN 2150-8208 (online)
www.scirp.org/journal/iim

IIM is a peer reviewed international journal dedicated to the latest advancement of intelligent information management. The goal of this journal is to keep a record of the state-of-the-art research and promote the research work in these fast moving areas. The journal publishes the highest quality, original papers includ but are not limited to the fields:

- ★ Computational intelligence
- ★ Data mining
- ★ Database management
- ★ Distributed artificial intelligence
- ★ General systems theory and methodology
- ★ Information security
- ★ Intelligent information management systems
- ★ Knowledge discovery and management
- ★ Nonlinear system and control theory
- ★ Optimal algorithms
- ★ Other related areas and applications

We are also interested in short papers (letters) that clearly address a specific problem, and short survey or position papers that sketch the results or problems on a specific topic. Authors of selected short papers would be invited to write a regular paper on the same topic for future issues of the IIM.

Website and E-Mail

http://www.scirp.org/journal/iim **E-Mail:iim@scirp.org**

Intelligent Information Management, **2010, 2, 391-436.**
Published Online July 2010 in SciRes (http://www.SciRP.org/journal/iim/).

TABLE OF CONTENTS

Volume 2 Number 7 **July 2010**

