2004

Multi-event crisis management using non-cooperative repeated games

Upavan Gupta
University of South Florida

Multi-Event Crisis Management Using Non-Cooperative Repeated Games

by

Upavan Gupta

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science in Computer Science
Department of Computer Science and Engineering
College of Engineering
University of South Florida

Major Professor: N. Ranganathan, Ph.D.
Dewey Rundus, Ph.D.
Srinivas Katkoori, Ph.D.

Date of Approval:
November 19, 2004

Keywords: game theory, nash equilibrium, emergency response, multi-player games, resource optimization

DEDICATION

To My Loving Parents

ACKNOWLEDGEMENTS

I would like to express gratitude to my major professor, Dr. N. Ranganathan, for his encouragement, guidance, support and friendship throughout my Master's Program. Without his patience and valuable suggestions, this work would not have been completed. I would like to thank Dr. Dewey Rundus and Dr. Srinivas Katkoori for guiding me and being so helpful all the time.

I would like to thank Narender Hanchate for his valuable ideas and help throughout the thesis work. I wish to thank Rashmi Shetty for the useful discussions relevant to thesis. I would also like to thank Ashok Murugavel for helping me understand the concepts of Game theory and Auctions.

I really appreciate the invaluable support I received from my parents and my sister. The support and help provided by my friends was great and was substantial in the completion of the work. I would really like to acknowledge and appreciate it.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

MULTI-EVENT CRISIS MANAGEMENT USING NON-COOPERATIVE REPEATED GAMES

Upavan Gupta

ABSTRACT

The optimal allocation of the resources to the emergency locations in the event of multiple crises in an urban environment is an intricate problem, especially when the available resources are limited. In such a scenario, it is important to allocate emergency response units in a fair manner based on the criticality of the crisis events and their requests. In this research, a crisis management tool is developed which incorporates a resource allocation algorithm. The problem is formulated as a game theoretic framework in which the crisis events are modeled as the players, the emergency response centers as the resource locations with emergency units to be scheduled and the possible allocations as strategies. The pay-off is modeled as a function of the criticality of the event and the anticipated response times. The game is played assuming a specific region within a certain locality of the crisis event to derive an optimal allocation. If a solution is not feasible, the perimeter of the locality in consideration is increased and the game is repeated until convergence. Experimental results are presented to illustrate the efficacy of the proposed methodology and metrics are derived to quantify the fairness of the solution. A regression analysis has been performed to identify the statistical significance of the results.

CHAPTER 1

INTRODUCTION

Crisis Management is the most intriguing issue of the twenty-first century. Sensing a crisis event, which could encompass a wide spectrum of conditions, ranging from but not limited to a road accident, fire, plane crash, natural disaster like hurricane and tornado, willful act of destruction, economical crisis, terrorist attack, nuclear emergency and security breach of any kind and responding to it in order to mitigate its effects is considered as crisis management. An efficient crisis management system is of paramount importance, because if not responded to promptly and managed properly, even a small mishap could lead to a very big catastrophe with significantly severe consequences.

1.1 Crisis Management

Being equipped with a profusion of resources does not ensure a successful recovery from a crisis situation. The key to the successful recovery necessitates an effective and expedited allocation of the requested resources to the emergency locations. Each crisis situation requires some minimum number of units of certain types of resource to be allocated to it. The type and the number of resource units to be allocated depend upon the criticality, severity and the nature of incident. Additionally, depending upon the location of incident, a crisis event may request extra units of some resource types which could potentially be required, so that an already existing crisis situation does not spawn another crisis condition. For example, in a hypothetical scenario, a crisis situation due to a road accident involving collision between two cars, essentially requiring only a few police cars, a few ambulances, and a towing truck, may request for some additional police

cars in order to manage and redirect the traffic because failure in doing so would result in increased traffic congestion and could potentially give rise to another crisis situation.

Crisis management is not restricted to sense and response. It comprises of a complete life cycle of activities which are carried out starting from definition of the crisis conditions, to actual identification of such conditions in a scenario, followed by recovery from them by allocation and scheduling of required resources to the emergency locations in an optimal manner and most importantly, learning from those situations, setting up new standards and laying down plans for a better recovery and mitigation in future [1,2]. Before understanding the crisis management life cycle, we need to know about two types of crisis conditions that could exist in a system.

1.1.1 Types of Crisis Situations

At any point of time, in a particular area, there may exist one of the following two conditions.

1.1.1.1 Single Crisis

If only one crisis event has occurred in a particular locality (lets assume a county), and it requires some resources to alleviate the situation, then such an event is easier to handle because all the requirements of the crisis can be satisfied from the resources available in that locality or can be made available from nearby locations. All the expendable resources available in the area are exclusively reserved for the crisis and will be allocated as requested. Hence, there will be no competition for the allocation of resources to that crisis situation.

1.1.1.2 Multiple Crises

When more than one crisis event occurs in a locality, all the crisis events will need resources of different types in different quantities, to be allocated to them in order to recover from those situations. In addition to some distinct types of resources, some resources of same type may be required by a few of those crises. Since there is a limited supply of those resources in the locality, such a situation would lead to a competition among the different crises events, where

each of those events would try to acquire the required resource units, at a minimum *cost.* The cost in the scenario would be a function of a number of factors including the criticality of the crisis situation, the distance between the crisis location and the resource center, the average allowable speed on the route between crisis and resource, and the parameters related to the time of the occurrence of the event. These attributes contribute towards the cost in terms of total response time for getting a resource, which a crisis tries to minimize. The optimal allocation of resources to the crises in a multiple crises scenario where crises are competing for receiving the required number of resource units from limited supply is a challenging and more intricate issue.

1.2 Crisis Management Life Cycle

The important elements of the complete life cycle of a crisis [1] are shown in Figure 1.1.

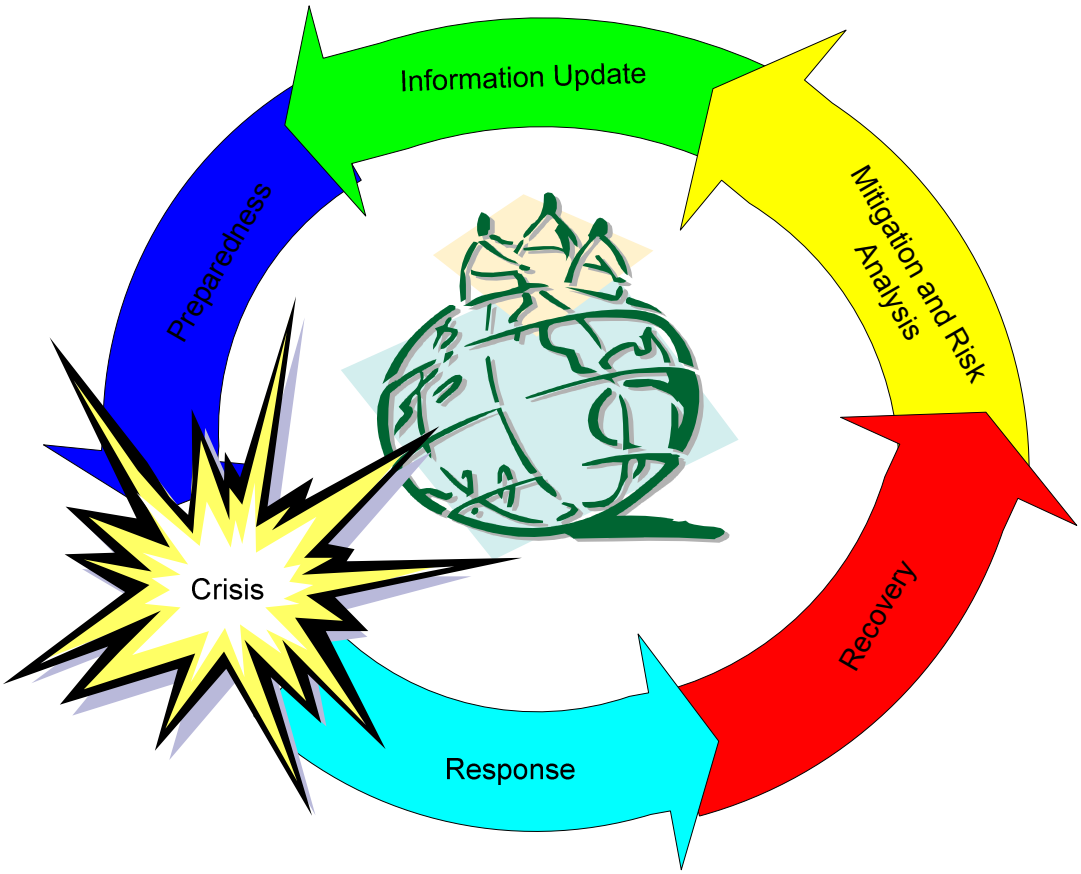

Figure 1.1 Crisis Management Life Cycle [1]

As soon as a crisis condition surfaces, a series of actions are required in order to manage that crisis.

1.2.1 Response

A swift and appropriate response to an event is possible only if a condition is sensed as a potential crisis. In order to diagnose such a condition, the crisis management system should have relevant information to detect if a critical threshold has been reached, leading to an emergency [3,4]. Response involves the allocation and mobilization of the required resource units and emergency equipment to the crisis locations. In a multi-event crisis situation, an optimal allocation of those units is of paramount importance.

1.2.2 Recovery

After a prompt response to the emergency, an apposite recovery is essential. Recovery includes discarding the wreckage, taking care of the people and the commodities affected by the crisis, resurrection of the infrastructure and a lot many activities to assuage the effects of crisis. Loan and grants may be required for the recovery purposes and if the crisis is big, federal assistance could be required.

1.2.3 Mitigation and Risk Analysis

Mitigation is a very critical step in the emergency response life cycle. Though it appears in the third phase of the crisis life cycle, it essentially is a continuous process with intent of improving people's life. This is done by improving the infrastructure and taking better precautions. This reduces the impact of disasters, including natural disasters like hurricanes and floods, on lives of people and communities.

Risk analysis and reduction is essentially the goal of mitigation process. Better and successful mitigation from emergencies is possible if the risk analysis for those emergencies is apt. This in turn leads to more effective risk reduction.

1.2.4 Information Update

Whenever a crisis event occurs, it uncovers a lot of issues that were not addressed earlier and hence led to the crisis. Such an event also provides a lot of information for dealing with such conditions in future. Type and number of resource units required, maximum acceptable response time, and potential crisis events associated with the emergency are a few examples of the information we can get from the crises. All this information needs to be updated in the system's database. This information would be useful for the future endeavors of dealing with similar situations.

1.2.5 Preparedness

Preparedness ensures that if the same type of disaster strikes in future, people and response agencies are better prepared for it and the loss due to disaster is minimized. A better preparedness ensures a swift and effective recovery from the disaster. A good risk analysis, updated information and proper mitigation process attributes towards a better preparedness and hence a successful recovery. These attributes tries to prevent the crises from occurring in future, and if it can not be prevented, tries to subdue its effects at minimum cost.

1.3 Multi-Event Crisis Management

As mentioned earlier, a multi-event crisis scenario consists of simultaneous occurrence of more than one crisis event. Managing multiple crisis events is an intricate problem, because of the existence of competition among the crises for allocation of resources that are available in a very limited quantity. In such a situation, the allocation of resources to the crises should such that it

satisfies the requirements of all the crisis events within an acceptable time frame. The problem could be better explained with the aid of an example scenario described in Figure 1.2.

At some instance of time, in a particular locality, the following crisis events have occurred.

- At the airport situated on south west of a county, a small size cargo air-plane has crashed. While landing, the pilot lost the control over the plane and it ended up hitting the trees near the runway. There were two people in plane, a pilot and a co-pilot.
- Around the same time, somewhere in the northern part of the county, a house caught fire because of some leakage in the gas pipe. No one was at home when the fire started.
- During the same time frame, at the intersection of Fowler Ave. and Bruce B. Downs, which are localities in Tampa, a car coming from the wrong direction collided with a small truck, thereby causing some major injuries to the drivers and passengers.
- A football game was happening at the Raymond James stadium between a local and a visiting team at the same time. During the game, some of the decisions taken by the referee got into controversy and the local team supporters got agitated, which very soon resulted in a fight between supporters of the two teams.

Now, each of these crisis events needs some resources to be allocated to them in order to recover from those conditions. Let us assume that the plane crash would need around 3 police cars, two fire engines and a couple of ambulances, while the house fire would just require a police car for supervision and 4 to 5 fire engines. The road accident would require around 5 to 6 police cars to look over the accident and re-route and manage the traffic. Also, it needs at least 3 ambulances to quickly send the injured people to hospital and a toeing truck to move the vehicles and clear the traffic. The fight at the stadium would require 7 to 8 police cars and a few ambulances to handle the situation. Additionally, each of the crisis events has a priority associated with it, which depends upon the criticality of the event. In the present scenario, the

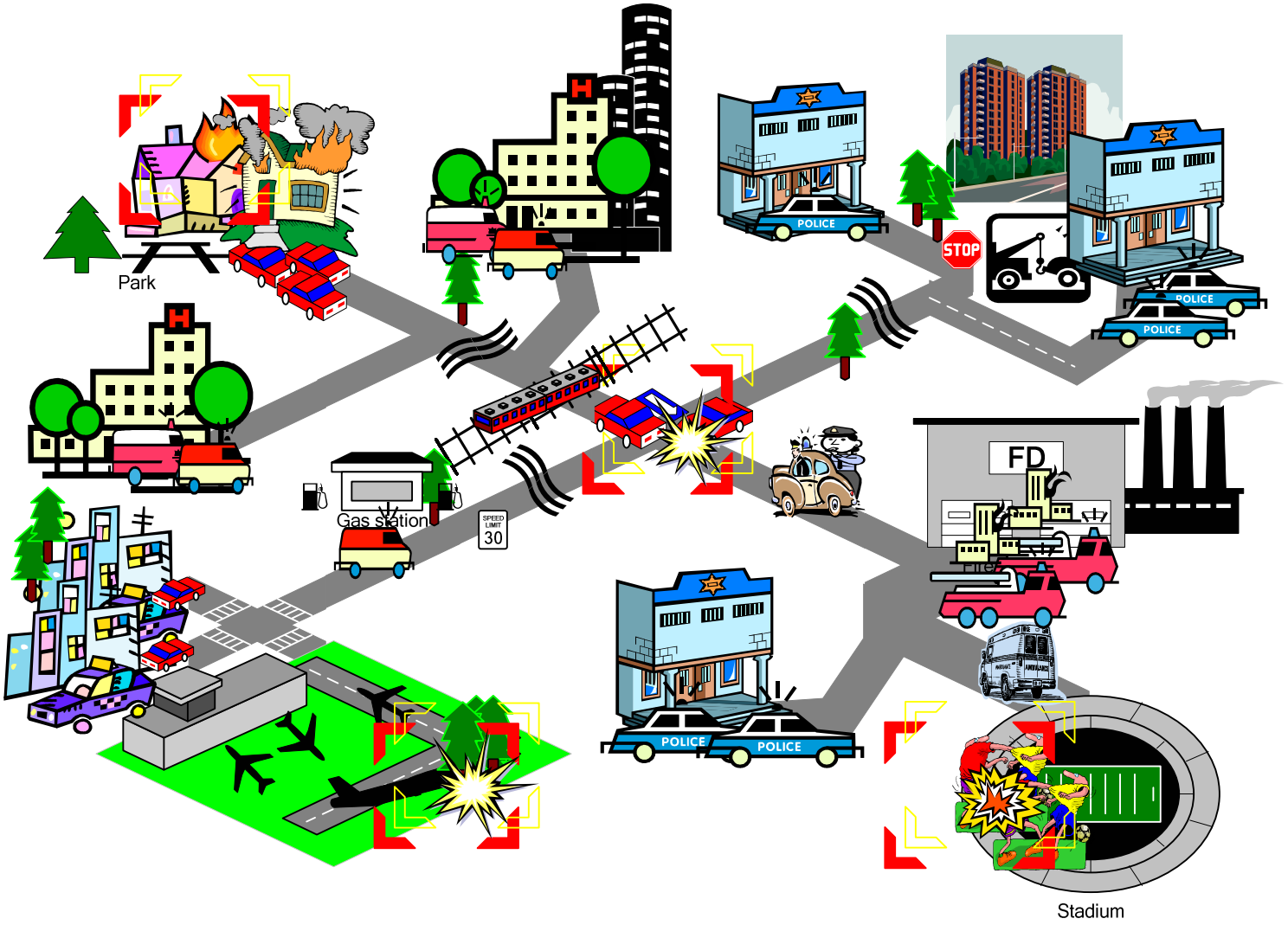

Figure 1.2 Multi-Event Crisis Scenario

most critical incidence is the road accident followed by the plane crash. The fire in building and the fight could be kept at the same priority level, less critical than the crash.

The availability of the resources is shown in Figure 1.2. There are three police stations (blue building with a star on it), two hospitals (H), one towing agency (next to one of the police stations) and one fire department (FD) in the locality. Since only one incident (road accident) needs the assistance of towing agency, there will be no competition for that particular type of resource. Among the other resource types, police cars are required at all the crisis event locations, ambulances are required for handling the road accident, the fight and the plane crash and fire engines for the house fire and the plane crash. Now, as we see here, there is a limited supply of these resources in the locality and almost all the crises events require some units of each of these resources for recuperating from crisis condition. Such a situation demands a sense and response system, which could facilitate an optimal allocation of resources to the emergencies.

In a distributed environment, the system would monitor all the events and if a critical condition arises, it would respond to it in a proactive manner [3,4,5]. The system would ensure that the following issues are addressed in the most optimal way.

- Each crisis location receives the requested service (resources) at an optimal cost
- The total cost for servicing the crises in the system (environment) is optimal

1.4 Optimization

A mechanism of allocation of resources satisfying the requirements of each crisis in such a way that the overall cost of allocation of resources for the complete scenario is optimal, would be considered as an optimal allocation. In such an allocation, the total cost of allocation of resources for the complete system would be optimal and would satisfy the requirements of individual crisis events in the best possible manner [30,31].

Optimization protocols or mechanisms can be evaluated on the basis of different criteria including knowledge, computational complexity, quality of solution [6], individual rationality,

stability, pareto efficiency and distribution and communication efficiency [7]. Different optimization algorithms address these issues with different priorities. A few optimization algorithms include simulated annealing [8,9,10,11], tabu search [12,13,14], hill climbing method [6,15], genetic algorithm [10,14,16,17,18,19], greedy search [20], Bayesian optimization[21], and random walk. All optimization algorithms generate results which are optimal for some of the criteria listed above, while providing non optimal results for other criteria. Some of these optimization algorithms are unusable for some specific cases, like the one in which a player has incomplete information about other players. A detailed comparative study of various optimization algorithms has been carried out in the following chapter.

For the multi-event crisis management scenario, the crisis events require a methodology of allocation of resources that is optimal on almost all the criteria listed above, but is specifically optimal in terms of quality of solution and rationality.

1.5 Motivation for this Thesis Work

Multi-Event crisis management is an issue of supreme importance which needs to be addressed and dealt with in the best possible manner. Many agencies like FEMA [1,2] and projects like Infospheres [3,4,5] are dealing with this issue of sensing the crises and responding to them. Most of the work has been carried out in the direction of identifying the crises conditions, prioritizing them according the criticality, identifying the types of emergency response units each type of crisis condition would require, alerting the response units when a system enters a critical condition, preparing the emergency managers for such emergency and disaster conditions and responding to the emergency conditions by positioning the response units to crises location. After recovering from the emergency situations, the next important task is to learn from those emergencies and update the criticality parameters accordingly. Laying down some new ground rules for preventing such conditions in future or to reduce the risk of loss in such situations have also been addressed by these projects. Despite all this, none of the projects deal with the issue of

providing a definitive method for allocation and scheduling of the response units to the crises locations so that they receive those resources in an optimal time.

The work being carried out by these projects would be more effective if the emergency locations in a multi-crisis scenario, receive the resource units in a prompt manner. This is the motivation to develop an automated computer based solution for allocating the resources to the crises in a multi-crisis scenario. A solution system which could take the resource requirements of crises as input and could implement a resource allocation optimization algorithm on those inputs to come up with a proposed allocation of resources to each crisis as an output, would be really helpful for the agencies like FEMA in realizing their goals.

In a multi-crisis scenario, where each crisis location requires a few units of different types of resources to be allocated to it and the availability of resources is very limited, there is a competition among the crises, for the allocation of resources. In order to allocate the resources to the crises location in an optimal manner, the system inputs the resource requests by the crises and using a resource allocation optimization algorithm, outputs an allocation which could satisfy each crisis location's request in an optimal manner.

In the proposed application, the scenario is modeled as a normal form non-cooperative game [23,24,25] where the crises events are considered as players of the game, the emergency response locations are considered as the resource locations and the cost of getting the units from a resource location is modeled as a function of different attributes including distance between source and destination, average allowable speed on the route between them, the time at which crisis occurred and the criticality of the crisis event. Each Crisis location, based on the cost and the requirements, has an associated payoff for getting the resource units from a resource location. Each crisis location has a set of strategies, which essentially is a set of different combinations of the resource locations from where it could receive the required units. These strategy sets are given as an input to the game, which employs an optimization algorithm and outputs a strategy combination, one strategy for each player, satisfying the requirements of players in an optimum

manner. *Nash Equilibrium* algorithm [24,26,27,47] is used as the optimization algorithm for allocation.

According to the Nash Equilibrium allocation, each player receives the best possible utility, taking into consideration that the other players adhere to the allocation made to them. Nash Equilibrium solution is considered to be the most socially optimal solution which ensures a rational behavior of players. The algorithm generates a good quality optimal solution. Though the computational complexity of the algorithm is high [29], we have applied some heuristics in order to speed up the convergence to the equilibrium solution.

1.6 Organization of Thesis

The rest of the document is organized in five chapters. In chapter 2, we will briefly discuss about the prior work being done in the field of crisis management and optimization algorithms. We will review some of specific issues being addressed in the projects referring crisis management. Also, we will describe some of the optimization algorithms which are prominently used in the field of resource allocation and scheduling, and will qualify the aptness of using game theory for our system. The chapter will also give a short introduction of Game theory, including the various classifications of games. Chapter 3 will describe the actual formulation of the game, including the problem statement, the various notations used and the objective function. A hypothetical scenario will also be explained as an example.

The actual algorithm of the game and Nash Equilibrium solution methodology will be discussed in chapter 4. In chapter 5, we will present the experimental results which will be generated by running the algorithm on different sets of inputs, using different values for the various parameters. The chapter will have the comparative study and results generated by our algorithm in comparison with some other implementations of the problem. We will also talk about the computational complexity and space complexity of the game and will be compared against those of other algorithms. Finally, in chapter 6 we will talk about the conclusions drawn

from implementing the system and the results, and will give an overview of the future work we intend to pursue in this field.

CHAPTER 2

BACKGROUND AND RELATED WORK

Homeland security being the most compelling concern of every country, demands for a comprehensive and efficient system to manage and recover from any kind of emergency situation [49]. The events encompass every possible adversity ranging from an illegal network intrusion to a major terrorist attack. A significant amount of research work has been performed as a part of academic as well as federal and corporate projects to address the subject of multi-event crisis management and resource scheduling.

In this chapter, some of key projects that have been implemented in the field of crisis management and resource allocation will be reviewed and the different domains of crisis management being addressed by these projects will be addressed. Then, the various resource allocation methodologies available in literature will be studied. Each of these methodologies achieves an optimal solution, and possesses some special characteristics which differentiate it from other algorithms. Next, a concise introduction to *Game Theory* and the different classifications of games will be presented. The chapter will conclude with a qualification for the choice of implementing game theory and Nash Equilibrium algorithm as the resource allocation mechanism for current research. This will be accomplished by performing a comparative study between the different methodologies including Nash Equilibrium algorithm, on the basis of different parameters.

2.1 Multi-Event Crisis Management Projects

The field of crisis management incorporates a wide range of events and different projects explore different domains of it. In this section, some projects which specifically concentrate on emergency management and disaster recovery will be addressed. The discussion will include the prominent issues which these projects address.

2.1.1 FEMA

Federal Emergency Management Agency (FEMA) is the largest unified crisis management organization [1]. It is an integral part of United States of America's Department of Homeland Security's Emergency Preparedness and Response Directorate. The headquarters of FEMA is situated in Washington D.C. Along with 2,600 full time employees it has nearly 4,000 standby disaster assistance employees who are available for deployment after disasters. FEMA works in partnership with other federal, state and local emergency management agencies to accomplish its goals of efficient crisis management.

FEMA covers a broad spectrum of activities that includes advising people about the disasters and teaching them get through such conditions, helping equip state and local agencies in emergency preparedness, and responding to the disasters like hurricanes, floods, volcanic eruptions, fires, nuclear emergencies and terrorist attacks. Mitigating the effects of disasters on people, learning from the disasters and working towards a better preparedness for handling them in future are some of the most important tasks performed by FEMA. Effectively, FEMA performs a series of activities during every stage of the life cycle of a crisis. The life cycle has been described in figure 1.1 and a detailed strategic plan of FEMA for the fiscal years 2003-2008 is described in [2].

FEMA's perspective of emergency management does not incorporate the actual physical allocation of the resources to the emergency locations. These activities are administered by the local and state agencies, which manually allocate the required resources to the crisis locations.

The agencies do not use any automated tool that could provide a solution suggesting an optimal allocation of resources to all of the crisis locations.

2.1.2 Infospheres

A research group at the Department of Computer Science at California Institute of Technology has conducted a project called *Infospheres* [3], which concentrates on the development of compositional systems. Compositional systems are the systems built from interacting components. The project is concerned with the theory and the implementation of compositional systems that support peer-to-peer communication among persistent multithreaded distributed objects. The system finds one of its applications in the field of crisis management.

Infospheres system holds a repository of data from multiple sources and normalizes it to a standard vocabulary, proactively monitors all the components in a distributed system and determines if some critical condition emerges. When such a condition holds, the system responds to it by generating an alert to all the concerned response objects which will be required to manage that crisis [4,5]. The system is essentially a distributed networked alert system.

Confined as an alert system, *Infospheres* does not perform any actual allocation of those response units to the crisis locations. But, the system could prove to be a perfect infrastructure for the crisis management system being proposed by us since the system is equipped with all the sensors and network monitoring paraphernalia that are key requisites for our system.

2.1.3 British Columbia Emergency Response Management System

BCERMS is an emergency response program being developed by the province of British Columbia, Canada [6]. This is a comprehensive emergency management and disaster relief system, which covers almost all the aspects of emergency management. The project defines the objectives of the system, identifies the crises, analyses the requirements of each type of crisis

event, allocates the required resources to the crisis locations according to the requests, addresses the rehabilitation issues, performs a good risk analysis and documents the experiences.

The system incorporates all the important issues of crisis management and performs each of them efficiently. But, it allocates the resources to multiple crises events manually, which does not result in an optimal allocation of resources. An automated emergency response system which could allocate the resources in an optimal manner would be the logical solution to this problem.

2.1.4 Other Relevant Projects

A few more emergency management projects have been implemented at different places. Some of the projects are MEMA (Massachusetts Emergency Management Agency), IEMA (Illinois Emergency Management Agency), GEMA (Georgia Emergency Management Agency) and Police Department Emergency Response Management Plan at University of California at San Francisco. All of these projects explore more or less the same domain of crisis management as FEMA and address similar issues. None of these projects refer to the actual allocation of the resources to crises.

2.2 Crisis Management Tools

A multi-event crisis management system requires an efficient and capable toolset for successful recovery and mitigation from emergencies. With latest advances in the field of telecommunications and information systems, crisis management personnel can monitor the proceedings of a locality in real time. All the crisis management projects that have been discussed in the previous section use one or more of these tools. Some of the most appropriate tools have been briefly described here.

- *Sensors and Actuators:* Different types of sensors including the sensors to find speed of vehicles, sense the existence of vital signs at an emergency location and sensing fire in a building are available. These sensors are of primary utility for the emergency

management systems. The sensors generate event streams when any critical parameter changes are sensed. These event streams are consumed by the actuators which try to either tone down the conditions which caused those critical changes or send an alert to the relevant response locations [4,5].

- *Geographical Information System (GIS):* GIS is an organized collection of computer hardware and software designed to efficiently create, manipulate, analyze and display all types of geographically or spatially referenced data. This system allows complex spatial operations that are very difficult to do otherwise. A GIS system can be specifically developed for the purpose of crisis management by encoding the potential crisis locations, storing all the data related to those locations for the purpose of analysis and outputting the required service if the crises occur [50,51].

  A GIS receiving data and information from sensors and actuators can act as foundation for the emergency management system.

- *Mobile Devices:* A lot of mobile devices are also required for realizing the emergency management system; Global Positioning System (GPS) [53], Automated Vehicle Tracking System, Cellular Phone, Networked alert System, Personal Digital Assistant (PDA), and Intelligent Personal Assistant (iPA) to name a few [52]. These devices are required at the emergency locations for sending relevant data regarding resource requirements to the GIS system [53].

- *Communication Network:* An emergency management system can achieve its goals only if a good network system is available for communication. Sensor and Ad-hoc networks are the most appropriate networking technologies for an emergency management system. These are self reconfigurable networks which adapt themselves to the scenario and to the number of available system resources. Protocols have been developed for the mobile devices to establish connections to these networks.

2.3 Resource Allocation Algorithms

Resources allocation is an optimization problem with the objective of minimizing the cost of allocation while satisfying the constraints being posed by the system. Several algorithmic approaches have been proposed in the literature to achieve the optimal allocation of resources. Although most of these algorithms achieve globally optimal solution, it's the unique properties exhibited by each algorithm that differentiates it from others and makes it more feasible for certain set of applications. This section discusses some of the widely used algorithms to solve the resource allocation problem.

2.3.1 Genetic Algorithms

Genetic algorithms are a class of evolutionary algorithms for resource allocation and scheduling that works on the principles of genetics. A genetic algorithm uses the techniques of recombination, mutation, inheritance and natural selection to evaluate an optimal solution. This is a non-deterministic algorithm which works on the law of survival of the fittest [11, 12, 15,16].

In genetic algorithms, the problem domain is formulated as the population of chromosomes, where each chromosome is a string of bits representing one of the possible solutions of the problem. Each chromosome has a "fitness" value associated with it. This fitness function is the objective function. The chromosomes follow the evolutionary steps of recombination, mutation and inheritance for many generations, giving rise to a new set of chromosomes at each generation, to come up with a promising, optimal solution. The fitness of a chromosome determines its ability to survive and reproduce. During this process, the unfit chromosomes are deleted from the population and are replaced by the more fit chromosomes [19,20].

The non-deterministic nature of genetic algorithms is attributed to the fact that it is not possible to predict the number of recombination that would be required before it converges to a solution. Hence, the algorithm searches the complete solution space in an undirected manner. If

the chromosomes only recombine and inherit at each generation, they may lead to a solution which is locally optimal. The process of mutation ensures that algorithm searches the entire solution space and generates a globally optimal solution [21,22,23]. A pictorial representation of local versus global optimization is shown in Figure 2.1. Genetic algorithms are very useful for the cases when we have very limited or almost no knowledge of the solution space.

In a genetic algorithm the search space never shrinks, which makes it non-deterministic and hence infeasible for a lot of scenario where an optimal result is expected within a time limit. This property makes the methodology computationally expensive. Also, since the algorithm works on principles of natural selection, it is very susceptible to some of the dominant members of population, which have survived for multiple generations and has a high fitness.

2.3.2 Hill Climbing Algorithm

Hill climbing algorithms are a class of algorithms based on the concept of local search. These algorithms tend to move towards a solution point which has a lower cost than the previous solution point, which generates a local optimal solution rather than a global one [18,23]. This would be a suboptimal solution. By accepting some inferior solution points the algorithm could be modified and directed towards globally optimal solution. Some of the algorithms which implement these modifications are simulated annealing and tabu search and are discussed in this subsection.

2.3.2.1 Simulated Annealing

Simulated annealing is an optimization technique which simulates the actual process of annealing. Annealing is a process of cooling solids in a heat bath. When a solid is heated past the melting point and then cooled down, the structural properties of the solid depend upon the rate of cooling. If the liquid is cooled at a slow rate, perfect crystals will form. However, if cooled quickly, the crystals will be imperfect. The algorithm simulates this process, by starting from a

valid solution point and randomly generating new states, and calculating the associated cost functions [9,10].

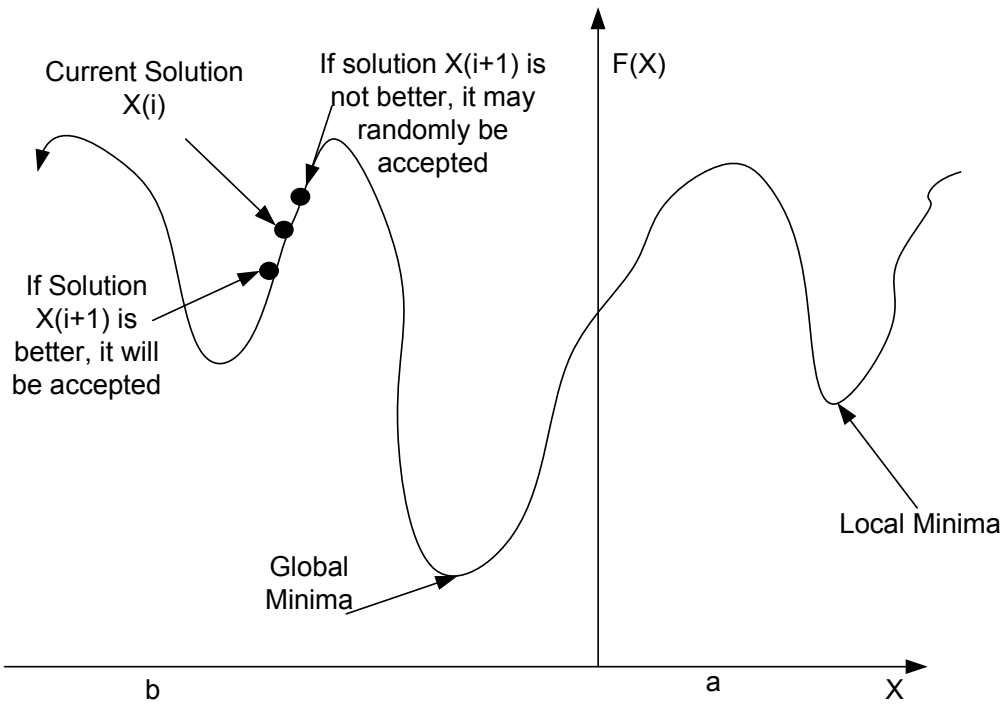

Figure 2.1 Global vs. Local Minima and Simulated Annealing

If the cost of the new state is lower than the cost of the previous state, i.e. if $\Delta(current\_cost - previous\_cost) \leq 0$, it is accepted and the algorithm moves to the newer state. This process would lead the system towards optimum solution, but this solution could easily be a local minima. In order to overcome this problem, the algorithm, with a certain probability, accept a state whose cost is more than the previous state. The probability depends upon the temperature of the system. If the temperature is high, the probability of accepting a higher cost state is high and as the temperature goes down, the probability of acceptance of higher cost stages decrease. An explanatory picture of simulated annealing process is shown in Figure 2.1.

One of the major drawbacks of simulated annealing is the computation time of the algorithm [11,24]. The algorithm's computational complexity is NP complete, because it is a search technique. Also, the algorithm does not generate a solution that is socially optimal. The algorithm is non-deterministic in nature.

2.3.2.2 Tabu Search

Tabu search represents a set of algorithms which have descended from the local improvement techniques. Technically, the basic algorithm is of deterministic nature, but the variations of the algorithm which are used in most of the implementations, are classified as stochastic because of their properties and behavior.

A tabu search algorithm randomly explores the complete solution space on the basis of local searches with controlled up-hill moves. A tabu list is maintained to prevent the algorithm from choosing the same solution sequence and falling into a locally optimal space. Starting from an initial point, the algorithm searches the approximate optimum of the solution space by moving from the current solution to the next solution in the neighborhood of the current solution. The solution points listed in the tabu list are not considered as the candidates of the next solution. The next solution in the search space is represented as

$$N(s_k(x)) = Min\ (N(s(x))):\ s \in S(x\text{-}T) \tag{2.1}$$

where;

x – Current Solution

$s(x)$ – Candidate of next solution

T – Set of solution points in tabu list

$N(x)$ – Objective function

$s(x)$ – Set of neighborhood of x

$s_k(x)$ – Selected k^{th} move

If $s_k(x)$ is selected as the k^{th} move, then the inverse move s_k^{-1} is memorized as the tabu move in the tabu list. A solution point remains in the tabu list only for a limited number of iterations, after which it is removed from the list and could be selected as probable outcomes in future. This algorithm does not guarantee an optimal solution, but provides a near-optimal solution in worst case scenario [7,13,14].

Although tabu search is a very pragmatic and successful algorithm since it searches the solution space for a globally optimal solution, it still has some drawbacks which make it undesirable for some applications. Practically, the algorithm is non-deterministic and it requires some additional space to store the tabu list [16]. Also, it is an arduous job to choose the search space for the algorithm and decide the effective neighborhood structure.

2.3.3 Bayesian Optimization

Bayesian algorithms are based on the assumption that an unknown function could be modeled heuristically as a sample function of a Gaussian random process. The function satisfies the following properties:

- The expected value of the unknown function f(x) conditioned on all the measurements taken, is a piecewise linear approximation of f(x) itself.
- The conditional variance of the approximation is quadratic between observation points.

At each iteration of the algorithm, the next solution point is chosen to be a point in the search domain which maximizes the probability that the existing function exceeds the largest value by some positive constant conditioned upon all past evaluations of the function [26].

$$Prob[\, f(x) \geq f_n^{max} + \varepsilon_n \mid X_1, X_2, \ldots, X_n \,] \tag{2.2}$$

where,

$f(X)$ – Unknown function

X_n – Selected solution point at n^{th} stage

f_n^{max} – Solution with maximum value

ε_n – Positive constant

Bayesian optimization methods find the globally optimal solution by minimizing the number of evaluations of the function, because it searches for each solution point in the solution space only once. This optimality comes at the expense of increased computational time. The search space for the algorithm is bounded and thus the results are deterministic. This ensures that the algorithm generates same results for different runs on same solution space. The high computational complexity makes the algorithm impractical for the problems with more than 8-10 design variables.

2.3.4 Random Walk

Random walk is a simple process of visiting the solution points of a solution space, in a random order. The algorithm starts from a fixed solution point, and at each step, moves to one of the neighboring solutions randomly chosen according to a predefined distribution [27]. The algorithm keeps selecting different solution points till it reaches an optimal solution.

The algorithm is very simple to implement as it does not involve any type of complex calculations to select the next neighbor. No additional memory is required to keep track of the previous solution points. But, one of the important concerns for the methodology is to how to decide that the random walk has reached a satisfactory solution node? Also, since there is no recordkeeping of the previous stages, the algorithm does not learn and hence has no intelligent. This makes it impossible to identify the direction of search in the search space and determine the time it will take to converge to an optimal point.

2.3.5 Greedy Search

The algorithms which work on the concept of choosing the best possible outcome at each stage without taking into consideration the effects of those choices in future are classified as greedy algorithms [29,47]. These algorithms are easy to implement because at each step the decision is made on the basis of very limited information available at that stage and the information is not carried forward for use in the stages followed.

Convergence to an optimal or sub-optimal solution is very fast when a greedy algorithm is used. But, greedy algorithms get stuck in local minima and hence the results generated by them are not globally optimal.

2.4 Game Theory

Game theory, developed by John Von Neumann and Oskar Morgenstern in 1944 [30], is a collection of mathematical models formulated to study the situations of conflicts and cooperation. More precisely, it's a general theory of rational behavior for situations in which rational decision makers (players) have available with them a finite number of courses of action, each leading to a well defined outcome or end with gains and losses expressed in terms of numerical payoffs associated with each combination of courses of action for each decision maker [34]. Game theory studies the phenomenon of negotiation between rational agents in conflict situations, in a very general setting. Game theory does not cover the games of changes where the decision makers do not have any influence on the results [32,35].

2.4.1 Elements of Game

The essential elements of a game are [32,36,48]:

- *Players:* Players are the individuals who make decisions. The decision maker's goal is to maximize its utility by choice of strategies.

- *Action:* An action or a move by a player is the choice it can make.
- *Strategy:* A player's strategy is the rule that tells him which action to choose, on the basis of the information available with the player.
- *Strategy Combination:* A set of one strategy for each player in the game.
- *Strategy Space:* A set of all the possible strategies of all the players in the game.
- *Payoff:* Payoff is the utility or expected *utility* a player receives as a function of the strategies chosen by him and all other players.
- *Information:* A player's information set is his knowledge at a particular time, of the values of different parameters.
- *Equilibrium:* It is the strategy combination consisting of the best strategy for each player in the game. The equilibrium is represented as 1.1:

$$S^* = \{S_1^*, S_2^*, S_3^*, \ldots, S_n^*\} \tag{2.3}$$

where

S^* - Equilibrium strategy set

n - Total number of players

S_i^* - Best strategy for player i

- *Objective:* Objective is the goal of game, i.e. whether we wish to maximize the payoff for each player or to minimize it.

2.4.2 Classification of Games

The games could be classified on the basis of different parameters of classification [33,37,38,47,48]. Some of the major classification criteria are discussed here.

2.4.2.1 Cooperation

- *Cooperative Games*: The games in which either players can make commitments to coordinate their strategies or some external factors could enforce coordination are

called cooperative games and the solution to a cooperative game is a cooperative solution.

- *Non-Cooperative Games:* The games in which neither players, nor any external agency could enforce the coordination are called non-cooperative games. In these games, the cooperation is not enforced externally but is self-enforcing. In a non-cooperative game, all the possibilities of commitment are incorporated in the rules of game and the solution of game reflects that.

2.4.2.2 Number of Players

- *Two-player Game:* The games with just two players competing for resources.
- *Multi-player Game:* In games with more than two players, the complexity of game is much higher than a two player Game.

2.4.2.3 Time Dependence

- *Static or Simultaneous Move Game:* In such a game all the players make the moves simultaneously, without the knowledge of the strategies chosen by other players in the game.
- *Dynamic or Repeated Game:* The game in which the players interact by playing the game multiple times. By playing the same game multiple times, each player has some information about the strategies of other players and thus the strategy chosen by the player is contingent on the strategies being chosen by other players during past moves.

2.4.2.4 Degree of Opposing Interest

- *Constant-sum Game:* In these games, the sum of payoffs for all the players is a constant for any outcome. Here, the gain of one player is always at the expense of the loss to other player. A *zero-sum* game is a special case of constant-sum game in

which the sum of payoff for all the players is zero and is obtained by normalizing the payoffs. Sporting events are a good example of such games.

- *Variable-sum Game:* Here, the sum of the payoffs to all the players is not a constant value. This means that the gain to one player may not be a loss to another player i.e. the players have some strategies of common interest which could lead to an outcome where all the players are better off by cooperation. These are also known as *Non Zero-sum* games

2.4.2.5 Number of Moves

- *Finite Game:* In a finite game, there are some fixed set of rules and boundaries and the game is played within those set of boundaries in order to win and end the game. Here, the same game is played a finite number of times and the equilibrium generally remains constant.
- *Infinite Game:* The game has no fixed rules and boundaries and is played for infinite iterations and the equilibrium point in such games changes over the course of time and tries to learn from the previous results.

2.4.2.6 Types of Strategies

- *Pure Strategy Game:* In a pure strategy game, the players have pure strategies. A pure strategy for a player is a complete plan of play for him for its information set.
- *Mixed Strategy Game:* In such a game, there is a probability (>=0) associated with a player's each set of actions. The sum of the probabilities for the complete set of strategies is 1.

2.4.2.7 Amount of Information

- *Complete Information Game:* In games of complete information, the player has all the information about other players' strategies before it makes a move. He knows at least as much as those who have moved before him. These games are further

categorized in two types; *Perfect Information,* where the player knows about all the possible moves of other players and outcomes of the game, and *Imperfect Information,* where the player know all the possible outcomes of the game but not the moves of other players.

- *Incomplete Information Game:* Here the player who has to make a move does not know everything that has happened so far. Most of the simultaneous moves games are imperfect information games.

2.4.2.8 Type of Mathematical Formulation

- *Normal Form Game:* A normal form game is a game with a finite set of players, each having a finite set of strategies and utilities associated with them, make simultaneous moves and receive some payoff.

$$\Gamma = (S_i, u_i);\ i \in I \tag{2.4}$$

where,

I – Set of players from i = 1, 2,…, n

S_i – Strategy set of player i

u_i – Utility function of player i

$$u_i = \prod_{i \leq n} S_i \rightarrow \Re \tag{2.5}$$

- *Extensive Form Game:* An extensive form game can be represented as a tree, whose each node represents a choice made by a player. In such a game the players make their choice in a sequential order. It gives a detailed description of all the possible moves, outcomes and information for the game.
- *Characteristic Function Form Game:* This type of game provides the possible outcomes for a set of players forming a coalition and working in cooperation.

2.5 Nash Equilibrium Algorithm

In a game theoretic setting, the solution of the game should reflect rational behavior of players. *Nash Equilibrium* optimization algorithm is one such solution. Nash Equilibrium for a game is a combination of strategies, one for each player, such that for a player *i,* if the strategy combination of all other players remains the same, player *i* would do worse by not playing the Nash Equilibrium strategy [31].

Mathematically, for a Normal form game represented in (2.4), the strategy combination $\{s_1^* \in S_1, s_2^* \in S_2, \ldots, s_i^* \in S_i, \ldots, s_n^* \in S_n\}$ is a Nash Equilibrium, if for all *i,*

$$u_i\{s_1^*, s_2^*, \ldots, s_i^*, \ldots, s_n^*\} \geq u_i\{s_1^*, s_2^*, \ldots, s_i, \ldots, s_n^*\} \tag{2.6}$$

where,

u_i – Utility for player *i*

s_i^* – Nash Equilibrium strategy for player *i*

S_i – Strategy set for player *i*

If the strategy sets S_i for all the players are convex, then there always exists at least one Nash Equilibrium. More than one Nash Equilibrium can exist for a game, but there is no way to select if one is better than other.

Nash Equilibrium algorithm finds a globally optimal allocation of resources to the players. The solution is a good quality solution, which means that the allocation made to each player is the in the best interest of the players and the complete game as well. But, evaluating the Nash Equilibrium solution is an NP complete problem [44] and hence the convergence of the algorithm is non deterministic. Also, it is not necessary that a Nash Equilibrium solution is pareto optimal and visa versa.

Numerous methodologies have been proposed for finding the Nash Equilibrium solution for the games and some of those have been discussed in [39,40,41,42,43].

2.6 Why Nash Equilibrium?

For the multi-event crisis management scenario, we propose to define the resource allocation problem as a game and generate the optimal allocation using Nash Equilibrium algorithm. The decision of adopting Nash Equilibrium algorithm against the other candidate algorithms that we have discussed in section 2.3 is based on the analysis of the comparative study of these algorithms (Table 2.1) based on the various parameters relevant to the problem of multi-event crisis management under consideration [7,16,17].

Table 2.1 Comparative Study of Algorithms

Parameter → / Algorithm ↓	Knowledge	Optimal	Quality of Solution	Complexity		Individual Rationality	Stability
				Space	Time		
Genetic Algorithm	limited or none	Global	Satis.	High	High	Satisfactory	Satis.
Tabu Search	Prior Info. Required	Global	Average	High	High	Non determin.	Satis.
Simulated Annealing	limited or none	Global	Satis.	Medium	High	Non determin.	Satis.
Bayesian Method	Prior Info. Required	Global	Average	high	High	Non determin.	Satis.
Random Walk	limited or none	Local	Non Satis.	Low	Non determ.	Non Satisfactory	Non Satis.
Greedy Search	limited or none	Local	Non Satis.	Low	Low	Non Satisfactory	Non Satis.
Nash Equilibria	Information Required	Global	Satis.	High	High	Satisfactory	Satis.

As evident from the comparative study, the algorithms which perform better on most of the parameters are genetic algorithms, hill climbing algorithms, Bayesian method and Nash Equilibria. Among these algorithms, genetic algorithms can not be chosen for our scenario, because they do not provide individual rationality since some of the members of population

becomes dominant as the algorithm progresses. In case of multi-event crisis management, each crisis event is very important and the decision maker needs to make rational decision for it. There are different priority levels for crisis events, but each of them is required to be serviced in the optimum and rational manner.

Hill climbing algorithms (tabu search and simulated annealing) do not provide a good quality solution and the individual rationality for these algorithms is non deterministic. So these algorithms are not acceptable for our scenario where the quality of solution is of primary importance.

Bayesian algorithms have high space and time complexities which can not be controlled. Also, they provide an average quality of solution and the individual rationality of player is non deterministic, which is unacceptable in the scenario of multi-event crisis management.

Nash Equilibrium algorithm exhibits unique property of *social optimality* [45] that is very important in the case of Multi-event crisis management and has not been addressed by any other algorithm. In such a situation, the individual fairness for each player is optimum and the average fairness of the system is high. Hence, the Nash Equilibrium solution provides a better quality of solution where the individual rationality of players is very good.

The property of *social optimality* or social utility ensures that each player of the game receives the best utility for himself and for the complete system. In this setting, the optimization of individual utility ultimately leads to a system wide optimization.

CHAPTER 3

FORMULATION OF A NON-COOPERATIVE GAME

In a crisis scenario with multiple emergencies existing at some instance of time, the allocation of requested response units to the emergency locations is a challenging problem. We propose a game theoretic solution for the emergency response problem which uses Nash Equilibrium algorithm for the optimal allocation of the requested resources to the appropriate crises. In this chapter, first the problem of multi-event resource allocation will be formally defined and realized in a game theoretic setup. Then, the notations that would be used in this and the following chapters will be specified and explained. The chapter will conclude with the definition of objective function for the game.

3.1 Formulation of Game

A hypothetical example of a multiple crisis environment is illustrated in Figure 1.2. The same example scenario will be studied in a more comprehensive manner in this and the following chapters and a multi player, non-cooperative, non zero-sum normal form game for resource allocation will be formulated.

Figure 3.1 shows a minor variant of the previously described scenario. This variation will be used in addressing some more intricate issues that could arise in a multi-crisis scenario. As shown in Figure 3.1, at some moment of time T_0 four crisis events have occurred; an air-plane crash, a fire in a building, a road accident and fight at a stadium. The locality in which these crises have occurred is “AREA A”. In AREA A, there exist two hospitals, two police stations and

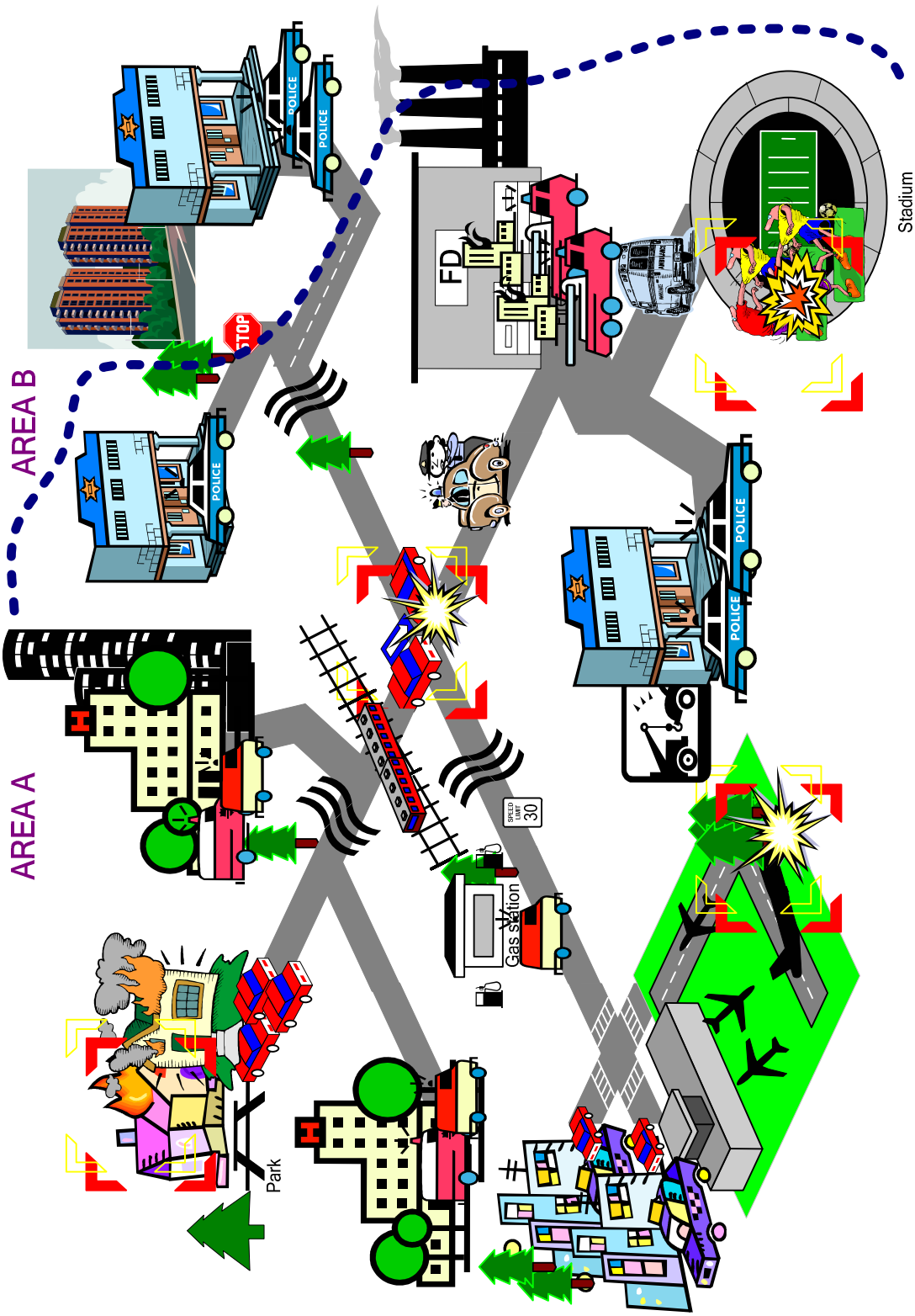

Figure 3.1 Multi-Event Crisis Scenario with Area Distinction

a fire station. Now, all these crisis events have certain resource requirements. The issue is to optimally allocate the types and the number of units of resources required these crises.

The resource requirements of the crisis events have been listed in Table 3.1.

Table 3.1 Resource Requirement of Crises

	Ambulances	Fire Engines	Police Cars	Towing Trucks
Road Accident	3	0	6	1
Building Fire	0	5	1	0
Air-plane Crash	2	3	3	0
Stadium Fight	2	0	7	0
Total Req.	7	8	17	1

Table 3.2 Resource Availability at Depots

	Number of Units Available	Number of Units Expendable	Residing Area
Police Station 1	10	9	A
Police Station 2	5	5	A
Police Station 3	4	4	B
Hospital 1	5	5	A
Hospital 2	4	3	A
Fire Station	10	9	A
Towing Agency	2	2	A

Table 3.2 shows the total number of units available at each emergency response location as well as the number of units expendable by that location. The residing area of each response location is shown in the third column of the table.

Now, if a methodology could be formulated to optimize the allocation of one type of resource, be it police cars or ambulances, we can manage the allocation of resources of other types can also be implemented by implementing the same methodology repetitively. Since all the crisis events in the scenario require allocation of one or more police cars, a game for the allocation of police cars to the required crises locations has been formulated in the following sections.

3.1.1 Modeling of Single Resource Type Non-Cooperative Game

The players and the resources are two building blocks of a game theoretic setup. The players compete for a minimum cost allocation of available resources. In the context of this work, the crisis locations are the players of game and the emergency response locations are the resources. Each player has a resource requirement (number of police cars in the present scenario) which needs to be satisfied by the resources in the vicinity. Also, each player wants to receive the required resources in least amount of time. So, the players do not form a coalition and play a *non-cooperative game* in which each one tries to maximize its utility.

3.1.1.1 Identification of Game Area

In order to realize a game, the players and the resources need to satisfy the following condition.

$$\textit{Total number of resource units required} \leq \textit{Total availability of resources} \quad (3.1)$$

Figure 3.1 shows the existence of four players (crises) in our game area “A” and the resource depots (police stations) that exist in that region. The total requirement and the availability is displayed in Figure 3.2.

Total resource requirements of players = 6 + 1 + 3 + 7 = 17 units

Total Resource Availability in Area A = 10 + 5 = 15 units

Total number of units expendable (Area A) = 9 + 5 = 14 units

Current game area = “A”

Figure 3.2 Player Requirements and Resource Availability in Area A

The total number of expendable resource units available in current game area A is just 14 units, which is well short of the total requirement of 17 units. This does not satisfy the necessary condition for the game formulation as defined in equation 3.1. So, the game area needs to be incremented such that the resource depots in the nearby locations are also incorporated in the game area and the total number of expendable resource units is at least equal to the total requirements. For this scenario, the game area is incremented from area A to Area B, so that a third resource depot that exists in area B is also included in the game. Now the total availability in the combined area A + B is recalculated. Figure 3.3 shows the summary.

Total resource requirements of players = 6 + 1 + 3 + 7 = 17 units

Total Resource Availability = 10 + 5 + 4 = 19 units

Total number of units expendable = 9 + 5 + 4 = 18 units

Current game area = “A + B”

Figure 3.3 Player Requirements and Resource Availability in Area A+B

The new allocation satisfies the necessary condition defined in 3.1, for formulation of game. The next step is to associate a cost for the allocation of each unit of resource from a resource depot to the crisis.

3.1.1.2 Allocation Cost

Once the minimum resource requirement constraint is satisfied, the next important concern is to decide how to allocate the resource units to the players. In order to play a game, a cost associated for availing a resource unit from a depot to a crisis location is calculated. Based upon the costs, the players form their strategies.

In the case of emergency management, the price (cost) is actually a function of a number of parameters. The parameters are:

- Distance of Resource location from the crisis (player)
- Degree of criticality of the crisis event
- Number of units requested from a particular resource location
- Minimum and Maximum speed limits on the route between resource and player
- Traffic conditions on the route

The cost function is also used to find the payoff for selecting a particular strategy by a player.

3.1.1.3 Definition of Strategy

The most crucial step of realizing a game is to define a strategy for a player. In the scenario being discussed here, a strategy could be defined as the number of units each player requests from each resource location, in order to satisfy its requirements. For example, a strategy for the road accident which requires 6 police cars could consist of a request of 2 units of resources from police station 1, 1 unit from police station 2 and 3 units from police station 3. Figure 3.4 shows a pictorial view of the requests and requirements.

This is a simple as well as a realistic way of realizing a strategy, but when the number of available resource locations increase above a certain extent, the strategy set for each player explodes and the game could not be played because the time complexity of Nash Equilibrium algorithm is polynomial. So, an alternative notion of strategy has been defined.

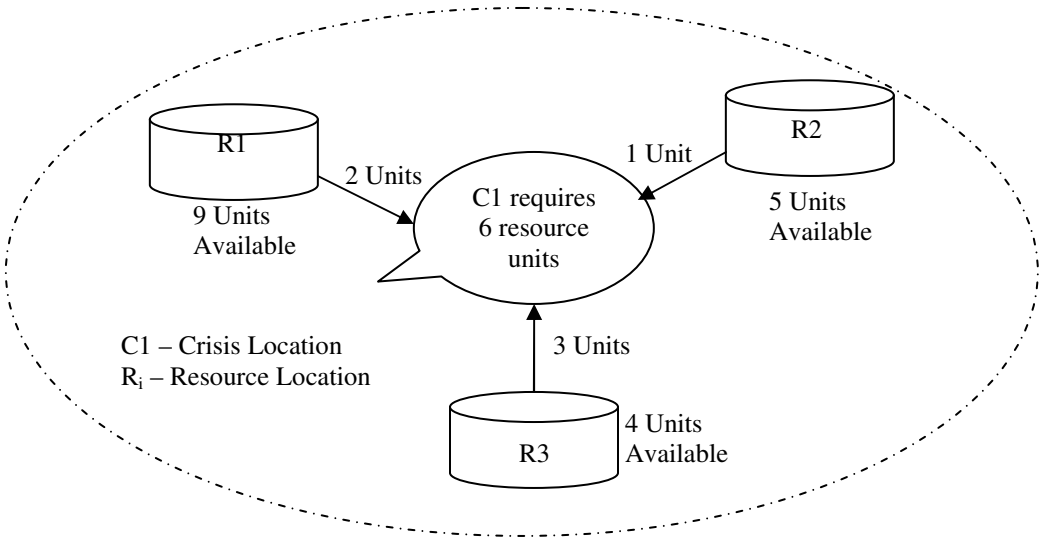

Figure 3.4 Resource Requirements and Availability

Initially, the allocation of the resources is done according to the minimum cost allocation methodology. For each player, the resources are allocated according to the sorted cost of resource locations for it, irrespective of the allocation of resources to the other players. After the initial allocation is done, for each resource location, if the total number of units allocated to the players is more than the number of expendable units actually available with the resource location, the number of overhead units is found. Now each player who was allocated one or more units of that resource will form a set of strategies. The strategy for each player would be the number of units it may have to loose in order for the resource location to be consistent i.e. the resource center is not allocating more than what is available with it.

In the example scenario, assuming that the sorted order of resources for each player is as given in Figure 3.5, the initial allocation will be according to the results in Table 3.3.

As shown in Table 3.3, the total allocation for Police Station 2 is 6 units, but the actual availability of resources at the location 5 units. So, there is a conflict of one resource unit, which either player 2 or player 4 would have to loose. So, for Police station 2, a game with the strategies

of Player 2 as {loose 0 unit, loose 1 unit} and strategies of Player 4 as {loose 0 units, loose 1 unit} will be formulated.

Player 1 (Road accident): Police Station 1 < Police Station 2 < Police Station 3

Player 2 (Building Fire): Police Station 2 < Police Station 1 < Police Station 3

Player 3 (Air-plane Crash): Police Station 3 < Police Station 2 < Police Station 1

Player 4 (Stadium Fight): Police Station 2 < Police Station 3 < Police Station 1

Figure 3.5 Sorted Orders of Resources for Players

Table 3.3 Initial Resource Allocation

	Police Station 1	Police Station 2	Police Station 3
Player 1	6	0	0
Player 2	0	1	0
Player 3	0	0	3
Player 4	0	5	2
Total	6	6	5

The definition of strategy ensures that there are a defined number of moves possible for each player and those moves are independent of the choices made by other players. This leads to a *normal form game* in which the players play their strategies simultaneously.

3.1.2 Payoff

For each strategy chosen by a player, there is a utility (or expected utility) associated with it that the player would receive as a function of the selected strategy and the combination of

other players' strategies. This utility is known as the payoff paid/received by the player, depending upon whether the payoff is modeled as a gain or a loss to the player.

In our scenario, we model the payoff as a function of loss incurred to the player playing the strategy and gains for the other players when they play their respective strategies. A payoff matrix is created for each player with rows as its strategies and columns as the combination of other players' strategies.

3.1.3 Nash Solution

After the payoff matrices are created for all the players playing the game, an optimal solution for resource allocation is found by implementing a Nash Equilibrium algorithm. The algorithm results socially optimal solution. For this system, the Nash Equilibrium algorithm described in [42] has been implemented. The algorithm is based on a combinatorial theorem [43] that is expressed in terms of primitive sets. Some modifications have been made to the algorithm in order to make it work for the proposed methodology. The algorithm accepts the payoff matrix as the input and generates a probability vector as the output. These probability vector values are used for selecting the equilibrium strategies.

3.2 Notations

In order to formally define the problem and its solution, the following notations (figure 3.6) have been defined for the game.

P	Set of crisis locations (Players), P = $\{P_1, P_2,\ldots,P_N\}$
N	Total number of players
Q	Set of resource locations, Q = $\{Q_1,Q_2,\ldots,Q_K\}$

Figure 3.6 Notations

K	Total number of resource locations containing divisible resources
Z_i	Total number of resource units required by Player P_i; i=1, 2,…,N
q_j	Number of Resource units available at resource location Q_j; j = 1,2,…,K
S_i	Set of feasible strategies of player P_i
S	Set of all feasible strategies in a game, S = $\{S_1, S_2, \ldots, S_N\}$
r(i)	Total number of strategies of player P_i
s_i^b	b^{th} strategy of player P_i, where b = 1…r(i) and $S_i = \{s_i^1, s_i^2, \ldots, s_i^{r(i)}\}$
u_i^b	Number of resource units specified in strategy b of player P_i
e_i^k	Number of resource units requested by player P_i from the resource location Q_k
c_i^b	Price which player P_i pays if it selects the strategy b
m(i)	Total number of strategies combinations of other players for a strategy of player P_i $m(i) = r(1)*r(2)*\ldots*r(i-1)*r(i+1)*\ldots r(N)$
B_i	Set of combinations of all the strategies of players other than player P_i, $B_i = \{B_i^1, B_i^2, \ldots, B_i^h, \ldots, B_i^{m(i)}\}$
$g_i^h(b)$	The gain incurred by the other players when player P_i its strategy b and they play the strategy combination B_i^h
$X_i^b(h)$	Payoff for player P_i if it plays a strategy b and other players play a strategy combination B_i^h
D_i^k	Distance of player P_i from a resource location Q_k

Figure 3.6 Continued

V_i^k	Average speed on the route between player P_i and resource location Q_k
t'	Inter-arrival time between two resource units of same type, coming from same resource location
L_i	Criticality level of the crisis event P_i. The values of L_i range from 1 to 10, where 1 is the lowest and 10 is the highest level of criticality
Y_i^k	Unit Cost of resource Q_k for player P_i (Base Cost)
W_i^k	Constant for the traffic conditions on the route between player P_i and resource location Q_k. It could have three values 0.8, 1 or 1.2 depending upon the traffic conditions, 0.8 when the traffic is very crowded, 1 when it is normal, and 1.2 when the traffic conditions are favorable i.e. there is not much traffic on route.
V'_i^k	The adjustment in average speed, which depends upon the value of W_i^k

Figure 3.6 Continued

3.3 Formulations

- The change in average speed under the effect of varying traffic conditions on the route between the player (crisis location) and the resource center is given as:

$$V'^k_i = V_i^k * W_i^k \tag{3.2}$$

- *Base Cost:* The base cost i.e. the cost of allocation of a single unit of resource to the player is defined as:

$$Y_i^k = L_i * (D_i^k / V'^k_i)$$

$$= \infty \text{; if resource does not reside in game area} \tag{3.3}$$

- *Cost Function:* Cost of a strategy for a player is defined as the overhead which the player P_i pays for loosing the number of units specified in the strategy and getting those units allocated from the next available resource location in order to satisfy the requirements.
- *Payoff Function:* Payoff function is a function of the Cost overhead for the player P_i and the combined gains incurred by other players when P_i plays the strategy s_i^b and the other players play the strategy combination B_i^h. We model the payoff as the combined minimization function.

$$X_i^b(h) = c_i^b + g_i^h(b); \text{ if the strategy combination is valid}$$
$$= \infty; \text{ otherwise} \quad (3.4)$$

Valid strategy combination is the one in which the total number of units being sacrificed by the player P_i playing strategy b and the sum of units sacrificed by all other players by playing strategy combination B_i^h, is equal to the overhead allocation.

3.4 Objective

The objective for each player is to minimize the gains incurred by other players and to minimize its own loss for playing a particular strategy. Objective function is:

$$\text{Minimize } (X_i^b(h)) \quad (3.5)$$

In this chapter, a non-cooperative normal form game was formulated for a multi-event crisis scenario where the crisis locations require some resources to be allocated for recovery and the resources are limited in quantity. In such a situation, for each such resource type there is a competition among the crises for allocation. Hence, a game theoretic formulation for optimal resource allocation was proposed. The notations were also specified and explained in this chapter.

CHAPTER 4

NASH EQUILIBRIUM SOLUTION

In this chapter, an algorithm for the actual implementation of allocation of resources to the players will be provided. The discussion will encompass all the steps from identification of the region that would be included in the game, to the implementation of Nash Equilibrium algorithm for optimal allocation of required resources to the players.

In the last section of the chapter, the software architecture of the proposed system will be discussed in details. This would identify the appropriate place of the proposed subsystem for optimal resource allocation in the domain of emergency management system like FEMA. The chapter will conclude with a short discussion on the Nash Equilibrium methodology being used.

A top level workflow view of the proposed methodology is described in Figure 4.10. It specifies the sequence of operations that are carried in order to generate an optimal solution.

4.1 Requirement Analysis

At a particular time stamp, one or more crisis locations may exist in a region. The very first step in the algorithm is to identify those crises and their resource requirements. The system also has information about the resources available in that region. Once the resource requirements are identified, the algorithm checks if the total number of requested resource units are available in the initial game region. If the current game area is short of resources, the area is incremented to the point that the system finds another resource center. Now the verification step is executed again to check if the total request by all the crises can be satisfied in the increased area.

This cycle continues till we have enough resources available with us to satisfy the requirements of all the crises.

Figure 4.1 explains the steps followed in the requirement analysis step.

1. For the initial game area, find the total requirements of all the crisis locations

$$z = \sum_{i=1}^{N} z_i$$

2. In the initial region (1), find the total number of resource units available.

$$q = \sum_{j=1}^{K} q_j$$

3. If $z > q$ then
   - 3.1. Increment the region till a new resource depot is found.
   - 3.2. Add the resource depot to the list of available resource depots and then go to step 2.

Figure 4.1 Requirement Analysis

4.2 Cost of Resources

Once the game area is determined on the basis of requirements and the availability, the price that a crisis location will have to pay for the potential allocation of resource units from a resource location is computed. The cost is calculated for each player and for each resource location. The cost function is defined in Equation 3.3. The next important step is to sort the resources in non-decreasing order of cost for each resource. The sorting is required because the initial allocation in the implementation is the minimum cost allocation.

For sorting the resources *Quick Sort Algorithm* has been implemented, because the average case time complexity of algorithm is $\Theta(n \log n)$. The worst case for quick sort surfaces when the array is already sorted. In that case, the performance is $\Theta(n^2)$. The algorithms performs an in-place sorting i.e. it does not require additional space for sorting. Also, the sorting

is stable and has a small constant factor. Small constant factor ensures that the time-complexity expressed in terms of theta is realistic and the actual time required by the algorithm is a very small multiple (of $n\log n$ or n^2). [55]

The cost module is described in Figure 4.2.

1. For each crisis location in the scenario do,
   1.1. Find the base cost of receiving a single unit of resource from each resource location. The cost function is Y_i^k where i∈ N and k∈ K.
   1.2. Create an array *overheadCost* of the K elements and store the cost values and the corresponding array index number in array.
   1.3. Call the *QuickSort* module and pass the array to the function.
   1.4. *QuickSort* module returns the array, sorted in non-decreasing order of cost and keeping the array index number along with the cost value.

Figure 4.2 Cost of Resources

4.3 Allocation

After the resources are sorted in the increasing order of cost for each crisis location, the following steps are performed in sequence.

- *Initial Allocation*: A minimum cost allocation of resources to the crisis location is carried out, irrespective of the allocation to other players.
- *Verify Allocation:* After the initial allocation is made, it is verified that for each resource location, the sum of the allocations made to all the crises is less than or equal to the total number of resource units available with the resource depot. If the total allocation is more than the total availability, a game is required to be played for reallocation of the extra units of conflicting resources.

The module is described in Figure 4.3.

1. *Initial Allocation*
    1.1. For each crisis location do
        1.1.1. Define *remRequirement* $\leftarrow z_i$ as remaining number of units required
        1.1.2. Allocate the *remRequirement* number of units to the crisis location such that the crisis first receives all the available units from the resource location with minimum cost and then the remaining requirements from the next resource location in the sorted order and so on, till the total requirements of the crisis location are satisfied.
    1.2. Store the complete allocation in a *N*K* matrix called *allocationMatrix*
2. *Verify Allocation*
    2.1. For each resource location do
        2.1.1. Calculate the total number of units allocated to all the crisis locations
    2.2. Store the total allocation count for each resource location in an array *conflictResource*.
    2.3. For count from 1 to k do
        2.3.1. If *conflictResource[count]* > q_{count} then
            2.3.1.1. There is a conflict in allocation and we need to play a game.
            2.3.1.2. Exit from this module. Go to the strategy generation module.
    2.4. If none of the resource locations are in conflict then allocation is an optimal allocation. Exit successfully.

Figure 4.3 Initial Allocation

4.4 Generation of Strategy

If there is no conflict found during the verification step, the allocation would be the most optimal allocation because each crisis receives the resources at the lowest possible cost and the complete system is in the optimum state. But, if the verification step finds a conflict in resource allocation, i.e. it detects a condition when there is at least one resource location with number of units allocated greater than the number of units actually available, a game is required to be played for each such resource center.

The strategy for each crisis location is defined as the number of units it can loose in order for the resource location to reach a consistent state. A criticality level (L_i) is associated with each crisis that decides the sequence in which games are to be played. If more than one resource location is in conflict, then we will first select the conflicting resource center which is the least cost resource center for the highest criticality level crisis location that is part of the conflict and play a game for that center. Once a game is played for the resource center and the overhead allocation is removed from that location and reallocated to the next available resource centers (in sorted order of their costs) for the players of the game, that center attains a consistent equilibrium allocation state. The algorithm then moves on to the next conflicting resource location and repeats the same process, but it takes into consideration that the resource locations that have attained a consistent state are not affected.

Figure 4.4 discusses the steps involved in generation of strategy in detail.

1. *Find total number of conflicting resource locations and units*
   1.1. For count = 1 to k do
      1.1.1. If *conflictResource[count]* > q_{count} then
         1.1.1.1. *conflictingUnitCount[count]* ← *conflictResource[count]*- q_{count}
         1.1.1.2 *conflictingResourceLocations*++
      1.1.2 Else
         1.1.2.1 *conflictingUnitCount[count]* ← *0*
2. *Select resource location for game*
   2.1. For criticality level *m* from 1 to 3 do
      2.1.1. For each crisis location *i* do
         2.1.1.1. If L_i == *m* then
            For each resource location *k* in *i*'s sorted order do

Figure 4.4 Generation of Strategy

If *allocationMatrix[i][k] > 0* and

conflictingUnitCount[k] 0 Then

Go to *GenerateStrategy(k)*

2 *GenerateStrategy(k)*

2.1 For each crisis location *i* do

2.1.1 If *allocationMatrix[i][k]>0* then

2.1.1.1 If *conflictingUnitCount[k] > allocationMatrix[i][k]* then

strategyCount[i] = allocationMatrix[i][k]+1

2.1.1.2 Else

strategyCount[i] = conflictingUnitCount[k]+1

2.1.2 Else

2.1.2.1 *strategyCount[i] = 1*

2.2 For each crisis location *i* do

2.2.1 If *strategyCount[i] > 1* Then, include the crisis in game otherwise do not include.

2.2.1.1 Create an array *stratArray[i]* with size equal to *strategyCount[i]*

2.2.1.2 Create a *strategyCost[i]* array of same size, which will store the loss incurred by the player for each strategy.

2.2.1.3 Strategy for a player is the number of units it may need to loose and the cost of each strategy is the number of units lost * *overheadCost[k]*

stratArray[i] = {0, 1, 2, …, strategyCount[i]-1}

*strategyCost[i] = {0*overheadCost[k], 1* overheadCost[k], …, (strategyCount[i]-1)* overheadCost[k]}*

2.3 After creating a strategy array for each player in the game, call the payoff matrix module.

Figure 4.4 Continued

4.5 Formulation of Payoff Matrix

After the strategy set for each player of the game is finalized, the next important step is to formulate a payoff matrix for each player. The payoff matrix for a player *i* is of the size *r(i)* * *m(i).* Each cell of the payoff matrix for a player contains a payoff value which is defined in Equation 3.4.

For a player P_i whose payoff matrix is being formulated, corresponding to each strategy it plays, all the other players play their own strategies. These players acquire some gain due to loss of player P_i, which is greater than or equal to zero. The gain acquired for each set of strategy combination in payoff matrix is calculated by implementing a recursive function. Figure 4.6 explains all the steps that are involved in formulation of payoff matrix.

A sample payoff matrix for a player P_i is displayed in Figure 4.5

P_i	B_i^1	B_i^2	…	$B_i^{m(i)}$
s_i^1	$X_i^1(1)$	$X_i^1(2)$	…	$X_i^1(m(i))$
s_i^2	$X_i^2(1)$	$X_i^2(2)$	…	$X_i^2(m(i))$
…	…	…	…	…
$s_i^{r(i)}$	$X_i^{r(i)}(1)$	$X_i^{r(i)}(2)$	…	$X_i^{r(i)}(m(i))$

Figure 4.5 Payoff Matrix

After formulating, the payoff matrix is passed to the Nash Equilibrium function, which calculates the probability vectors that decide the strategy combination to be chosen.

4.6 Nash Equilibrium Solution

After the payoff matrix is formulated for all the players of the game, Nash Equilibrium solution is found by implementing the algorithm provided by Scarf-Hansen fixed Point theorem

[42,43] to approximate Nash Equilibrium point in an *N*-person non-cooperative game. The algorithm is based on a combinatorial theorem [42] which is expressed in terms of a primitive set. Considering *X* to be a collection of *n*-dimensional vector $X = (x^1,\ldots,x^h)$ of the form $(m_1/D, \ldots, m_h/D)$' with each m_k greater than or equal to -1 and summing up to *D* which is a very large integer (mostly a multiple of the number of crises).

1. *Total number of strategies combinations of other players*
   1.1. For each player *i* do
      1.1.1. For each player $j \neq i$ do
         1.1.1.1. *NumCols[i] = NumCols[i] * strategyCount[j]*
2. *Create a payoff matrix*
   2.1. For each player *i* do
      2.1.1. Create a two dimensional array *Pays[i]* of size *strategyCount[i]*NumCols[i](=m(i)).* This is the payoff matrix for player P_i,
      2.1.2. Call *recCombiCost()* that calculates the gains of other players' strategy combinations and returns an array *returnArr[numCols]*containing the gains for all such strategy combinations.
      2.1.3. For each *j* from *1* to *strategyCount[i]* do
         2.1.3.1. For each *r* from *1* to *NumCols[i]* do
            If $(\sum$ *sacrifice units of* $B_i^j + s_i^j$ *==conflictingUnitCount[k])*
               Pays[i][j][r]=strategyCost[i][j]+returnArr[r]
            Else
               Pays[i][j][r]= ∞
3. Call Nash Equilibrium function and pass the payoff matrix *Pays,* number of Players and number of strategies of each player to the function.

Figure 4.6 Formulation of Payoff Matrix

Now,

$$M = \begin{bmatrix} m_{11} & m_{12} & \dots & m_{1N} \\ m_{21} & m_{22} & \dots & m_{2N} \\ \vdots & \vdots & \dots & \dots \\ m_{N1} & m_{N2} & \dots & m_{NN} \end{bmatrix}$$

The columns of matrix *M* represent the numerators on *n* vectors in *X* [42]. *M*atrix *M* is a *primitive set* iff and there is a rearrangement of the columns and a permutation of the labels of the columns, *I*(*l*) such that:

1. The *l*th column of *M* is identical to column *l-1*, except for the two rows *I*(*l*)-1 and *I*(*l*)

2. $m_{k,l} = m_{k,l-1} + 1$ for $k = I(l)-1$

   $m_{k,l} = m_{k,l-1} - 1$ for $k = I(l)$

Note: For $l = 1$, $l-1 = h$. Similarly, $I(l) = 1$, $I(l) - 1 = h$

The Nash Equilibrium algorithm based upon this concept of Primitive set is detailed in Figure 4.7. The algorithm accepts the payoff matrices, number of players *n* and *X* as input and generates an output that is a set of output vectors *p*.

1. Select *D* such that it is a multiple of *N* where *N* is the number of Players
2. Calculate $h = \sum_{i=1}^{N} r_i - N + 1$
3. Let x^j be an arbitrary nonnegative vector from *X*. We associate it with the probability vectors ($p_1^j, p_2^j, \dots p_N^j$) in the following way:

   $t_0 = 0$ and $t_i = r_{i-1}$ where $i = 1, 2, \dots, N$ and r_i is the number of strategies of player P_i

   3.1. Define

   $$u_{ki} = 1 + \sum_{v=0}^{i-1} t_v + k \; ; \qquad \text{where } k = 1, \dots, t_i \text{ and } i = 1, \dots, N$$

Figure 4.7 Nash Equilibrium Algorithm

$p^j_{k,i} = N \cdot x^j_{u_{ki}}$; where $k = 1, ..., t_i$ and $i = 1,..... ,N$

$p^j_{r_i,i} = 1 - \sum_{i=1}^{t_i} p^j_{k,i}$ where $p^j_{k,i} \geq 0$

4. While label of $x^j \neq 1$ do

4.1. If $p^j_{r_i,i} \geq 0$ then

4.1.1. Let $B_{ki} = B_{ki}(p^j_1,.., p^j_{i-1}, p^j_{i+1}....p^j_N)$ be the expected payoff to a player if it uses its k^{th} strategy and such that k_i is the lowest index for which

$B_{k_i i}(p^j_1,.., p^j_{i-1}, p^j_{i+1}....p^j_N) \geq B_{ki}(p^j_1,.., p^j_{i-1}, p^j_{i+1}....p^j_N)$

Then, the vector x^j is labeled $1 + \sum_{w=0}^{i-1} t_w + k_i$;

where $i = \min\{ l \mid p^j_{r_l,l} > 0 \text{ and } k_l \neq r_i\}$

4.2. If $p^j_{r_i,i} < 0$ OR $p^j_{r_i,i} = 0$ for all i OR $k_i = r_i$ for all i with $p^j_{r_i,i} > 0$ then

4.2.1. x^j is labeled 1

4.2.2. Exit from algorithm an return the probability vector to the calling function.

Figure 4.7 Continued

4.7 Reallocation

The Nash Equilibrium approximation algorithm returns a probability vector on the basis of which the reallocation of resources to the players is carried out. The game has been played for one resource location which was in conflict, so once the optimal strategy set is found, the number of units to be sacrificed by each player according to the selected strategy are reallocated from the next available least cost locations. Since, the resource location for which the game has been played just now is in consistent state, the location is labeled as "*consistent*" and no player in following iterations of the game can change the resource allocation at that location. Marking a

resource location as *consistent* ensures that the algorithm is not undirected and is moving towards the desired solution.

The sequences of operations carried out in the reallocation stage have been delineated in the Figure 4.8.

1. *Reallocation*
    1.1. According to the probability vector returned by the Nash Equilibrium algorithm, for each player, select the strategy which is assigned highest probability.
    1.2. For each player do
        1.2.1. The number of units of resources to be sacrificed according to the selected strategy is removed from the resource.
        1.2.2. The sacrificed units are allocated from next available resource locations in player's sorted order that that have units available with them and are not marked as "*consistent*".
    1.3. Call the *Verify Allocation* module with new allocation matrix.

Figure 4.8 Reallocation Algorithm

The above mentioned modules are executed repetitively until the *Verify Allocation* module ensures that none of the resource locations are have allocated more resource units than they actually have. When such a situation arises, the algorithm successfully terminates returning the final optimal allocation of resources to the crisis locations. Also, solution of each iteration of the game is a Nash Equilibrium solution, so the solution of the complete system also a Nash Equilibrium solution.

4.8 Why Terje-Hansen Algorithm?

The algorithm being implemented for finding the solution of the game is Terje-Hansen approximation algorithm. Herbert Scarf's fixed point theorem (in collaboration with Terje

Hansen) which is used as the basis for approximation of Nash Equilibria is considered to be the most successful and most implemented theorem. The solution generated by the algorithm is a probability vector and not a specific strategy combination or a set of combination. So it provides a generalized solution and the decision of choosing the strategies according to the probability vector values is left at the player's discretion. The time of convergence of solution can be controlled by the players playing the game. Also, the methodology works well for two player games as well as multiple player games and provides stable results.

4.9 Software Architecture

Emergency management is a vast area that incorporates everything from the identification of an anomaly in the system as a crisis, to the response and recovery from that crisis. After recovery, the critical process of mitigation of its effects, the risk analysis and a better preparedness for those types of crises are also integral parts of the emergency management system. A detailed study of the life cycle of a crisis [1] was performed in chapter 1 of this work and is diagrammatically described in Figure 1.1. A multi-event resource allocation and management system has been proposed in this work. The proposed system is an integral part of the emergency management system and performs the actual physical allocation of resources to the crisis locations.

The system tries to optimally allocate the resource units which are available in a limited quantity, to the crisis locations existing in the system at some point of time. Figure 4.9 shows the top level view of an emergency management system. The proposed resource allocation system fits in the response module of the main system which itself consists of a set of four activities including alert, assessment deployment and monitoring. The proposed system is an integral part of deployment module which is the shaded block in figure 4.9.

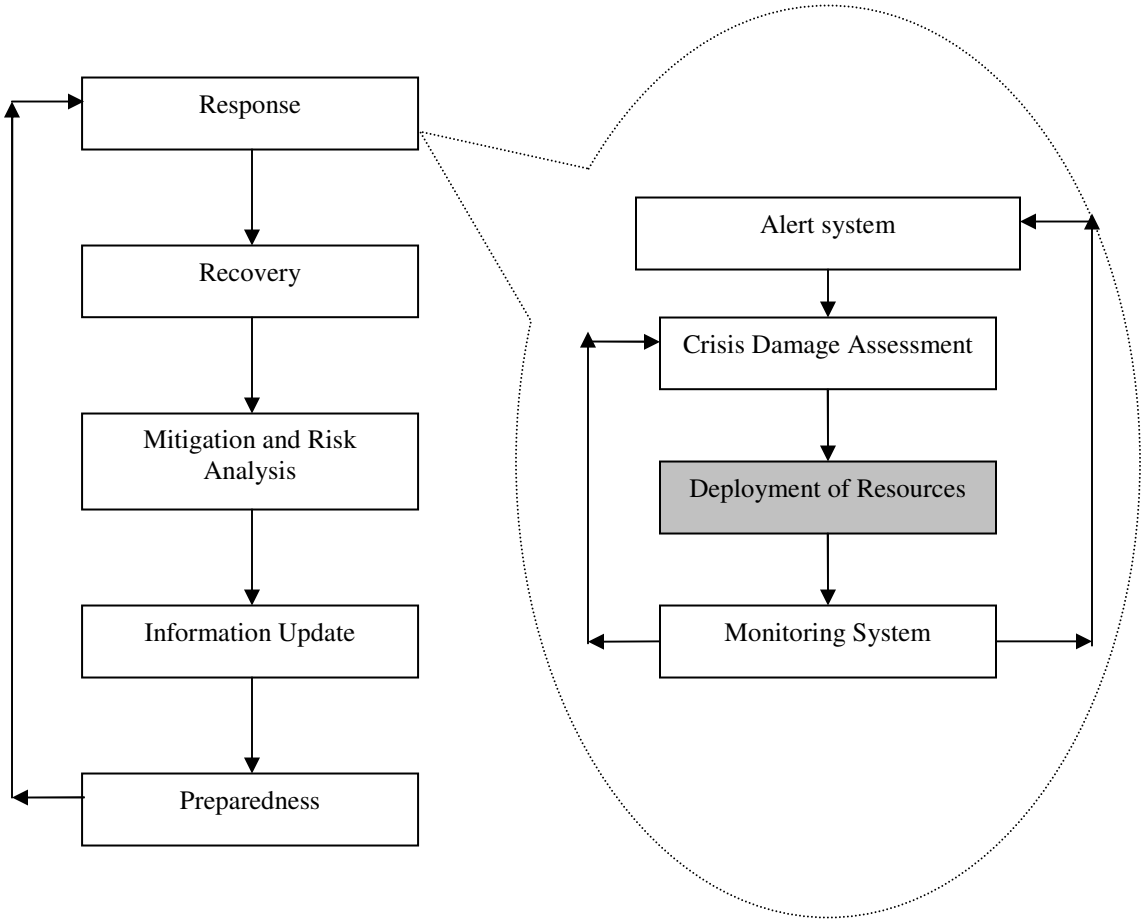

Figure 4.9 Position of Resource Allocation System

4.9.1 System Input and Output

Our system receives a set of different inputs for each crisis location from the automated systems like the GPS system or a GIS system. The inputs are used to calculate the cost and payoffs for each crisis location in order to provide an optimal allocation of resources to it. The inputs and outputs of the system are defined below.

Inputs:

- Number of crises (2 to 6)
- Number of emergency response centers (3- 17)

- Number of response units at each resource center (12-40)
- Resource request of each crisis (<=60)
- Criticality level of each crisis (vary from 1 to 10, 1 being the most critical and 10 being least critical)
- Response time for receiving units from each location to each crisis location (in seconds)

Outputs:

- The final allocation of resources to each crisis location.

4.9.2 Object Oriented Modeling

The proposed system has been implemented in C++ programming language because the modeling of the problem is most realistic in an object oriented paradigm. The complete system is a collection of objects which interact with each other in order to accomplish the goal of optimal resource allocation. For example, the resource centers and the crisis locations are real world objects which are interacting with each other to receive the best allocation.

Additionally, there are other advantages of modeling the problem in an object oriented paradigm. The complete system is divided into specific logical classes which identify each individual component of the system which performs certain functions and has some data structures associated with it. The classes provide an ease of reusability of its functions. Also, the program is more manageable if it is logically modularized. This functionality is best provided by a class.

Another important aspect of object oriented design is its capability of providing data abstraction and encapsulation. The system acts as a black-box for the external users. A set of interfaces are provided to the end user, which in this case is an automated system that provides inputs to the system, the outputs are generated by the system and are supplied to the user through

an interface. The functionality of the system is completely hidden from the end user, which ensures ease of use and system security.

4.9.3 Overview of Classes and Functions

The system defines the following classes in the implementation:

Table 4.1 Overview of the Classes

Class	Attributes	Behavior
GameArea	Initial Area, Availability, Requirements	• Identification of final game area
Cost	Unit Cost, Sorted Cost	• Identification of the cost of each resource unit for each player. • Sorting of resource locations in increasing order of cost for each crisis location
Allocation	Sorted cost, Allocation Matrix	• According to the sorted order of cost, initial allocation is done • Allocation is verified for conflicts • If conflict exist, then for each conflicting resource location a game is played • Reallocation is done according to Nash Equilibrium results.
StrategyGenerator	Allocation Matrix, Conflicting players, sorted cost	• If conflict exists, the strategy set is generated for each player, which is number of units each player looses. • Cost of each strategy is calculated
Game	Payoff Matrix	• A payoff matrix is formulated for each player.
NashEquilibrium	Probability Vectors	• Nash Equilibrium solution is evaluated

The following functions have been implemented in the classes.

SatisfyRequirements():

- Class: GameArea
- Input: Requirements, Availability, Initial Game Area
- Output: Final Game Area

BaseCost()

- Class: Cost
- Input: Crisis Parameters; distance, speed, criticality
- Output: Base cost from each location for each player

SortCost()

- Class: Cost
- Input: base cost
- Output: Call to QuickSort()

QuickSort()

- Class: Cost
- Input: base cost
- Output: sorted cost array for each crisis location

InitialAllocation()

- Class: Allocation
- Input: sorted cost array, requirements, availability
- Output: Minimum cost allocation matrix

VerifyAllocation()

- Class: Allocation
- Input: allocation matrix
- Output: final allocation if no conflict and call to GenereateStrategy() if conflict

ReAllocation()

- Class: Allocation
- Input: Probability vector
- Output: allocation matrix based on the reallocation

GenerateStrategy()

- Class: StrategyGenerator
- Input: allocation matrix, cost, conflict resource and crises
- Output: Strategy set for each player

GameMatrixGen()

- Class: Game
- Input: strategy set for each player
- Output: payoff Matrix for each player

recCombiCost()

- Class: Game
- Input: strategy set of a player
- Output: payoffs of all other players for each strategy of input player

Nash_Nplayer_3Competitor()

- Class: NashEquilibrium
- Input: payoff matrix of all the players
- Output: probability vector of nash equilibrium solution

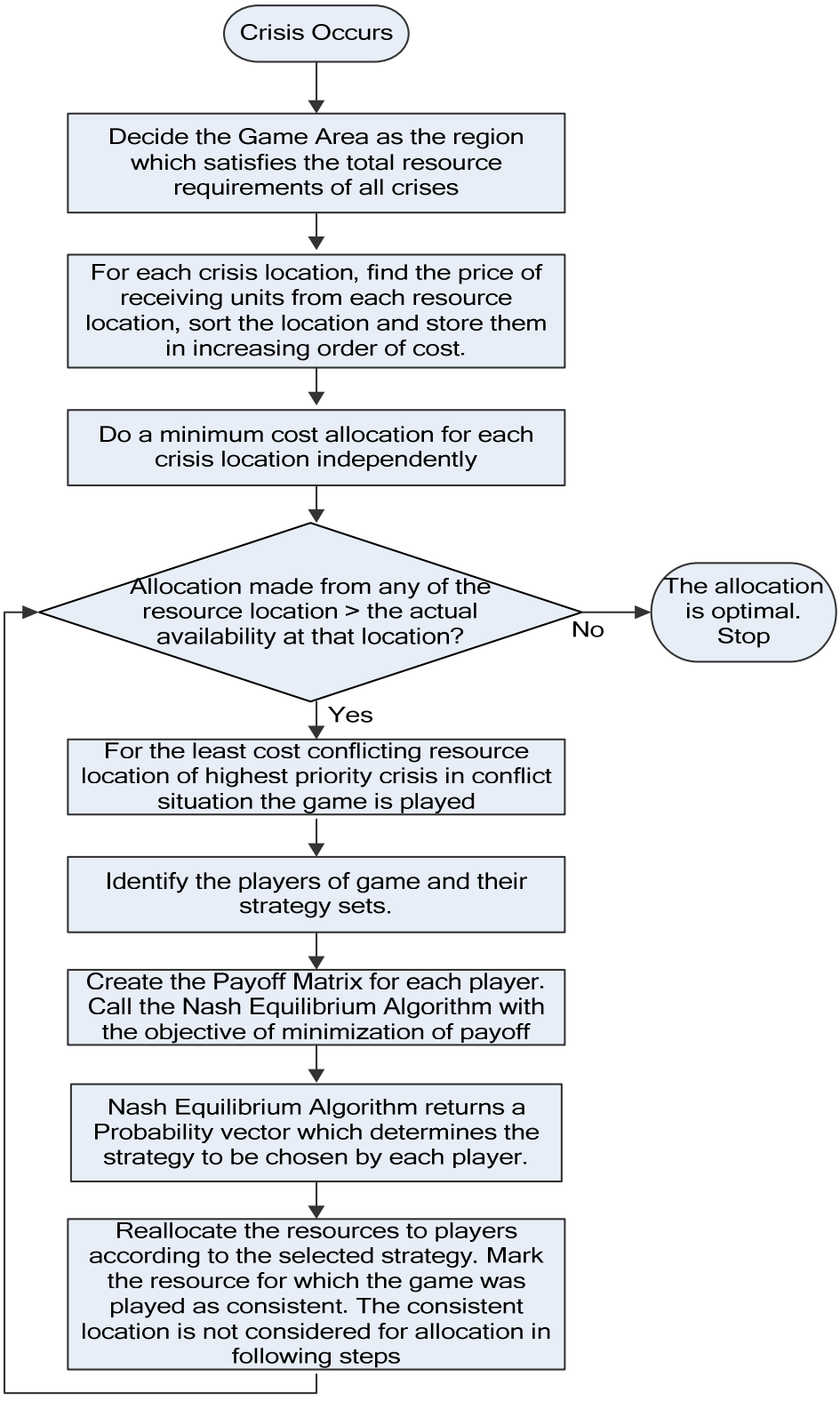

Figure 4.10 Workflow Model of System

In this chapter, the algorithm that has been implemented for optimal allocation of resources to the crises locations in a multiple crisis environment was described in details. A pseudo-code was presented and discussed for each module of the system. Also, the rationale behind using the Terje Hansen algorithm as the Nash Equilibrium methodology was discussed. The last section of the chapter provided a detailed description of the software architecture of the system.

CHAPTER 5

EXPERIMENTAL RESULTS AND ANALYSIS

This chapter presents the experiments that were performed to evaluate the apposition of the proposed game theoretic model for multi-crisis resource management system. Analysis of the experimental results provided some key observations that are vital for the implementation of such real time emergency management systems. The chapter studies some of these observations and their effect on the proposed system.

Response time is the key attribute that determines the practicality of an emergency management system. Experiments were conducted to determine the response time of the system. The effect of each component; number of players, number of resource location and the number of resource units at each location, on the response time of system was studied and is been presented using comparative charts and graphs. The experimental test set consisted of a realistic set of conditions that would exist in a multiple crisis scenario. For example, at any particular time, there maximum of four to five crises with a total requirement of no more than 100 units of resources can occur in a locality.

The rationale behind the choice of game theoretic methodology for multi-crisis resource allocation scenario is the existence of fairness of allocation. A quantitative analysis of fairness of the system was performed and the fairness was compared with the minimum cost allocation methodology. The proposed system outperformed the other methodology. Also, a comparative study was performed for the difference between the total cost of allocation of resources to the highest priority player against the lowest priority player, and the results justified the choice of proposed methodology.

5.1 Experimental Setup

Experiments were conducted on a test set that consisted of at most six players, each player requesting a maximum 60 units of resources with a total system wide requirement of utmost 150 resource units. The number of resource centers in a locality was varied from three to seventeen. The values of the attributes like the distance from source to destination, the average speed on the route and the criticality of the crisis event were randomly generated.

The experiments were executed on a SunOS system with each experimental result being an average of fifty iterations on the same set of players and resources with randomly varying values of distance, velocity and criticality. A total of around 7000 tests were performed and the results were stored in text files. The execution time of the algorithm was calculated using the functions of <sys/timeb.h> library. The specifications of the Sun system are listed in Figure 5.1. Table 5.1 provides a summary of experimental test set.

The results obtained from execution of experiments were graphically analyzed using Microsoft Excel. MS Excel was also use for finding the fitness of the proposed model.

System	SunOS
Node	sunblast
Release	5.9
KernelID	Generic_112233-11
Machine	sun4u
OEM#	0
Origin#	1
NumCPU	8

Figure 5.1 System Details

Table 5.1 Experimental Setup

Number of Crisis Locations	2 – 6
Maximum Request by a crisis	<= 60
Total Request by all crises	<= 150
Number of Resource Centers	3 – 17
Units available at each resource center	>= 12
Test Cases	7014

5.2 Execution Time Analysis

Time of convergence of the Nash Equilibrium algorithm is the most significant component in the total execution time of the game theoretic algorithm. Test cases were executed with permutations of varying number of resource location, resource units and number of crisis locations in order to identify the prominent conditions that contribute towards the execution time of the algorithm.

5.2.1 Effect of Resource Units

In the first test scenario, the algorithm was executed with variable number of resource units at the resource locations. The number of resource locations was kept constant and different series were generated for different number of crises. Each test case was executed fifty times and the average of execution times was calculated. This ensured that a major portion of the possible permutations for allocation were incorporated in the test set. Figure 5.2 shows the graph that was obtained for the test case with number of resource locations as 10 and the total number of units requested by crises as 60.

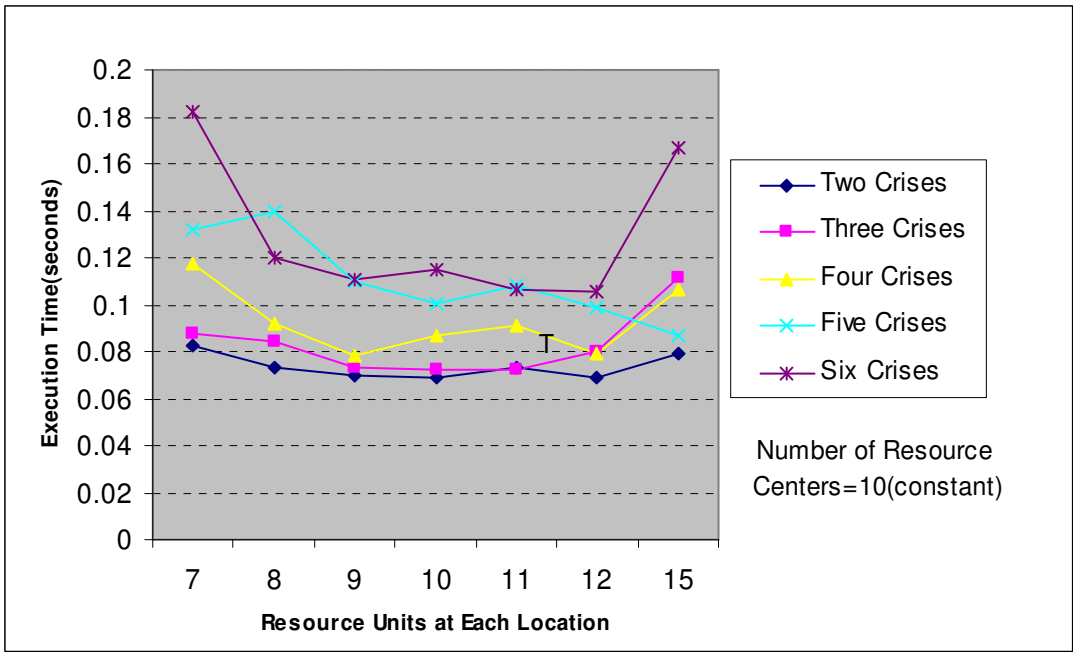

Figure 5.2 Dependence of Execution Time on Number of Resources

5.2.1.1 Observations and Analysis

The graph generated for different number of crises events, followed a particular trend. It was observed that as the number of resource units available at each location increases, the convergence time of the algorithm decreases. This is attributed to the fact that if more units are available at each location, the location can allocate more resources to the players and the contention for resources would be less. But, after a certain limit, the convergence time again increases. This is because if there are too many units available at a resource center and the resource center is the lowest cost center for more than one player, then those players would try to receive maximum available units from that location, thereby resulting in a game with large strategy set. Hence, having a large number of units at resource center could actually attribute towards an increase in the convergence time of the system.

5.2.2 Effect of Resource Locations

The number of resource centers was varied for the second set of test cases. If a game was played with each player's strategy being the number of resources requested from each resource location, the strategy set would have exploded if the number of resource locations were beyond nine locations. The same behavior was expected from the proposed methodology. But, the test cases executed with the number resource centers varying from 3 to 17, keeping the total number of resource units constant, provided some interesting and useful observations about the working of algorithm.

The logarithmic graphs were plotted for the test results because the range of execution time was very wide and there were considerable number of output values within a very small range (0-1 seconds). Figure 5.3 shows the average executions time of the algorithm for varying resource locations. This graph, though shows the behavior of the program, conceals some quantitatively important values of actual execution times in worst case scenarios. In order to study those values, another logarithmic graph was plotted with the peak value of the set of outputs. The graph is displayed in Figure 5.4.

5.2.2.1 Observations and Analysis

As evident from the general trends in the graphs shown in Figure 5.3 and 5.4, the execution time of the algorithm is very high when the number of resource centers is very limited (between 3 to 5). This is due to the fact that when the number of resource centers is very limited, all the crises would have some allocation of units from most of resource centers. This would lead to a series of games that needs to be played for the optimal allocation. An increase in the number of games would lead to an increase in the total execution time of the algorithm. Conversely, as the number of resource locations increase, the sort order for all the players is different and the resources are allocated from different resource centers leading to a lower contention rate and hence a lower execution time.

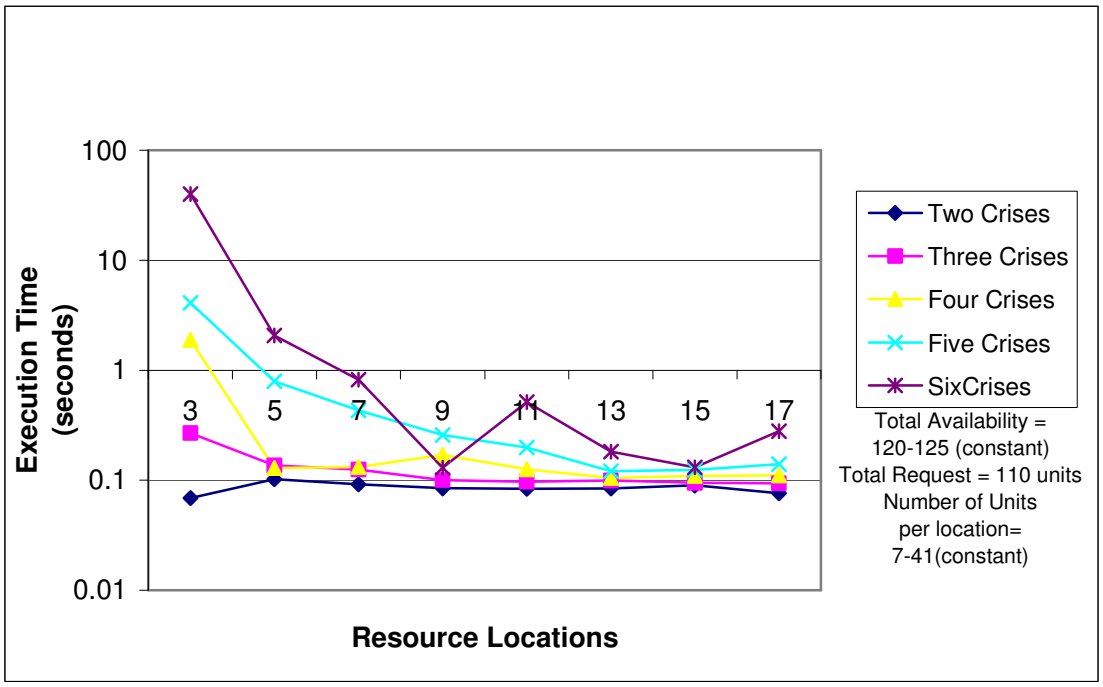

Figure 5.3 Dependence of Average Execution Time on Resource Centers

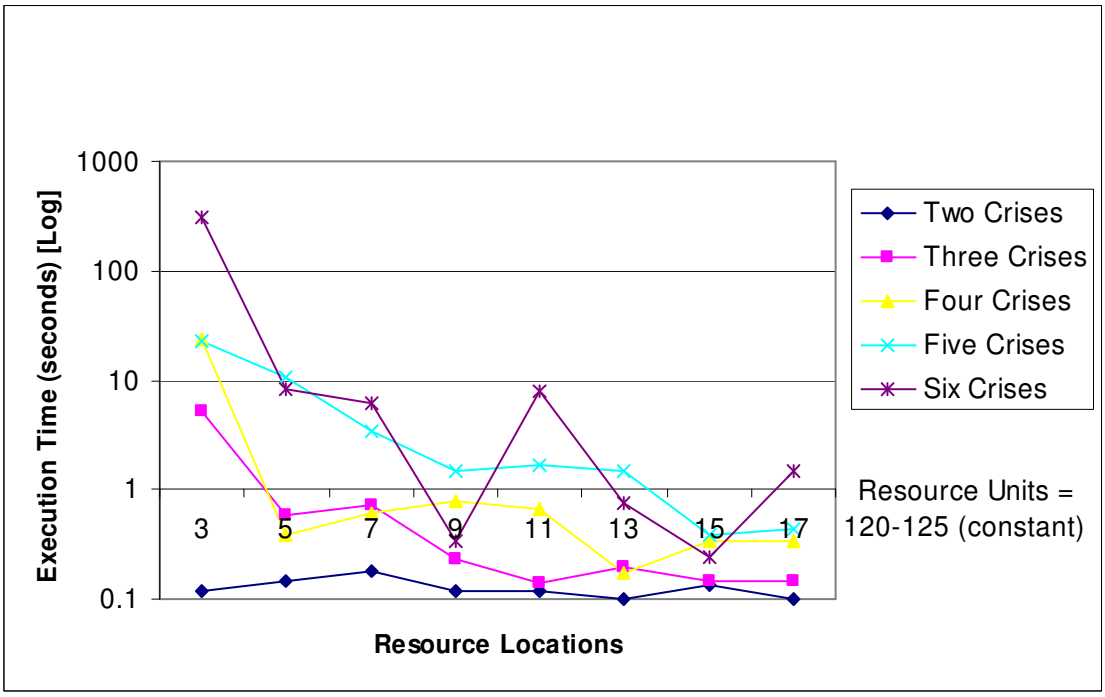

Figure 5.4 Dependence of Peak Execution Time on Resource Centers

5.2.3 Effect of Distribution of Resource Units

Another set of tests were executed on the algorithm to identify the effect of amount of resource availability on the convergence time of the algorithm. Ideally, an increment in the availability of resource units would have resulted in a significant decrease in the convergence time because of the availability of larger set of resource for the crises.

The results generated from the experiments conducted for a four player game with varying number of resource units at each location are described in Figure 5.5.

5.2.3.1 Observations and Analysis

One of the most important observations from this set of experiments was the establishment of the idea that excess availability of the resources does not always ensure a smaller execution time. According to the results, if the total availability of resources is up to 125% of the request, the algorithm converges with a small convergence time (as shown in Figure 5.5). The performance actually degrades in terms of execution time when the availability is 150% or more than the request. This occurs because in some cases in this situation, since the number of units available at each location is more, if more than one player has the same sorted order of resource centers as some of the other players, there will be more resource units in conflict at that location. This would result in a bigger game and hence would take longer to converge.

5.3 Fairness of Solution

The rationale for modeling the multi-crisis resource management system is to provide an optimal allocation of resources to the crises while ensuring the fairness of solution. Qualitatively, fairness is defined as an allocation that is fair for each player of the system as well as for the system itself. Nash Equilibrium solution for a game theoretic implementation provides a socially optimal solution in which each player receives its best utility if other players adhere to the allocation made to them.

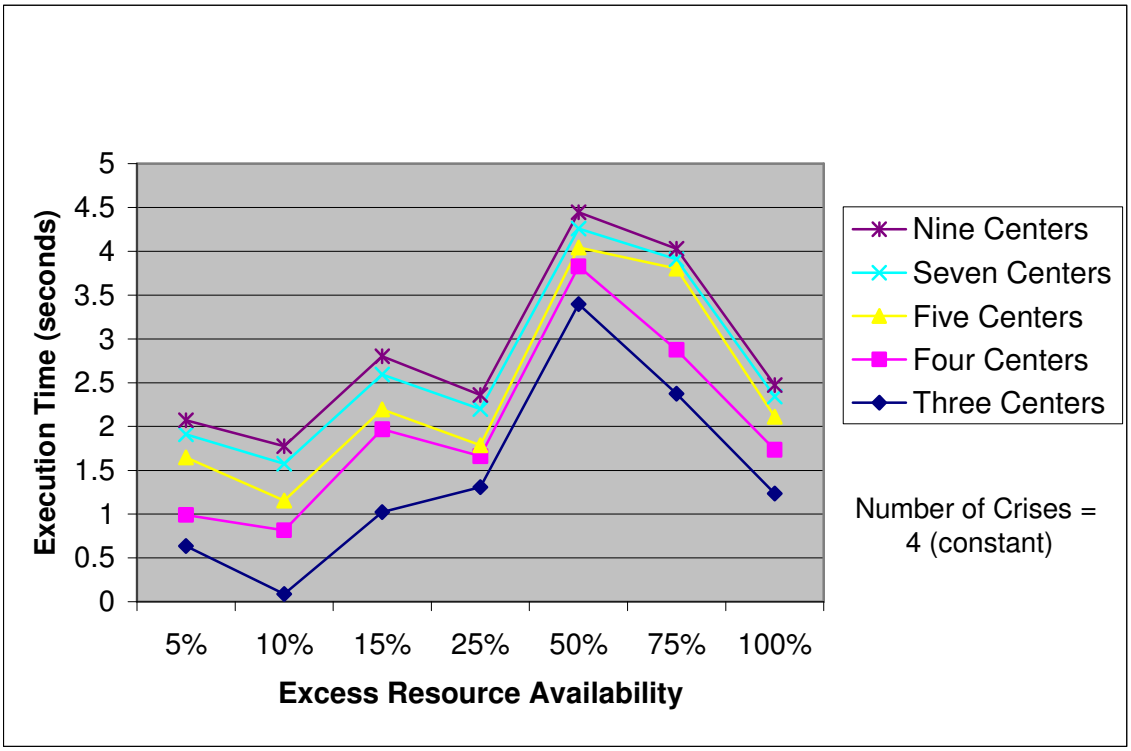

Figure 5.5 Dependence of Execution Time on Excess Resource Availability

The quantitative measure of fairness for a system can be determined by identifying if the system meets certain criteria on throughput and delay. If the system does meet these criteria it is deemed to be unfair. In the scenario of multiple crises, fairness would be determined by whether or not a crisis receives its fair share of resources.

A methodology for quantitative measure of the fairness of resource allocation in domain of computer networks has been discussed in [56]. The methodology identifies the fair proportion of the resources which a user should receive according to its weighting and the proportion which the user actually received, and finds the user's weighted self fairness or unfairness value. These self fairness values are then used to calculate the weighted average fairness of the system.

The concept defined in [56] was used to find the self fairness of each player of the game the multi-crisis scenario and the self fairness values were used to find the average fairness of the system. In this scenario the weighting of each player is identified by its level of criticality and

amount of request. Figure 5.6 describes the methodology in details. The value of average fairness of a system for N users ranges from 0 to 1 inclusive. The maximum value of fairness is unity which is obtained when each player (user) of the system consumes its weighted fair allocation.

The measure of fairness of system in our scenario will be determined by comparing total cost of the actual allocation made to the player by implementing Nash Equilibrium optimization algorithm with the fair allocation that the player should have received according to its level of criticality and total requirements. The quantitative value of system wide fairness was calculated by implementing the methodology being described in Figure 5.6.

The value of fairness was calculated for different sets of parameters; number of players, resource location and the number of units per location and the results are described in Table 5.2

Table 5.2 Fairness Measure

Number of Crises	Resource Locations	Resource Units per Location	Requirements (units)	Average Fairness
3	5	22	100	0.853252
4	5	25	100	0.6976
5	5	23	100	0.953268
6	5	21	100	0.886795
4	7	15	100	0.84533
4	9	12	100	0.793

In all the test cases, the average fairness of the system was in the range of 0.68 – 0.96, which is a good measure of fairness.

The measure of fairness in the game theoretic setup was then compared with the minimum cost allocation methodology where the allocation of resources to the players is done on the basis of their criticality levels. The highest priority players receive the resources in the increasing order of their costs, followed by the lower priority players receiving the remaining

resource units. The comparative results for a scenario with five resource locations are shown in Figure 5.7. The results of the graph establish the fact that the Nash Equilibrium provides a fair allocation of resources.

1. Weighting of Player P_i

   Weighting r_i of the player P_i is defined as the product of required number of units for the player and the reciprocal of its criticality level.

   $$r_i = Z_i * (1/L_i)$$

2. Fair Allocation

   Fair allocation is defined as the proportion of the total allocation which a player should receive. This is defined as the weighting of the player divided by sum of the weightings of all other players.

   $$p_{fair,\,i} = r_i / \sum_{k=1}^{N} r_k$$

3. Actual Allocation

   The actual proportion of allocation to the player P_i is the cost of allocation of resources to that player divided by the total cost of the system.

   $$p_i = c_{final}[p_i] / \sum_{k=1}^{N} c_{final}[p_k]$$

4. Weighted Self Fairness

   Weighted self fairness of each player P_i is defined as:

   $$\Im_i = \log(p_i) / \log(p_{fair,i})$$

   The values of p_i and $p_{fair,\,i}$ are normalized to stay in the range between 0 and 0.3679

5. Normalization Constant

   The normalization constant for the average system fairness is defined as:

   $$\wp_i = \frac{1+1/\ln(p_i)}{1+1/\ln(p_{fair,i})}$$

Figure 5.6 Measure of Fairness

6. Weighted Average Fairness of System

The weighted average fairness of the system is:

$$\overline{\Im} = \frac{r_T \sum_{k=1}^{N} \wp_k p_k \Im_k}{\sum_{k=1}^{N} \wp_k r_k}$$

Figure 5.6 Continued

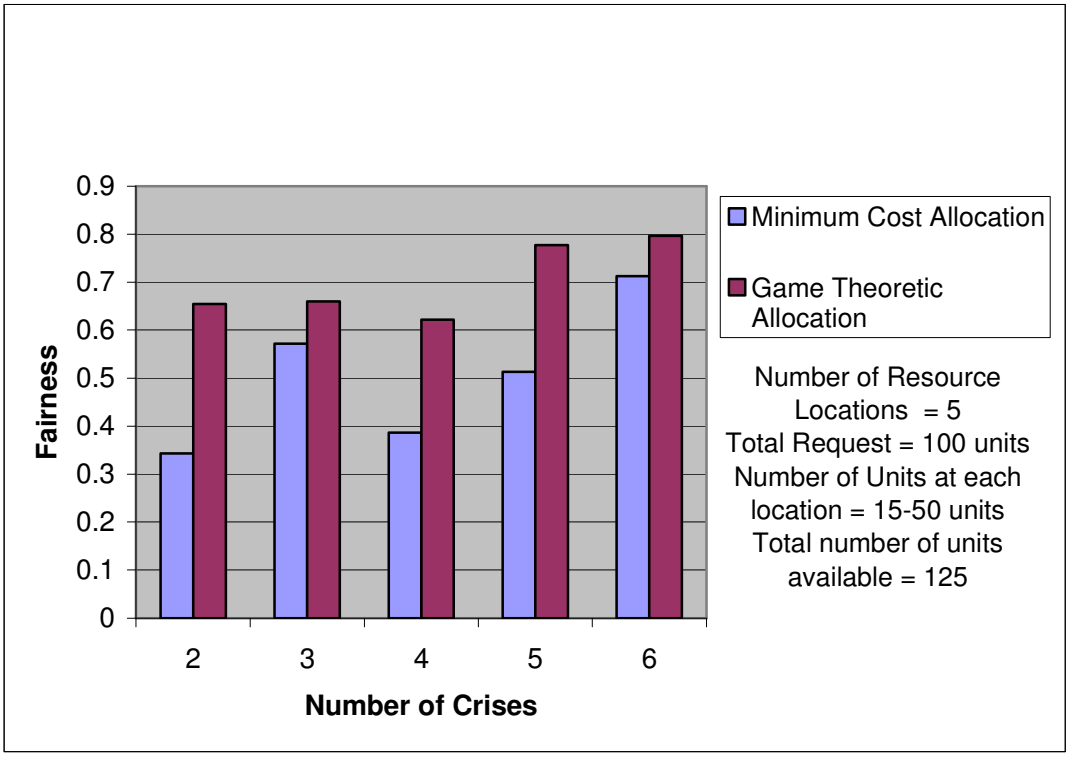

Figure 5.7 Fairness Performances against Minimum Cost Allocation

5.4 Statistical Significance of Results

Regression analysis was performed on the test data to find the statistical significance of the results. Table 5.3 summarizes the results of regression test.

Table 5.3 Regression Analysis Results

	Coefficient of Independence (Log)	P – Value	R Square Value
Number of Crisis Locations (Log)	1.79	0.000279	0.297
Number of Resource Locations (Log)	-1.345	7.75E-05	0.34
Number of Resource Units per Location (Log)	-0.123	0.096	0.014

5.4.1 Coefficient of Independence

The coefficient of independence determines the relationship between the dependent (execution time) and independent (number of crisis locations, number of resource locations and number of resource units at each location) variables. A positive value of coefficient of independence reveals a direct relationship between the dependent and independent variables and a negative value identifies an inverse relationship. In the crisis management scenario, the Number of crises is directly related to the execution time and the number of resource locations and resource units are inversely related. The value 1.1533124 means that if log of the number of crises in the system increases by one unit, the log of execution time of the algorithm would increase by 1.79 milliseconds. Similarly, an increase in log of one resource location or one resource unit at a location would result in decrease of log value of execution time by 1.345 milliseconds seconds and 0.123 milliseconds respectively.

5.4.2 P – Value

The P – Value presents the statistical significance of the results obtained by executing the test set. For the crisis management scenario, the results were found to be significant at 10% confidence level. The number of units per location variable has a P – Value of 0.096, which signifies that there is 90.4% confidence that the test set results are representative of the actual results that would be obtained from the complete solution set. Similarly, for the number of resource locations and the number of players, the confidence is very high.

5.4.3 R Square Value

R Square values determine the fitness of model. The results obtained from the regression test provide an R Square value of 0.297 for the number of crisis location. It means that the crisis location factor governs the model fitness by 29.7%. The remaining 70.3% fitness of the model is governed by the factors other than crisis locations. Number of resource locations and the number of resource units at each location determines 35.4% of the fitness factors.

The chapter presented the relationship between various attributes of a multi-event crisis management scenario being modeled as a game theoretic system. The effect of these attributes on the convergence time of the algorithm was studied and some significant inferences were derived. It was identified that if the availability of total resources is more that 25% above the total requests, the performance of the system may actually degrade. Similarly, the test cases identified that an increase in the number of resource locations actually leads towards a better converge time. The fairness of the methodology was evaluated quantitatively, which fortified the decision of modeling of the system as a game. Finally, a regression test was performed on the experimental test set to ensure that the test set was representative of the actual solution space. It provided a quantitative relationship between the dependent and independent attributes of the system.

CHAPTER 6

CONCLUSIONS AND FUTURE WORK

6.1 Conclusions

In this thesis work, a novel game theoretic approach for multi-event crisis management was proposed. The methodology identifies the crisis events as the players and the emergency response centers as the resource locations available in the system. Each crisis location, according to the severity of the incident, requests for some resources to be allocated to it. In a scenario where multiple crises exist and there is a limited availability of resources, the crises compete for the allocation of resource units. In the proposed approach, each crisis location starts with the minimum cost allocation of resources. If a conflict situation originates at any resource location, a game is formulated for reallocation of conflicting number of units to other locations. Nash Equilibrium optimization algorithm has been implemented for allocation. The rationale behind using Nash Equilibrium algorithm is the fact that it provides a solution which is fair for each player of the system as well as for the system itself. The fairness of solution was quantitatively analyzed and was against the minimum cost allocation methodology. The results endorsed the aptness of proposed game theoretic methodology for multi-event crisis management.

The methodology was tested with a test set of more than 7000 experiments. The analysis of the test results provided important insights into the relationship between the number of resource locations, the crises, number of resource units at each location and the time of convergence of algorithm. A regression test was performed on the test set to establish the statistical significance of the results and it was identified that the test set was representative of the

actual solution space, at a 10% confidence level. The value of coefficient of independence provided a quantitative measure of the dependence for execution time on the independent variables like the number of players and the number of resource locations etc.

The proposed game theoretic framework is the most apposite architecture for multi-event crisis management, because a condition modeled as a game with Nash Equilibrium as the optimization algorithm achieves a social optima by providing fair allocation of resources to all the crises in the system according to their criticality levels in an optimum time.

6.2 Future Work

The future work would focus on generalization of the proposed methodology for a dynamic scenario where the crisis events occurring dynamically would be monitored and serviced in real time. In such a scenario, as soon as a new crisis would occur in a locality, it would be included in the system and the process of allocation of resources would restart from that point. Here, the distribution of resources would change because resources that were already allocated to the crises locations would either be at the source, at the destination or in transit. So, the reallocation of some of those resources may be required according to the criticality of new event.

As the next step of research in this domain, a framework would be developed to enable the algorithm to receive and interpret the various parameter values from a GIS or GPS system in real time, generate an optimal allocation of requested resource units to the crisis locations using our proposed system and alert the response locations to send the allocated units to the crisis locations. A complete automated crisis management system would be realized in this proposed research work.

REFERENCES

[1] Federal Emergency Management Agency, United States of America. URL: http://www.fema.org.

[2] A Nation Prepared, Federal Emergency Management Agency, Strategic Plan. Fiscal Years 2003-2008. URL: http://www.fema.org/pdf/library/fema_strat_plan_fy03-08(append).pdf.

[3] Caltech Infospheres Project, California Institute of Technology. URL: http://www.infospheres.caltech.edu.

[4] Mani Chandy K., Aydemir B. E., Karpilovsky E. M. and Zimmerman D. M. "Event Webs for Crisis Management". *2nd IASTED International Conference on Communications, Internet and Information Technology*. November 2003.

[5] Mani Chandy K., Aydemir B. E., Karpilovsky E. M. and Zimmerman D. M. "Event-Driven Architectures for Distributed Crisis Management". *15th IASTED International Conference on Parallel and Distributed Computing and Systems*. November 2003.

[6] "Provincial Emergency Response Management System Overview (interim), Based on Operations and Management Standard 1000". *Provincial Emergency Program, British Columbia.* ISBN 0-7726-4363-6, September 2000.

[7] Faratin P., Klein M., Sayama H., and Bar-Yam Y. "Simple negotiating agents in complex games: Emergent equilibria and dominance of strategies". *Proceedings of the Eighth International Workshop on Agent Theories, Architectures, and Languages (ATAL-2001).* Pages: 42-53, 2001.

[8] Sandholm, T. "Distributed Rational Decision Making". In the textbook *Multi-agent Systems: A Modern Introduction to Distributed Artificial Intelligence, Weiss, G., ed., MIT Press.* Pages: 201-258, 2000.

[9] Kirkpatrick, S. "Optimization by Simulated Annealing - Quantitative Studies". *J. Stat. Phys*. 34: 975-986, 1984.

[10] McDonald S. and Wagner L. "Using simulated annealing to calculate the trembles of trembling hand perfection". *The 2003 Congress on Evolutionary Computation, CEC '03*. 4: 2482 – 2489, Dec. 2003.

[11] Mutalik P. P., Knight L. R., Blanton J. L. and Wainwright R. L. "Solving combinatorial optimization problems using parallel simulated annealing and parallel genetic algorithms". *Proceedings of the 1992 ACM/SIGAPP symposium on applied computing: technological challenges of the 1990's.* March 1992.

[12] Takahara S. and Miyamoto S. “Adaptive algorithms of metaheuristics with application to optimal allocation problems”. *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*. 3: 545-550, Oct. 1999.

[13] Nara K. and Hayashi Y. “A solution algorithm based on multi-stage tabu search for nested combinatorial optimization problem [dispersed energy storage and generators]”. *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*. 3: 551-556, Oct. 1999.

[14] Lim A., Rodrigues B., Xiao F. and Zhu Y. “Crane scheduling using tabu search”. *Proceedings of the 14th IEEE International Conference on Tools with Artificial Intelligence, (ICTAI 2002)*. Pages: 146 – 153, Nov. 2002.

[15] Rawling G. (ed.) “Foundations of Genetic Algorithms”. *Morgan Kaufmann Publishers.* 1991.

[16] Zdansky M. and Pozivil J. “Combination Genetic/Tabu search algorithm for hybrid flowshops optimization”. *Proceedings of Algoritmy, conference on Scientific computing.* Pages: 230-236, 2002.

[17] Mantawy A.H., Abdel-Magid Y. L. and Selim S.Z. “Integrating genetic algorithms, tabu search and simulated annealing for the unit commitment problem”. *IEEE Transactions on Power Systems.* 14(3): 829-836, Aug 1999.

[18] Sullivan, K.A. and Jacobson, S.H. **“**A convergence analysis of generalized hill climbing algorithms”. *IEEE Transaction on Automatic Control.* 46(8): 1288-1293, Aug. 2001.

[19] Sannomiya N. and Iima H. “Genetic algorithm approach to an optimal scheduling problem for a large-scale complex manufacturing system”. *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*. 3: 622-627, Oct. 1999.

[20] Reuter C., Schwiegershausen M. and Pirsch P. “Heterogeneous multiprocessor scheduling and allocation using evolutionary algorithms”. *Proceedings of IEEE International Conference on Application-Specific Systems, Architectures and Processors.* Pages: 294-303, July 1997.

[21] Sherif M.R., Habib I.W., Nagshineh M. and Kermani P. “Adaptive allocation of resources and call admission control for wireless ATM using genetic algorithms”. *IEEE Journal on Selected Areas in Communications*. 18(2): 268 – 282, Feb. 2000.

[22] Palaniappan S., Zein-Sabatto S. and Sekmen A. “Dynamic multiobjective optimization of war resource allocation using adaptive genetic algorithms”. *Proceedings of IEEE at SoutheastCon 2001*. Pages: 160 – 165, 2001.

[23] Vincent T.L. and Vincent T.L.S. “Evolution and control system design: The evolutionary game”. *Control SystemsMagazine, IEEE.* 20(5): 20-35, 2000.

[24] Beck J.E. and Siewiorek D.P. “Simulated annealing applied to multicomputer task allocation and processor specification”. *Eighth IEEE Symposium on Parallel and Distributed Processing*. Pages: 232 – 239, Oct. 1996.

[25] Zheng Z. and Douligeris C. “Convergence of Synchronous and Asynchronous Greedy Algorithms in a multiclass Telecommunications Environment”. *IEEE transactions on communication.* 40(8): 1277-1281, August 1999.

[26] Stuckman B., Evans G. and Mollaghasemi M. “Comparison of global search methods for design optimization using simulation”. *Proceedings of the 23rd conference on Winter simulation, Phoenix, Arizona, United States.* Pages: 937 – 944, 1991.

[27] Avin C. and Brito C. “Efficient and robust query processing in dynamic environments using random walk techniques”. *Third International Symposium on Information Processing in Sensor Networks, IPSN 2004.* Pages: 277-286, April 2004.

[28] Nowak A. S. and Altman E. “ε-Equilibria for Stochastic Games with Uncountable State Space and Unbounded Costs”. *SIAM Journal on Control and Optimization.* 40(6): 1821-1839, 2002.

[29] Lo V. M. “Heuristic algorithms for task assignment in distributed systems”. *IEEE Transactions of Computers.* 37(11):1384 – 1397, Nov. 1988.

[30] Neumann J. V. “Zur Theorei der Gesellschaftsspiele”. 1928.

[31] John Nash. “Non-Cooperative Games”. *The Annals of Mathematics, 2nd Ser.* 54(2): 286-295, Sep. 1951.

[32] Friedman J. W. “Game theory with applications to economics”. *Oxford University Press*. 1986.

[33] Lucas W. F. “Some Recent Developments in n-Person Game Theory”. *SIAM Review*. 13(4): 491-523, Oct. 1971.

[34] Dutta D., Goel A., and Heidermann J. “Oblivious AQM and Nash Equilibria”. A*CM SIGCOMM Computer Communications Review*. 32(3), July 2002.

[35] Rextin A. T., Irfan Z. and Uzmi Z. A. “Games networks play a game theoretic approach to networks”. *Proceedings of 7th International Symposium on Parallel Architectures, Algorithms and Networks*. Pages: 451-456, May 2004.

[36] Eric Rasmusen. “Games and Information”. *Oxford, OX, UK ; New York, N.Y., B.Blackwell*. 1989.

[37] Game Theory .net, A resource for students and educators of Game theory. URL: http://www.gametheory.net/Dictionary.

[38] EconPort, A Digital Library for Microeconomics Education. URL: http://www.econport.org:8080/econport/request?page=web_home.

[39] McKelvey R. and McLennan A. "Computation of equilibria in finite games". *Handbook of Computational Economics*, Edited by H. Amman, D. Kendrick, J. Rust, Elsevier. Pages: 87-142, 1996.

[40] Lemke C. E. "Bimatrix equilibrium points and mathematical programming". *Management Science.* 11: 681-689, 1965.

[41] Lemke C. E. and Howson J.T. "Equilibrium points of bimatrix games". *SIAM Journal of Applied Mathematics*. 12: 413-423, 1964.

[42] Terje Hansen. "On the Approximation of Nash Equilibrium Points in an N-Person Noncooperative Game". *SIAM Journal on Applied Mathematics*. 26(3): 622-637, May 1974.

[43] Scarf H. E. and Hansen T. "The computation of economic equilibria". *Cowles Foundation Monograph, Yale University Press.*1973.

[44] Garey M. R. and Johnson D. S. "Computers and Intractability: a guide to the theory of NP-completeness". *W.H. Freeman*. 1979.

[45] Vetta A. "Nash equilibria in competitive societies, with applications to facility location, traffic routing and auctions". *The 43rd Annual IEEE Symposium on Foundations of Computer Science.* Pages: 416-425, Nov 2002.

[46] Ibaraki T. and Katoh N. "Resource Allocation Problems". *The MIT Press*. 1988.

[47] Webster Online Dictionary. URL: http://www.webster-dictionary.org.

[48] Wikipedia, the free encyclopedia. URL: http://en.wikipedia.org.

[49] Computer Science and Telecommunication Board, National Research Council, "Summary of a Workshop on Information Technology Research for Crisis Management". *The National Academy Press, Washington D.C.* 1999.

[50] John C. Pine. "Geographical Information Systems in Emergency Management". *A report prepared for the project funded in part by the Federal Emergency Management Agency, Emergency Management Institute, Emmitsburg, MD.* 1998.

[51] Rashed T. and Weeks J. "Assessing vulnerability to hazards through spatial multicriteria analysis of urban areas". *International Journal of Geographical Information Science.* 17(6): 547-576, 2003.

[52] Berfield A., Chrysanthis P. K. and Labrinidis A. "Automated Service Integration for Crisis Management". *Proceedings of the First Workshop on Databases in Virtual Organizations (DIVO 2004) Held in conjunction with the ACM SIGMOD/PODS Conference.* 2004.

[53] Spencer J., Frizzelle B. G., Page P. H. and Vogler J. B. "Global Positioning System: A Field Guide for Social Sciences". *Blackwell Publishing.* 2003.

[54] Gradshteyn, I. S. and Ryzhik, I. M. "Tables of Integrals, Series, and Products, 6th ed". *San Diego, CA: Academic Press*. Page: 1132, 2000.

[55] Corman T. H., Leiserson C. E. and Rivest R. L., "Introduction to Algorithms", *McGraw-Hill Company*, 1990.

[56] Elliott R. “A Measure of fairness of service for scheduling algorithms in multiuser systems”. *IEEE Canadian conference on Electrical and Computer Engineering, IEEE CCECE.* 3: 1583 – 1588, May 2002.