An Application Study of Multi-Agent Systems in Multi-Criteria Ship Design Optimisation

Bekir S. Türkmen, The Ship Stability Research Centre, Glasgow/UK bekir.turkmen@na-me.ac.uk
Osman Turan, The Ship Stability Research Centre, Glasgow/UK o.turan@na-me.ac.uk

Abstract

The overall complexity of optimisation problems in ship design results in high computation time requirements. The high computation time requirements for objective function evaluations in a complex design optimisation problem can practically be reduced by either using parallel execution of the objective functions in state-of-the-art super computers or concurrent execution of the objective functions in a distributed environment. In this paper, the distribution of the function evaluations is chosen to improve the performance of the optimisation algorithm. In the proposed distributed decision support environment, the tool for each objective function evaluation is represented as an agent. Agents are the excellent metaphors for devising a distributed environment, because they are able to encapsulate intelligence and tasks modularly as well as taking into account other agents in the environment. However, agents are not the entities, which do all the things for a person without human intervention. The designer of an agent must take into account all the details of the intelligence that the agent requires during its lifecycle. Furthermore, agents need to communicate in an environment, where the naming and role features are solved in order to allow them to co-operate, co-ordinate and be controlled within a certain extent. Multi-agent systems research focuses on three aforementioned topics basically communication, control and collaboration or negotiation. In this paper, the focus is given more on the communication aspects of multi-agent systems, the collaboration and control of the system is addressed by the multi-objective algorithm and the optimisation agent respectively. The optimisation algorithm in the proposed environment is implemented as a multi-objective genetic algorithm because of its capabilities of finding diverse and near Pareto-optimal solutions. The optimisation algorithm is parallelized in the objective function evaluations part of the NSGA II (A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization) and realized by using a multi-agent systems software environment. Two optimisation studies are presented in the paper. In the first optimisation problem, a simple test function is used for the validation of the distributed optimisation algorithm. In the second problem, an internal hull subdivision optimisation problem from the damage stability point of view is implemented and the results and the efficiency of the distributed version are presented. Finally, as one of future directions, the incorporation of the designer's preferences in the optimisation study is discussed.

1. Introduction

Multi-agent systems open the new era of the research topics in computer science and partially in engineering communities. The research on distributed artificial intelligence is shifted to the multi-agent systems, since both address the same concepts. Multi-agent systems are widely used in application areas ranging from control industry to e-commerce. Introduction of the semantic web and XML (Extensible Markup Language) based knowledge systems have also increased the popularity of the agents and in a broader extent to multi-agent systems.

Agents are the entities, in literal meaning, that act or have the power of the authority to act on behalf of its designer or the user. Agents may vary in the literature, from user interface agents to matchmaker agents. The basic and important features of the agents can be listed as autonomy, proactivity and collaboration, especially when we are designing agents to be used for engineering design. An autonomous agent works in a way that it can have self-activation mechanism, for example if we design an agent to check the tolerance of a mechanical system, the agent should be able to have self-activation behaviour to check the tolerance and report to its user, even if the user has not given the stimuli to check except in the involvement of enabling agent at first place. Another example can be

easily given as simple auto-update mechanism, when there is critical operating system patch available, the agent can download and install the patch with user's proper authorization.

The proactive behaviour is a property for an agent to pursue and act according to its goals. There are different proactive approaches in the literature. The RoboCup is one of the main test cases for proactive behaviour. In RoboCup, agents either physical or virtual players, pursue a goal of scoring goals in football game. Planning and team formation methods are applied extensively in proactive collaborative behaviour.

Collaboration is a very important feature of an agent, which also makes an agent differ from an expert system. Collaboration gives the agent to communicate with other agents in the environment for either satisfying its goals or retrieving information in the environment. A very good example of collaborative behaviour can be seen in the nature. Ants randomly wander in the environment for foraging food. When an ant finds food, it carries the food back to its nest, while producing pheromone and leaving the pheromone to its path from the food location to the nest. Randomly wandering other ants sense the traces of the pheromone and follow the path to reach to the food. This simple collaborative behaviour is also investigated in Mars exploration. A reactive robot (There are three types of agent architecture in the literature, reactive, deliberative and hybrid, for further details, *Wooldridge (2002)*) while exploring in the planet collects the magnetic samples and while returning the samples into the base, it drops some magnetic crumbles in order to find its way back to the sample location as well as informing other robots in the environment. There are other agent properties in the literature. However, we omit them in here for the sake of brevity. Interested readers may refer to *Bradshaw (1997).*

Multi-agent systems combine both above-mentioned and unmentioned agent properties in a framework to make the agents work in a distributed, scalable, dynamic and an open environment. Communication, which is one of the properties of multi-agent systems, can be divided into two categories, semantics and syntax. The syntax addresses the combination of the words in a message among agents. There are few such languages that are proposed in the software community, KQML (Knowledge Query Manipulation Language), *Finin (1994)*, and ACL (Agent Communication Language). The FIPA (Foundation of Physical Agents) organisation created the ACL and encourages the use of ACL wide across in the software systems to enable general communication language among different agent software. Another feature, which may be considered as a more important feature in the communication, is the semantics of the messaging among agents. Semantics can be further divided into two categories, ontology and semantics language. Ontology, unlike the literal meaning, can be explained as the vocabulary of the content of the message in agent communication. Semantics Language is generally based on first order logic and used for information or knowledge queries such as FIPA-SL (FIPA Semantics Language), KIF (Knowledge Interchange Format). See Fig.1 for an example of messaging.

```
(QUERY-REF                  // FIPA-ACL Communicative Act
 :sender  ( agent-identifier  :name TestAgent@BEKIRLAP:1099/JADE :addresses)
 :receiver  (set ( agent-identifier  :name CarDeckAgent@BEKIRLAP:1099/JADE) )
 :content  "((Have (SideCasing :Side-Casing-Length 0.0 :Side-Casing-Width 0.0 :Side-Casing-
Height 0.0)))"              // Content Language, 'Have' actually is encapsulated within SL
 :language  fipa-sl          // FIPA Semantics Language
 :ontology  Deck-Ontology // Side-Casing Length , etc. Concepts and 'Have' Predicate
)                                // are defined in the Deck Ontology
```

Fig.1: A FIPA-ACL and FIPA-SL based Message between TestAgent and CarDeckAgent

Another property of the multi-agent systems is the control of the system. The control can be introduced into three categories, centralized, federated and autonomous. In a centralized system, one

controller agent co-ordinates task and result sharing generally using blackboard like approach. Tasks are published and results are written into the blackboard. In federated architecture, certain agents (facilitator, matchmaker or middle agents) do the message and task routing. The communication among agents happens via facilitators then facilitators direct the message to corresponding agent(s). The last control mechanism is the autonomous system, in which agent directly communicates with other agents directly without any control.

The last property is the collaboration of the agents in a multi-agent system. Collaboration can be accomplished in many ways, planning, auctions, game theoretic, etc. In this paper, the optimisation agent, a multi-objective genetic algorithm, is used to achieve collaboration or negotiation by means of Pareto-optimality.

Multi-agent systems are also used in engineering design. It is first initiated to enable different knowledge bases to perform in collaborative manner, *Cutkosky (1993)*, and extended to include design management features, Procura, *Goldmann (1996)*. For detailed overview of use of multi-agent systems in engineering design and ship design please refer to *Turkmen and Turan (2003)*, *Shen et al. (2001)*. In *Turkmen and Turan (2003)*, a multi-agent system for ship design decision support for conceptual and preliminary design is proposed, Fig.2. In the system, the multi-criteria decision making tools are added for giving decision support to the designer. The addition of decision support agents are new to previous systems, since in previous systems, the general idea is to have autonomous multi-agent systems in order to design almost without designer's involvement. However, ship design is so highly complex and interrelated that such an autonomous system will be utopia.

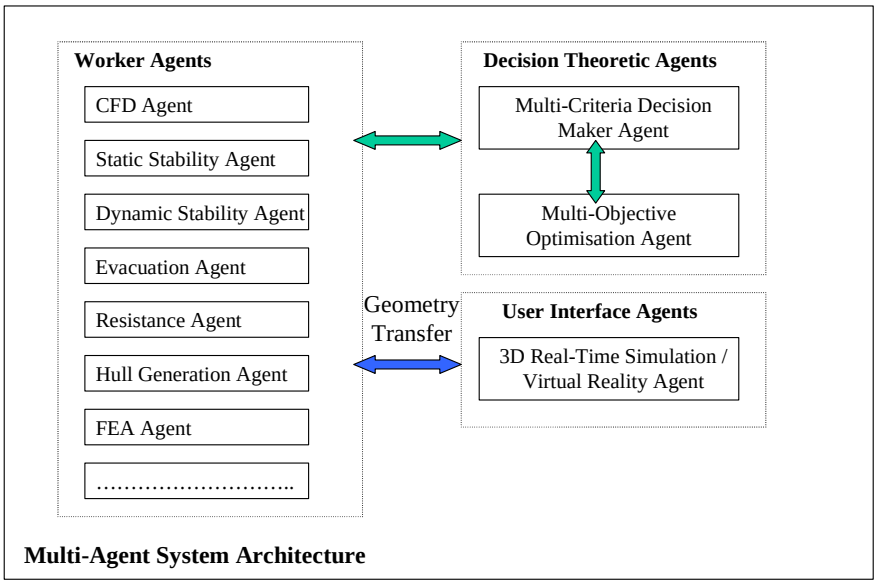

Fig.2: Proposed Multi-Agent System Architecture

In following sections, Multi-objective genetic algorithms, the parallel and distributed NSGAII (A Fast and Elitist Non-Dominated Sorting Genetic Algorithm for Multi Objective Optimization) *Deb et al. (2000)* and the application of the proposed multi-agent system architecture are given for an internal hull subdivision arrangement optimisation problem along with the validation of distributed genetic algorithm with a classical mathematical test function.

2. Multi-Objective Evolutionary Algorithms

The success of finding good Pareto-optimal solutions while maintaining a wide spread of solutions has increased the popularity of evolutionary algorithms in multi-objective optimisation problems. In this study, genetic algorithms are used as an evolutionary algorithm for dealing with multi-objective optimisation problems. Genetic algorithms (GAs) are based on the natural selection, recombination

and mutation concepts for evolving species. Genetic algorithms have the capability of finding global optimum solutions in non-continuous, non-smooth and discrete search domains. The basic definitions for a multi-objective problem are given as following.

Definition 1: Pareto-Dominance Principle

The principle is based on a simple ranking of the solutions by their domination over each other in objective space. Pareto domination can be defined for a minimisation problem as,
For two decision vectors u and v,

u dominates v $(u \succ v)$	iff	$f(u) < f(v)$
u weakly dominates v $(u \succeq v)$	iff	$f(u) \leq f(v)$
u indifferent v $(u \sim v)$	iff	$f(u) \nleq f(v) \wedge f(v) \nleq f(u)$

As can be seen from the definitions above u decision vector dominates v if and only if every f(u) objective is less than f(v).

Definition 2: Pareto Optimality

Pareto optimality is actually the optimal solutions, which are not comparable or indifferent from each other in the concept of Pareto dominance. More rigorously, *Zitzler (1998)* a decision vector $x \in X_f$ (feasible set) is said to be non-dominated (non-inferior) regarding a set $A \subseteq X_f$ iff, $\nexists a \in A : (a \succ x)$.

The Pareto optimal solutions of decision vectors are called Pareto-optimal set, non-dominated (non-inferior) set and the corresponding objective vectors form the Pareto-front. In practical terms, a solution is a Pareto-optimal solution, if you cannot gain an improvement in one objective without causing degradation in another objective. We can divide the use of genetic algorithms into two categories for multi-objective problems: Population and Pareto-based genetic algorithms.

2.1.Population-Based Genetic Algorithms

One of the first studies of genetic algorithms in multi-objective optimisation is devised by *Schaffer (1985)*. In his method, Vector Evaluated Genetic Algorithm (VEGA), the simple genetic algorithm is changed only at the selection stage. In the algorithm, every objective is represented by a subpopulation and in every generation, individuals are selected randomly to create a mating pool. After forming a mating pool, the selection is implemented in the mating pool with respect to chosen objective. The same number of individuals for each objective in the selection is preserved for avoiding any bias into search. After selection, individuals are shuffled randomly and then crossover and mutation operators applied. However, VEGA is still affected by the disadvantages of the weighting approach such as not being able to find to non-convex Pareto optimal fronts.

Hajela and Lin (1992) proposed to encode the weightings of the objectives into chromosomes and tried to evolve them via genetic operations. The reason for encoding weightings into chromosomes can be explained as to find best weighting combinations for the aggregation and to generate more solutions with different weighting combinations. In linearly aggregated approaches, the weighting combination is a very important measure to get the decision maker's true preferences. This method is also affected by drawbacks of the weighted aggregation method, since it uses the weighted sum approach in its core.

2.2.Pareto-based Genetic Algorithms

Most of the algorithms in this section use Pareto-based fitness assignment proposed by *Goldberg, (1989)*. In Goldberg's proposal, firstly non-dominated front is found, and then all the individuals on that front are assigned to a same dummy fitness. For the second front, the first front is ignored and the

dummy fitness values for those individuals in the second front are assigned but this time the dummy fitness values are reduced in order to preserve the non-dominance property of the first front.

Fonseca and Fleming (1993) proposed a different scheme for the fitness assignment, an individual's fitness assigned depending on how many individuals in the population it is dominated by. As a result all the non-dominated solutions are assigned to the same fitness, which is zero.

Horn and Nafpliotis (1993) approached the Pareto-based fitness assignment in a different way. The proposed method NPGA (Niched Pareto Genetic Algorithm) used the tournament selection operator. In NPGA, two individuals are chosen randomly from the population, such as A and B. Moreover, t_{dom} individuals from the current population (t_{dom} >3) are again randomly chosen. The algorithm begins with the comparison of selected individuals A and B with the selected t_{dom} individuals in the tournament set. If B is dominated but A is not, A is selected. If both are dominated or non-dominated, the fitness sharing is implemented. The fitness sharing uses continuously updated sharing, in which niche counts are calculated not by using current population, but rather partly filled next parent population. In other words, the distance metric for the sharing is implemented on the selected individuals for the next parent population.

Srinivas and Deb (1995) applied Goldberg's fitness assignment approach directly with Non-dominated Sorting Genetic Algorithm (NSGA). The fitness sharing is also applied in this method in each front in order to improve the wide spread of solutions. However, the method has disadvantages for its computational complexity.

While the above-mentioned methods are all dependent on a sharing parameter σ_{share}, for fitness sharing, the following methods do not need σ_{share} but they use the similar ideology which is fitness sharing.

SPEA, *Zitzler (1998),* (Strength Pareto Evolutionary Algorithms) is another multi-objective genetic algorithm method. SPEA uses an external set for creating a non-dominated set and adds this external set into selection and fitness sharing process. In SPEA, fitness values of the individuals are defined by its strength. Fitness sharing is implemented by new Pareto-based niching method. In Pareto-based niching method, an individual's strength can be reduced or improved by Pareto dominance principle instead of relative distance among individuals. The algorithm also uses clustering analysis to reduce the number of the non-dominated individuals in the external set not to slow down and unbalance the algorithm's performance.

PAES (Pareto Archived Evolution Strategy) is proposed by *Knowles and Corne (1999).* In this method, one parent and one children evolution strategy is used. "The child is compared with the parent. If the child dominates the parent, the child is accepted as the next parent and the iteration continues. If the child is dominated then the parent is mutated and the iteration is continued. If they cannot dominate each other then the child is compared with an archived external set created by the best solutions. The algorithm then accepts or ignores the child with respect to crowding property of the child", *Deb et al. (2000).* Details of the algorithm are given in *Knowles and Corne (1999).*

NSGAII, a fast and elitist multi objective genetic algorithm, is proposed by *Deb et. al (2000)* and in the algorithm NSGA's fitness assignment is improved with better book keeping algorithm. The algorithm also uses the crowding distance parameter to keep diversity in the population. It also defines the constraint-dominance principle in order to work better with constrained optimisation problems. There are other algorithms based on multi-objective genetic algorithms, however, they are omitted for brevity.

3. Agent Oriented Programming and Distributed NSGAII

Agent oriented programming differs from object oriented programming in a few ways. Agents are built up from behaviours and those behaviours represent mental states of the agents. Behaviours also

govern agents' interaction with the environment and with other agents. For an agent to accomplish all sorts of different behaviours in timely manner, the system should be able to handle different events asynchronously. In the implementation of agents and multi-agent systems in this paper, JADE, *Jade (2004)*] is used for multi-agent software platform. JADE is an agent platform, which comply with FIPA tests for interoperability. JADE agents can be embedded into web pages, as applets or JAVA Server Pages, or they can reside in the local host. JADE can also work in connection with JESS, *Jess (2003)* and Protégé, *Protégé (2004)*, for ontology support.

The optimisation algorithm is distributed and parallelized in the objective evaluations and update parts of NSGAII algorithm. Before going into detail of the software design issues, detailed explanation of NSGAII will be necessary for the background information.

3.1. The NSGA II Algorithm

We can divide NSGA II into three sections. The first one is non-dominated sorting for fitness assignment. The second one, is fitness sharing (crowding distance assignment) and the last section is the binary tournament selection with respect to fitness and crowding distance values.

Non-dominated sorting in NSGA II is a fast and modified version of *Srinivas and Deb's (1995)* proposal NSGA. In this sorting procedure, non-dominated individuals are found in the population with Pareto-dominance principle and assigned to rank zero, then they are ignored to find second front and so on.

Crowding distance assignment is used in order to penalize the individuals that are closer to each other according to fitness sharing. The assignment works in the following way; individuals are sorted with respect to each objective for the optimisation problem in ascending order. The border individuals, the individuals having lower and upper values for each objective function, are assigned to an arbitrary big value. Assigning big values to border individuals is applied in order to preserve the border individuals to find wide spread Pareto-front. Crowding distance values for every objective are then calculated and summed to find a single value for the individual's crowding distance. This procedure is applied for each front.

The third main part is the crowding distance selection. This selection method is a modified version of the binary tournament selection. The only difference in this selection method is that, the method compares crowding distance values of two selected individuals as well as comparing fitness values (ranks, in this case), when two individuals are in the same rank.

NSGA II is an elitist multi-objective genetic algorithm. The NSGA II implements elitism by combining parent and child populations for every generation. The selection of the individuals to create new parent population from two sets is realized by sorting the last front with respect to crowding distance in descending order and selecting the individuals having higher crowding distance values.

3.2. Distributed NSGAII

As mentioned before, for the objective function evaluations and updates we distribute the computational load to other computers in the environment. In order to achieve this, two modifications have been made to the current local version of NSGAII algorithm. The first modification is needed to devise a behaviour for allowing us to distribute the computation among agents and the second modification is the basic loop of generation module in the local version.

3.2.1. The Behaviour Design for Distribution of Objective and Constraint Evaluations

A behaviour must be devised to send the job to the other agents, and wait for the results for certain amount time and inform optimisation agent which embeds NSGAII algorithm. Thankfully, JADE has built-in behaviours for achieving the above-mentioned behaviour. The behaviour in the JADE is

called AchieveREInitiator/Responder which are defined in order to implement FIPA-Request like interaction protocols, such as FIPA-Request, FIPA-Query. This behaviour is used to achieve rational effect, the expected effect of the action. A protocol is used in here in order to keep track of the status of rational effect's achievement, *Bellifemine (2003)*. In the current implementation, the optimiser agent uses the initiator behaviour, but other worker agents use simple behaviour of JADE, which compares the communicative act and sends the result, Fig.3

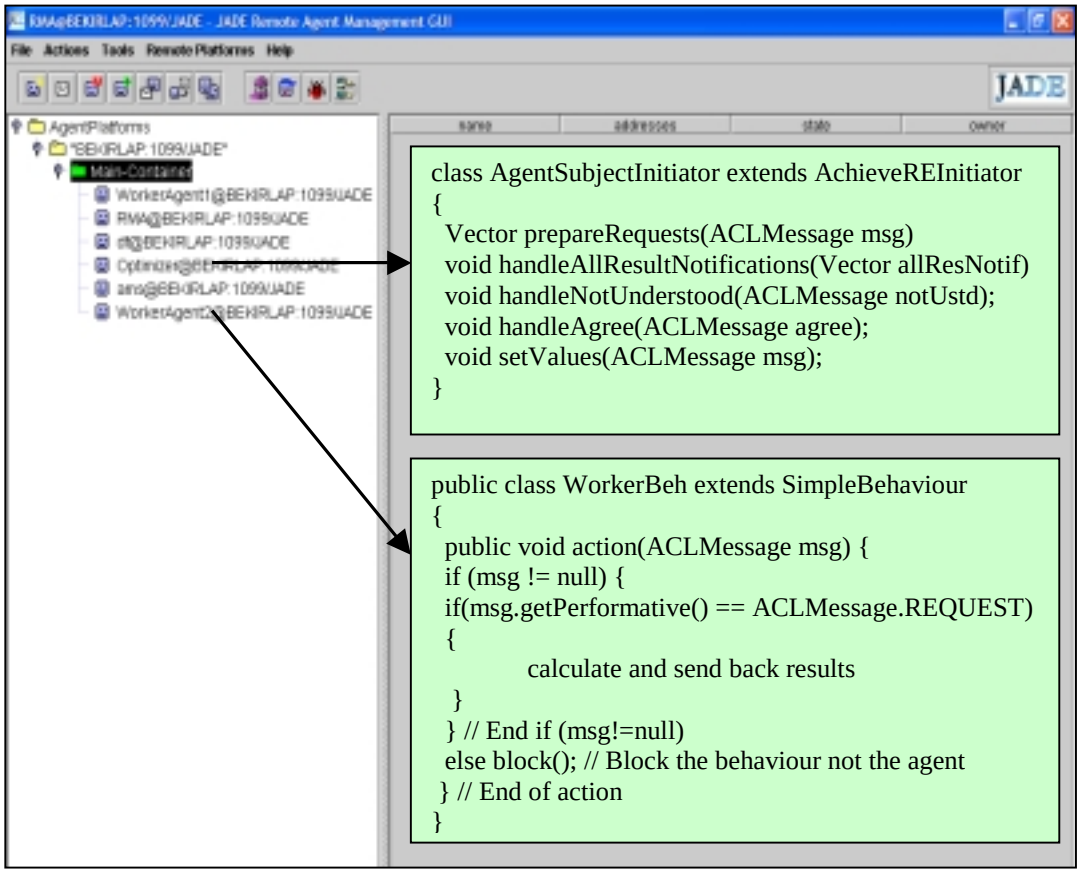

Fig.3: Description of Agent Behaviours and JADE GUI

Other types of behaviour, which need to be added to the agent's behaviour queue are the update and evaluate behaviours for the optimisation algorithm in order to set the objective and constraint function fields of the individuals in the population.

3.2.2. Modified Generation Method for NSGAII

The generation needs to be divided into groups of behaviours and pushed into agent's behaviour queue in order to run the algorithm. In JADE, when the behaviour is done and finished, it is also removed from the behaviour's queue. This results in adding behaviours in every generation. The NSGAII is divided into five main groups, evaluation, first generation, update, second generation and update. The generation behaviour is devised as a sequential behaviour loop. In this sequential behaviour, evaluate and update behaviours are added as parallel behaviours, Fig.4.

There are two missing implementation details in this distributed algorithm. The first one is related to the task sharing, instead of AchieveREInitiator, the more advanced version of Initiator, Contract Net protocol may have been used, and secondly the optimisation algorithm waits for results from every agent to proceed (Granularity problem). Those missing parts can be easily implemented and they will be further detailed in the future research part.

```
// Create generation behaviour
SequentialBehaviour gen=new SequentialBehaviour(myAgent)
{
        onEnd() {
        if(genCounter<nGens-1)  {++genCounter; …}
        else { // clean-up and print out results
        }
}
// Evaluate Objective & Constraint Evaluations
 addEvaluateBehaviours(); // Parallel Behaviours added to the gen, in a sequential
behaviour loop
// First Generation
gen.addSubBehaviour(new RunFirstGenerationBehaviour()); // OneShotBehaviour
// Update ChildInds
addEvaluatebehaviours();
// Run Second Generation
// Loop Through Until Generation Counter Exceeded The Max. Limit
runAndUpdateSecondGeneration(); // Add generation behaviour to the agent's queue
//myAgent.addBehaviour(gen);
```

Fig.4: Generation Behaviour for NSGA II Algorithm

3.3. The Test of The Algorithm with a Classical Test Function

In order to validate the code, the classical Schaffer test function is used. The local code has been already tested in various test functions for validation. The Schaffer Test Function F2, *Schaffer et al. (1985)*, is given below.

$$\text{Min. } f_1(\mathbf{x}) = x^2,$$

$$\text{Min. } f_2(\mathbf{x}) = (x-2)^2, \qquad (1)$$

$$-1000 < x < 1000$$

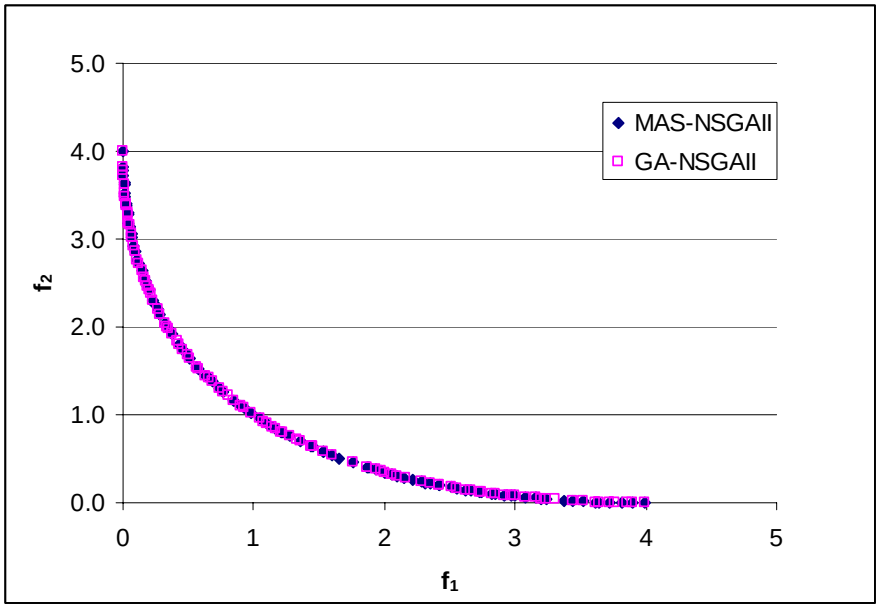

Fig.5: Comparison of Distributed Version with Local Version

The test run parameters for this optimisation problem are as follows; The population parameters for this problem are; Probability of single point crossover, p_c, 0.9, probability of random mutation, p_m, 0.01, and number of generations, nGens, 250 and the population size, nInds, 100. Two agents are created in order to calculate above functions. The agents are executed locally in order to measure the performance. Fig.5 compares results showing that the algorithm works as expected, however, the time took for the calculation in distributed version is increased. (That is already expected, since agents and JADE Agent platform will cause the algorithm to run slower than its local counterpart)

That is also important to note that, the validation of the code is just to ensure that the distributed code works the same way as its local version does. Although, the distributed code uses the local code segments, the behaviour management must be carefully investigated in the software. JADE provides two tools for debugging agent applications, introrespector and sniffer agents. The sniffer agent is used in the implementation of developed software for debugging.

4. Application Study: Internal Hull Subdivision of a Ro-Ro Passenger (ROPAX) Vessel

The optimisation problem for applying the distributed algorithm in ship design is an internal hull subdivision problem for a Ro-Ro Passenger ship (ROPAX). The tragic accidents of the Herald of Free Enterprise in 1987 and the Estonia in 1994 initiated a significant surge of research related to the capsizing of Passenger Ro-Ro type of ships, which have a very large undivided car deck. Flooding of that undivided car deck even with a small amount of water caused rapid capsize of the ships and loss of significant number of lives as experienced with above-mentioned tragic accidents. This effort culminated in significant developments that helped the ferry industry to raise safety levels to demanding new heights. New regulations require all the ROPAX Vessels to comply with SOLAS' 90, *SOLAS (2001)*, standards as well as new water on deck standards known as the Stockholm Agreement, which determines the limiting wave height at which ROPAX vessel survives in damaged condition. In response to the strict new regulations, the shipping industry is demanding new modern Passenger Ro-Ro designs with very high standards cost effectively.

Furthermore, due to recent European Union (EU) regulations, which banned the tax-free sales during the journeys between EU countries, forced designers and operators to look for design and route alternatives to meet customer demands in line with high safety standards as well as cost effectiveness. The end of tax-free sales created a surplus store space in those designs, which cannot be used for passenger accommodation due to safety regulations. The main aim of this case study is to maximise the survivability and stability standards while improving the cargo capacity by introducing new design alternatives. The internal arrangement of the hull is a very important factor for a ship's damage stability and survivability, especially when damage occurred. The optimisation problem is therefore limited to arrangement of transverse bulkheads, car deck height, lower hold height, and lastly side casing, which is known to be important parameter for increasing stability and survivability, Fig.6. The problem described here is a successive work of *Olcer et al. (2003)* and objectives of this optimisation problem are given in Table I. The optimisation variables are also given in Appendix A.1.

Table I: Optimisation Objectives

No	Objectives	Type	Description
1	H_S value	Maximisation	for the worst two compartment damage case
2	KG limiting value	Maximisation	for the worst two compartment damage case
3	Cargo capacity	Maximisation	expressed in car lanes, lorry lanes kept constant

There are three objectives in this study. The first objective is the maximisation of Hs (Significant wave height), which is a measure used to assess survivability of ships in waves. Static Equivalent Method, *Vassalos et al. (1996), Tagg and Tuzcu (2002),* is deployed here in order to calculate Hs.

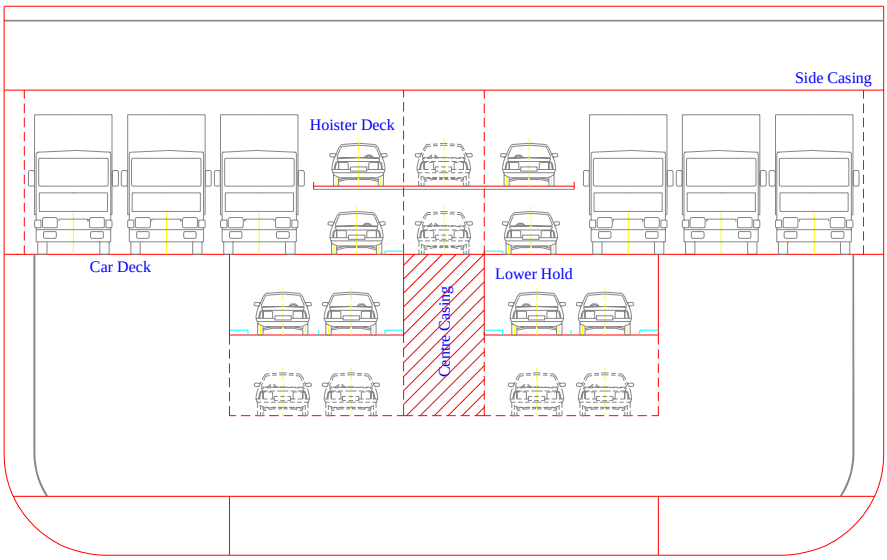

Fig.6: ROPAX Ship Sectional Representation

Another objective is to achieve the right stability standards by achieving limiting KG (vertical centre of gravity) as high as possible. Limiting KG is a way of expressing allowable loading condition of a ship. It is a crucial fact that limiting KG must be higher than the KG value for a loading condition (operational KG) in order to achieve enough stability. Operational KG is a very important parameter for a ship, since it tends to increase during the life cycle of the ship and that causes changes in loading condition. The higher limiting KG values enable designers to make design changes during the life cycle of a ship. The third objective is earning capacity of the ship, which is defined as the number of car lanes in the ROPAX ship.

There are also constraints in this optimisation problem in order to comply with SOLAS'90 Regulations, such as minimum GZ Area, and maximum heel. The minimum bulkhead distance for two adjacent bulkheads must be greater than $0.03.L_s$ +3.0m. We applied this approach in order to comply with SOLAS'90 regulations with no additional calculation cost. L_s is the subdivision length defined in the SOLAS' 90 regulations. Another important constraint in the case study is the operational constraint, which is used to satisfy operational KG requirements in the life cycle of the ship. The tabulated form of all constraints is given in Appendix A.2.

The constraint approach for minimum bulkhead distance is modified in order to make the search more robust. The approach applied here sums all violating bulkhead distances for an individual (string or inputs for optimisation problem in genetic algorithms) and NSGAII with constraint violation penalizes the individuals, which violate the bulkhead distance constraint more.

4.1. Representation of Results

Two different runs are carried out, the first one is the local version and the second is the one with multi-agent systems based distributed genetic algorithm. Same parameters are used in both runs. The population size, number of individuals in the population, is set to 80 and the number of generations is set to 32. Real coding is used as a chromosome representation. Although, the real coding of chromosome representation is used, the crossover and mutation operators are selected as one-point crossover and random mutation. The reason for using crossover and mutation operators, which are not generally suitable for real coding, was due to the problem's combinatorial property. The transverse bulkhead positions along with side-casing width and lower hold height are discrete and that is why, the problem can be seen as a combinatorial problem. One can argue that, using multi-point crossover may perform better in this kind of problems by better preserving building blocks. However, the aim is here to show the multi-agent systems based distributed genetic algorithm, and also we strongly

believe that, even if the multi-point crossover used, the best individual found would not differ too much, due to problem's nature.

In Table II, the best individuals found in both runs are shown. The difference between the best-found individuals in both runs is due to the random features of the genetic algorithms. The selection of the design will of course be the best design found from MAS-NSGAII.

Table II: The Original Design and Found Best Designs

No	Optimisation Variables	Original Design	GA-NSGAII	MAS-NSGAII
1	Car deck height	9.7m	9.9	9.9 m
2	Side-casing width	No side-casing	1	1 m
3	Lower-hold height (from car deck)	2.6m	5.2	5.2 m
Watertight Transverse Bulkheads		In frame numbers	In frame numbers	
4	Transverse bulkhead 02	27	29	26
5	Transverse bulkhead 03	39	41	38
6	Transverse bulkhead 04	51	52	49
7	Transverse bulkhead 05	63	65	65
8	Transverse bulkhead 06	81	83	83
9	Transverse bulkhead 07	99	97	98
10	Transverse bulkhead 08	117	115	115
11	Transverse bulkhead 09	129	127	127
12	Transverse bulkhead 10	141	143	140
13	Transverse bulkhead 11	153	155	151
14	Transverse bulkhead 12	165	166	164
15	Transverse bulkhead 13	177	177	179
16	Transverse bulkhead 14	189	190	191
Performance scores of designs				
1	Cargo capacity expressed in car lanes	8 (abt.1033 m.)	14 (abt.1450) m	14 (abt.1450) m.
2	H_S value	4.641 m	5.443 m	5.908 m
3	KG limiting value	13.845 m	14.048 m	14.044 m

The time required for the algorithm is reduced about 40 % percent. The question may arise why the reduction was not 50 %, the answer to this question is two folds. The first one is that the algorithm runs slower than its counterpart due to communication and Jade's involvement, and secondly, the distributed algorithm has a granularity problem as mentioned before.

5. Discussions and Conclusions

The paper represented a multi-agent systems based distributed genetic algorithm in a master/slave parallel genetic algorithm. The algorithm can be further improved with investigating on island model genetic algorithms to reduce the communication needs, however there could be other problems such as selecting migration rate or how many times the migration will be implemented. The algorithm has so called granularity problem, but we strongly believe that the algorithm may be improved with better scheduling with little effort.

The use of multi-agent systems can also be further improved by using contract net approach, for task and result sharing, and furthermore, Directory Facilitator (In JADE, agents can register themselves to the directory facilitator, with their service description, the semantics language and so on, in other words, the directory facilitator works as a yellow page service) may be used to find the agents in the environment in order to make optimiser agent more dynamic in changing environments.

The selection of the best individual is performed by designer's preferences, to select among almost 10~ 15 near Pareto-optimal solutions. The designer's preferences are valuable ingredient for the optimisation especially for engineering design problems, having time and resource bounds. The introduction of designer's preferences may have been introduced in three categories. The first category, a priori, such as the weighted aggregation techniques or posteriori as the way it has been introduced for the solution of this problem, or progressive, the algorithm can be guided during the optimisation with changing designer's preferences. The third approach, progressive designer's involvement, is obviously the best choice among three, however when there is an extensive need for designer's involvement may make the method discouraging and tedious to use. An intelligent agent which will be acting on behalf of designer in order to guide search very important asset to the solution of the problem.

In conclusion, a multi-agent systems based genetic algorithm is implemented and implementation details are outlined. The algorithm performed well both in mathematical test function as well as ship design problem. The original ROPAX design is improved for survivability and operational parameters, operational KG and Car- lane length. Increasing the lower hold height has also enabled to load truck's into the lower hold, which enables the ship to work on different load conditions.

References

BELLIFEMINE, F.; GIOVANNI C.; TIZIANA T.; GIOVANNI R. (2003), *Jade Programmer's Guide,* http://sharon.cselt.it/projects/jade/

BRADSHAW, J. M., Editor (1997), *Software Agents,* The AIAA Press

CUTKOSKY, M.; ENGELMORE, R.; FIKES, R.; GRUBER, T.; GENESERETH, M.; MARK, W. (1993), *PACT: An Experiment in Integrating Concurrent Engineering Systems* IEEE Computer, special issue on Computer Supported Concurrent Engineering, 26/1, pp.28-37

DEB, K.; AGRAWAL, S.; PRATAB, A.; MEYARIVAN T. (2000), A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA II, KanGAL Report, 200001, Indian Institute of Technology, Kanpur, India

FININ, T.; FRITZON, R.; MCKAY, D.; R MCENTIRE. (1994), *KQML as an Agent Communication Language*, Proceedings of 3rd Int. Conf. on Information and Knowledge Management, ACM Press

FONSECA, C. M.; FLEMING, P. J. (1993), *Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization*, Proceedings of The Fifth International Conference On Genetic Algorithms, Schaffer, D. Editor,Morgan Kaufmann, San Mateo pp. 416-423

GOLDBERG, D. E. (1989), *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison Wesley, Reading, Massachusetts

GOLDMANN, S. (1996), *Procura: A Project Management Model of Concurrent Planning and Design*, Proc. WETICE-96, IEEE Computer Soc. Press, pp. 177-183

HAJELA, P.; LIN, C-Y. (1992), *Genetic search strategies in multicriterion optimal design*, Structural Optimization**,** 4, 99–107

HORN, J. and NAFPLIOTIS, N. (1993), *Multiobjective Optimization using the Niched Pareto Genetic Algorithm*, Technical Report, IlliGAL Report, 93005, University of Illinois,USA

JADE (2004), *Java Agent Development Environment*, http://sharon.cselt.it/projects/jade/

JESS (2003), *The Expert System Shell for the Java Platform*, http://herzberg.ca.sandia.gov/jess/

KNOWLES, J.; CORNE, D. (1999), *The Pareto archived evolution strategy: A new baseline algorithm for multiobjective optimisation*, Proceedings of the 1999 Congress on Evolutionary Computation, pp. 98-105

OLCER, A.; TUZCU, C.; TURAN, O. (2003), *Internal Hull Subdivision Optimisation of Ro-Ro Vessels in Multiple Criteria Decision Making Environment*, Proceedings of The 8th International Marine Design Conference, pp. 339-351

PROTÉGÉ (2004), *Ontology Editor and Knowledge Acquisition System*, http://protege.stanford.edu/

SCHAFFER, J. D. (1985), *Multiple objective optimization with vector evaluated genetic algorithms*, In J. Grefenstette (Ed.), Proceedings of an International Conference on Genetic Algorithms and Their Applications, Pittsburgh,PA, pp. 93–100

SHEN, W.; NORRIE, D.H.; BARTHES, A.J. (2001), *Multi-Agent Systems for Concurrent Intelligent Design and Manufacturing*, Taylor&Francis

SOLAS Consolidated Edition (2001), International Maritime Organization, London

SRIVINAS, N.; DEB, K. (1995), *Multi-objective function optimisation using non-dominated sorting genetic algorithms*, Evolutionary Computation, 2, pp. 221-248

TAGG, R.; TUZCU, C. (2002), *A Performance-based Assessment of the Survival of Damaged Ships - Final Outcome of the EU Research Project HARDER*, Proceedings of the 6th International Ship Stability Workshop, Webb Institute

TURKMEN, B. S.; TURAN O. (2003), *Multi-Agent Systems in Ship Design*, Second Int. EuroConf. on Computer and IT Applications in the Maritime Industries COMPIT'03, Hamburg, pp. 459-472

VASSALOS, D.; PAWLOWSKI, M.; TURAN, O. (1996), *A theoretical investigation on the capsize resistance of passenger Ro/Ro vessels and proposal of survival criteria*, Final Report, Task 5, The NorthWest European R&D Project

WOOLDRIDGE, M. (2002), *An Introduction to MultiAgent Systems*, John Wiley & Sons

ZITZLER, E. (1998), *Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications*, PhD Thesis, Swiss Federal Institute of Technology, Zurich

Appendix A

Table A-I: Optimisation Variables

		Bounds		
No	Variables	**Lower**	**Upper**	**Increment**
1	Car deck height	9.6m	9.9m	0.025m
2	Side-casing width	1m	2m	0.5m
3	Lower-hold height (from car deck)	2.6m	5.2m	-
4	Transverse Bulkhead 02	25	29	1
5	Transverse Bulkhead 03	37	41	1
6	Transverse Bulkhead 04	49	53	1
7	Transverse Bulkhead 05	61	65	1
8	Transverse Bulkhead 06	79	83	1
9	Transverse Bulkhead 07	97	101	1
10	Transverse Bulkhead 08	115	119	1
11	Transverse Bulkhead 09	127	131	1
12	Transverse Bulkhead 10	139	143	1
13	Transverse Bulkhead 11	151	155	1
14	Transverse Bulkhead 12	163	167	1
15	Transverse Bulkhead 13	175	179	1
16	Transverse Bulkhead 14	187	191	1
Bounds for transverse bulkheads are given in frame numbers				

Table A-II: Optimisation Constraints

No	Constraints	**Requirements**
1	Range	Range of positive stability minimum of 15 degrees
2	Min. GZ Area	Minimum area of GZ-curve at least 0.015mrad
3	Maximum GZ Taking into account the greatest of those heeling moments	1. The crowding of all passengers towards one side 2. the launching of all fully loaded davit-launched survival craft on one side 3. due to wind pressure GZ (in metres) = $\frac{heelingmoment}{displacement} + 0.04$ However, in no case is this righting lever to be less than 0.1 m.
4	Maximum Heel	Maximum static heel not more than 12degrees (two adjacent compartment)
5	Minimum GM	Minimum GM (Metacentric Height) at least 0.05m
6	Margin Line	Margin line should not be immersed
7	Bulkhead distance	Minimum bulkhead distance ($0.03*L_{bp}+3m$)
8	KG Operational	Limiting KG greater or equal to 1.005 of Operational KG