A new method for searching optimal path on a raster plane including cost of direction changes

Rafał Szłapczyński
Gdańsk University of Technology

ABSTRACT

The article introduces a new algorithm for finding optimal routes on raster planes. This method takes advantage of the new data structure and results in minimizing the number of direction changes within a route. It has linear time and space complexities and is sufficiently fast to perform real-time routing on the raster grids. Both the algorithm and its data structure are presented in detail in the paper. Possible applications of this method are also discussed.

Keywords : navigation, optimal route, turn penalties, Lee's algorithm, vave propagation, raster charts

1. INTRODUCTION

Raster grids are a digital representation of planary or spatial data, that is currently in use in many fields some of which belong to the maritime industry. Examples of application include pipeline and airduct design as well as navigational raster charts. Although the paper is focused on the navigational use of raster grids, its results may be easily adapted elsewhere.

The technology of electronic navigational charts (ENC) and digital electronic maps (DEM) has been used increasingly in maritime navigation, GPS and GIS systems [4]. As the raster charts are cheaper to be produced than vector ones they cover the majority of the electronic charts currently in use. In the ENC system one of the most common application is to determine an optimal route between a start cell and a destination cell, that does not cross any obstacles (landmass, barriers, shoals). The algorithm performing this should fulfill certain conditions. Of these the most obvious is that it does indeed find an optimal route if only such exists. Other conditions are those of needed time and memory space. In reality only algorithms of linear time and space complexities are useful as only such may be accepted by a real-time system processing large numbers of cells. The big *O* notation is further used for complexities, with *O(n)* indicating linear complexity [2].

The first solution to meet the above mentioned conditions was the maze routing algorithm presented by Lee [3], often described as a wave propagation process. To date, Lee's algorithm and its variations are among the most widely used routing methods finding applications in maze games, VLSI design and road map routing problems. However, the original algorithm proposed by Lee has one serious drawback: it works only for the 2-geometry grid plane (also known as the Manhattan geometry). Only recently it has been upgraded to higher geometries while sustaining the linear time and space complexities. This new solution has been proposed by Chang, Jan and Parberry [1].

Despite the major progress, the potential use of the improved Lee's algorithm has still some limitations. Both the original algorithm and its upgraded version tend to find the shortest path which is not always identical to the optimal one. In presence of many obstacles the algorithms determine a route containing so many turning points and course alterations that no navigator would be able to follow it even being so advised by a real-time routing system. Furthermore over larger geographic areas both distance and number of turns will contribute to the total time spent traversing a tour. Thus minimizing the number of turns is a desirable objective. Although the Chang, Jan and Parberry algorithm may cover the aspect of varied terrain (among other improvements) its data structure, based on that of the original Lee's method, makes it unable to include the cost of a turn in path length.

There is a number of methods (invented mostly for VLSI and automation purposes) that cover the problems of minimum bend path and shortest minimum bend path. Unfortunately, determining the shortest minimum bend path is of no value for long distance sea routing. Instead, the objective is finding the shortest paths with bend penalizing. The bend penalizing issue has been also considered in a number of works but the methods presented there are either not sufficiently fast or not applicable for higher geometries.

The article proposes a solution to this problem. A new data structure has been so designed as to reflect the cost of all course alterations in each cell's arrival time. An algorithm utilizing such structure has been implemented. The algorithm uses the following input parameters: user specified values of the costs of time of course alterations (turn penalties). It returns the determined path.

In Sec. 2 the original Lee's method and its version upgraded by Chang, Jan and Parberry are described in detail. Sec. 3 presents a new algorithm that overcomes their limitations. In Sec. 4 some possible applications of this new solution are discussed, and finally the conclusions are presented.

2. THE MAZE ROUTING ALGORITHM : PAST SOLUTIONS

To facilitate checking out the Chang, Jan and Parberry version of the routing method its original notation is used here. In Sec. 3 the same notation is also used for the description of the new solution.

Lee's algorithm for the 2-geometry grid plane

The popularity of this method lies in its simplicity and the guarantee to find the shortest path if only such exists. Since it is widely known no formal description is given here.

Data structure

The cell map static data is stored in an array containing two fields for every cell :

SL – Boolean field whose value indicates whether a cell is assigned to sea or landmass (or any other static obstacle). *True* for sea, and *false* otherwise.

AT – Integer number field whose value is this cell arrival time, i.e. time needed to travel from the source cell to this cell.

The dynamic data are stored in two lists : L_1 and L_2 containing cell pointers or indexes in the cell array.

Algorithm overview

A cell array is initialized with appropriate *SL* and *AT* values. Two lists L_1 and L_2 are defined to keep track of the cells on the front of the wave (frontier cells) and their equal-distance-step neighbouring cells, respectively. Putting the source cell in the list L_1 initializes the search. After all neighbouring cells of L_1 are included into the list L_2, the list L_2 is processed, so that an expanded wave front is found. Then every cell of L_1 is deleted if all of its neighbouring cells are processed (updated) and L_1 is updated by this new front of the wave. The search is terminated when the destination cell is found or – in the case of the barrier of obstacles – there are no more new cells to be processed, i.e. the last front of the wave has not generated any neighbouring obstacle-free cells.

The Lee's algorithm version upgraded by Chang, Jan and Parberry

This method expands 2-geometry routing to any higher geometry. Details for an example 4-geometry routing are presented beneath. Varied terrain aspect is not considered here as it increases the number of necessary lists, though it is generally supported by this version.

2-geometry and 4-geometry movement rules

Possible moves from a cell on the 2-geometry and 4-geometry grid planes are shown on the diagrams below.

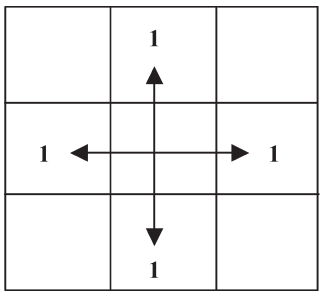

Fig.1. A 2-geometry neighbourhood

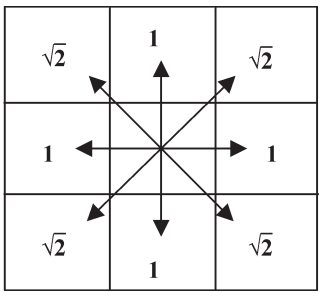

Fig.2. A 4-geometry neighbourhood

Data structure

The static data is stored in an array containing three fields for every cell :

SL – Integer number field whose value indicates whether a cell is assigned to sea or landmass (or any other static obstacle). Its value is 1 for sea, and infinity for a landmass.

AT – Floating point number field whose value is this cell arrival time – time needed to travel from the source cell to this cell.

VIS – Boolean field whose value indicates whether a cell has been already visited (inserted to *temp-list*) or not. *True* for a visited cell, *false* otherwise.

The dynamic data is stored in three circularly used lists: L_1, L_2, L_3 and one temporary list (*temp-list*). The index of a list currently in use is stored in a *bucket_index* variable. All lists contain cell pointers or indexes in the cell array.

Algorithm overview

There are some major differences between the 2 - and 4-geometry algorithms. Firstly, additional lists are introduced to control the propagation speed in different directions (the additional two diagonal directions have single-step distance equal to $\sqrt{2}$ instead of 1).

Secondly, the *VIS* cell field is introduced to make sure that each cell is inserted into temporary list exactly once. These two aspects make the algorithm keeping the original linear complexities.

Algorithm formal description

- The INSERT procedure inserts the index of c_i into the *temp-list*.
- The procedure CLEAR empties any given list.
- The RETRACE procedure retraces from the destination cell to the source cell according to their smallest *AT* value and builds a path LL_{path} which is the output of the 4-geometry maze router.

Algorithm (in Pascal) of 4-GEOMETRY-ROUTER (*Cell-map*, *S*, *D*, LL_{path})

```
Input: Cell-map, S, D
Output: LL_path
begin
 bucket_index : = 0 ;
 L_bucket_index : = S ;
 VIS_s : = TRUE ;
 temp-list : = Ø ;
 path-exists : = FALSE ;
 while (L_bucket_index ≠ Ø) or (L_bucket_index+1 ≠ Ø) do
  if (D cell in L_bucket_index) then
    begin
     path-exists : = TRUE ;
     break while ;
    end
  for each cell c_i in L_bucket_index do
    begin
     for each cell c_j neighbouring c_i do
       begin
        if (SL_j = 1) then
          begin
           if (VIS_j = FALSE) then
             begin
              VIS_j : = TRUE ;
              INSERT(c_j , temp-list) ;
             end
           Case 1: 2-geometry neighbours
            AT_new : = AT_i + 1 ;
           Case 2: diagonal neighbours
            AT_new : = AT_i + √2 ;
           if (AT_new < AT_j) then AT_j : = AT_new ;
          end
       end
    end
```

```
   CLEAR(L_bucket_index)
   if (temp-list ≠ Ø) then
    begin
     for each cell c_j in temp-list do
      INSERT(c_j , L_floor (Atj) mod 3)
     CLEAR (temp-list) ;
    end
   else bucket_index : = (bucket_index + 1) mod 3 ;
  end while ;
  if (path-exists = TRUE) then
   RETRACE(Cell-map(AT_D) , LL_path)
  else path does not exist ;
end .
```

3. PRESENTATION OF THE NEW SOLUTION

To simplify the presentation only the 4-geometry version is described here. However in general the algorithm is applicable to all geometries and it may also support varied terrain routing just the way the Chang, Jan and Parberry version does.

Data structure

The static data is stored in an array containing three fields for every cell :

SL – Integer number field whose value indicates whether a cell is assigned to sea or landmass (or any other static obstacle). Its value is 1 for sea, and infinity for a landmass.

GAT – A sub-array of the floating point numbers. The size of the array is equal to the maximum number of the neighbouring cells and is 8 for 4-geometry. Each field of this sub-array contains the gate arrival times for different incoming gates of the current cell. Gate arrival time is the time taken to travel from the source cell to the current cell via certain neighbour of the current cell.

VIS – Boolean field whose value indicates whether a cell has already been visited (inserted to *temp-list*) or not. *True* for a visited cell, *false* otherwise.

The dynamic data is stored in circularly used lists: $L_1 \ldots L_n$, and one extra temporary list *temp-list*. The number *n* of the used lists strictly depends on the specified maximum course alteration cost and thus it is indirectly configured by means of algorithm input parameters. *n = ceiling(maximum{single-step distances} + maximum{specified course alteration costs} + 1)*, where : *maximum(single-step distances)* is $\sqrt{2}$ for 4-geometry.

Algorithm overview

The key difference between the new solution and that proposed by Chang, Jan and Parberry is a replacement of each cell's *AT* field with *GAT* array. The idea of incoming gates arrival times makes it possible to take into account course alteration costs without sacrifying the linear complexities that characterized the previous versions. Every time cell data is updated, the arrival time of the appropriate gate is modified depending on the direction of the neighbouring cell that has initiated the update operation. The new candidate value of the gate arrival time field is determined according to the following formula :

$$GAT_{new,\,j,\,gate\_number} = \text{minimum}\ \{GAT_{i,1} + \\ + \text{distance}_{i\,,\,j} + \text{delay}_{gate\_number,1}, \ldots\ GAT_{i,8} + \\ + \text{distance}_{i\,,\,j} + \text{delay}_{gate\_number,8}\},$$

where :

i and *j* are indexes of the neighboring cells,
gate_number is the current gate of the c_j cell ,
numbers from 1 to 8 denote all gates of the c_i cell.

Delay values (penalties) are equal to zero for two gates of the same direction and have appropriate parameter values: d_1, d_2 or d_3 for two gates whose direction difference is 45°, 90° or 135°, respectively. The present *GAT* value is replaced with the candidate value if the new value is lesser than the current one.

The below presented diagrams illustrate the way the *GAT* array values of the wave front cells are updated.

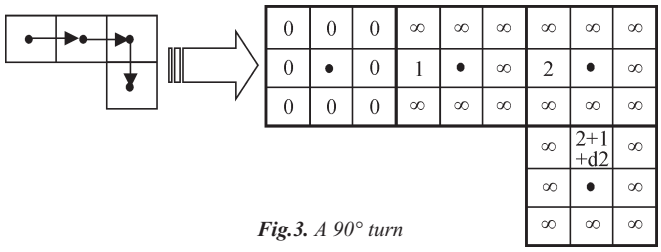

Fig.3. A 90° turn

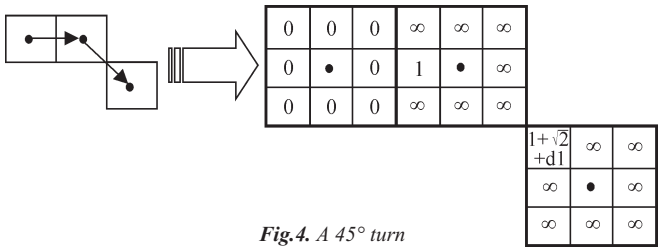

Fig.4. A 45° turn

Algorithm formal description

★ The procedures INSERT and CLEAR perform the same functions as those in the Chang, Jan and Parberry method.

★ The function GET_GATE_NUMBER (c_i,c_j) returns the number of the incoming gate of the cell c_j , through which the travel from the cell c_i is performed.

★ The procedure RETRACE retraces from the destination cell to the source cell and forms the output LL_{path} list. However, the rule of choosing the back way cells is different from that of the Chang, Jan and Parberry procedure. Here is chosen the cell whose sum of d_i ($i \in \{1,2,3\}$) modifier and the arrival time value of previous (closer to destination) cell's incoming gate is minimum.

Algorithm (in Pascal) of 4-GEOMETRY-ROUTER-WITH--TURN-PENALTIES (*Cell-map*, *S*, *D*, d_1, d_2, d_3, LL_{path})

```
Input: Cell-map, S, D, d_1 , d_2 , d_3
Output : LL_path
begin
 bucket_index : = 0 ;
 L_bucket_index : = S ;
 VIS_S : = TRUE ;
 temp-list : = Ø ;
 path-exists : = FALSE ;
 all-lists-empty : = FALSE ;
 number-of-lists : = ceiling(√2+ maximum{d_1 , d_2 , d_3}+1) ;
 while (all-lists-empty = FALSE) do
  if (D cell in L_bucket_index) then
   begin
    path-exists : = TRUE ;
    break while ;
   end
  for each cell c_i in L_bucket_index do
   begin
    for each cell c_j neighbouring c_i do
     begin
      if (SL_j = 1) then
       begin
        if (VIS_j = FALSE) then
         begin
```

```
            VIS_j : = TRUE ;
            INSERT(c_j , temp-list) ;
          end
        Case 1: 2-geometry neighbours
          distance : = 1 ;
        Case 2 : diagonal neighbours
          distance : = √2 ;
        gate_number : = GET_GATE_NUMBER(c_i,c_j) ;
        GAT_new : = GAT_i , gate_number + distance ;
        for each gate g_k of the cell c_i do
          begin
            Case 1: gate_number is
            the same direction as g_k
              delay_k : = 0 ;
            Case 2: gate_number and
            g_k difference is 45 degrees
              delay_k : = d_1 ;
            Case 3 : gate_number and
            g_k difference is 90 degrees
              delay_k : = d_2 ;
            Case 4 : gate_number and
            g_k difference is 135 degrees
              delay_k : = d_3 ;
            GAT_new , k : = GAT_i , k +
            + distance + delay_k ;
            if (GAT_new , k < GAT_new) then
              GAT_new : = GAT_new , k ;
          end
        if (GAT_new < GAT_j , gate_number ) then
          GAT_j , gate_number : = GAT_new ;
      end
    end
  end
  CLEAR(L_bucket_index)
  if (temp-list ≠ Ø) then
    begin
      for each cell c_j in temp-list do
        begin
          k : = floor (minimum{GAT_j , 1 ,... GAT_j , 8}) mod
          number-of-lists ;
              INSERT (c_j , L_k) ;
        end
      CLEAR (temp-list) ;
    end
  else bucket_index : = (bucket_index + 1) mod
  number-of-lists ;
  all-lists-empty : = TRUE ;
  for each list L_i do
    begin
      if (L_i ≠ Ø) then
        begin
          all-lists-empty : = FALSE ;
          break for
        end
    end
  end while ;
  if (path-exists : = TRUE) then
  RETRACE(Cell-map(GAT_D) , LL_path)
  else path does not exist ;
end .
```

Computational complexity

For each cell c_i whose distance from the source cell is equal to or lesser than that of the destination cell, the following actions are performed :

- each of its neighbours c_j is checked and possibly updated
- for each of its neighbours c_j, each of the gates of the cell c_i is checked.

This gives the total of $(2\cdot\lambda)\cdot(2\cdot\lambda)\cdot n$ steps for the worst case, where λ is a constant denoting geometry level (eight neighbours and eight gates for 4-geometry) and n is the number of cells whose distance from the source cell is equal to or lesser than that of the destination cell. Thus the computational complexity of the proposed solution is $O(n)$.

Example results of application of the proposed routing method and the Chang, Jan and Parberry algorithm

Below example results for the new routing method and the Chang, Jan and Parberry algorithm are presented. It has been assumed that costs of direction changes are: 1, 2 and 3 for 45°, 90° and 135° degree turns, respectively.

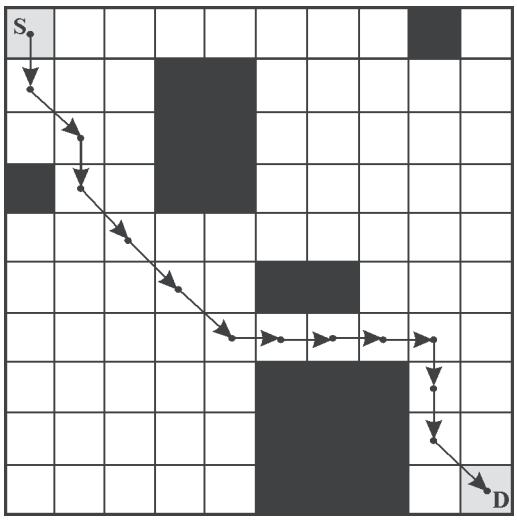

Fig.5. *Route determined by using the Chang, Jan and Parberry algorithm*

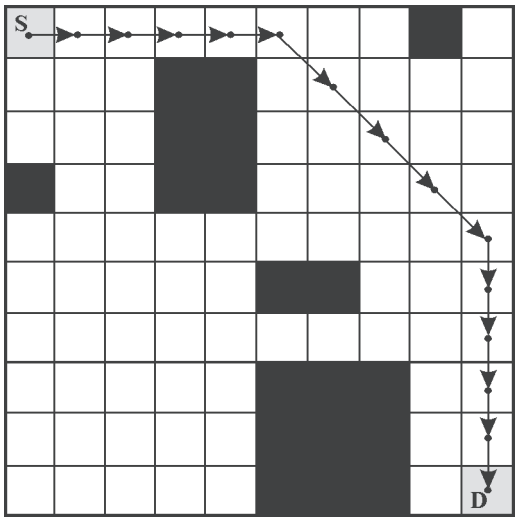

Fig.6. *Route determined by using the proposed algorithm with direction change costs (penalties) $d_1 = 1$, $d_2 = 2$, $d_3 = 3$*

In the above presented figures two different solutions of the same task are compared. The proposed algorithm (Fig.6), found a route with two turning points, different from that yielded by the Chang, Jan and Parberry method (Fig.5) which determined a route with six turning points. Thus the total path lengths and penalties for both methods are :

For the route by the Chang, Jan and Parberry method :

basic path length = $8 + 5\cdot\sqrt{2} \approx 15.07$

total penalties = $5\cdot d_1 + 1\cdot d_2 = 7$

total path cost = basic path length + total penalties ≈ 22.07

For the route by the proposed method :

$$\text{basic path length} = 10 + 4\cdot\sqrt{2} \approx 15.65$$
$$\text{total penalties} = 2\cdot d_1 = 2$$
$$\text{total path cost} = \text{basic path length} + \text{total penalties} \approx 17.65$$

From this comparison it can be stated that taking a seemingly longer (by 3.8%) route results in a much lower (by 22%) overall path cost.

4. EXAMPLES OF APPLICATION

The algorithm described in the paper is of general use. Therefore, whenever routing on the grid is involved and costs of direction changes matter, the benefits of using the proposed solution may be considerable.

In navigation, real-time routing and collision avoidance systems are the most obvious fields where the algorithm may be adapted. The method is applicable for ENC systems and ECDIS display on raster plane in particular. For a detailed description of a routing with collision avoidance system that may take advantage of the presented algorithm, the reader is referred to the paper [1].

Other potential maritime applications include pipeline and airduct routing, where minimizing the number of bends is crucial for cutting down the costs. However the algorithm would have to be first upgraded to a 3-D version. Fields of use outside maritime industry may include VLSI design.

5. SUMMARY

In the paper a general searching method on the raster plane was presented. Owing to its new data structure the algorithm is capable of including costs of direction changes in the total cost of an optimal path, while keeping the linear computational time and space complexities. Therefore when costs of direction changes are significant the solution is superior to those previously known. Possible applications include finding optimal ship routes since their course alterations might be expensive or even risky in presence of obstacles.

NOMENCLATURE

List of variables used in the algorithms :

all-lists-empty	– indicates whether all buckets (lists) are empty or not
AT_{index}	– the arrival time (a time when the cell of the given index can be reached)
bucket_index	– index of the bucket currently in use
Cell-map	– a map of cells, containing all relevant information
d_1 , d_2 , d_3	– the turn penalties (delay times for 45, 90 and 135-degree turns, respectively)
delay	– a delay resulting from the direction change
distance	– the distance between two neighbouring cells
D	– a destination cell
gate_number	– the number of the incoming gate, through which the cell is reached
$GAT_{cell\_index,\ gate\_number}$	– the gate arrival time (a time when the incoming gate of the given cell can be reached)
LL_{path}	– the returned path
number-of-lists	– the number of the buckets (implemented as lists)
path-exists	– indicates whether the searched path has been found or not
S	– a start cell
SL_{index}	– indicates whether the cell of the given index is sea (passable) or land (impassable)
temp-list	– a temporary list
VIS	– indicates whether a cell of the given index has already been visited
∅	– empty list

Abbreviations

DEM	– Digital Electronic Maps
ECDIS	– Electronic Chart Display and Information System
ENC	– Electronic Navigational Charts
GIS	– Geographical Information System
GPS	– Global Positioning System
VLSI	– Very Large Scale Integration
2- and 4-geometry	– two, four trajectories, respectively
3-D	– three dimensions

BIBLIOGRAPHY

1. Chang K. Y., Jan G.E, Parberry I. : *A Method for Searching Optimal Routes with Collision Avoidance on Raster Charts*. Royal Institute of Navigation. The Journal of Navigation No.56/2003
2. Kubale M. : *Introduction to Computational Complexity*. Wydawnictwo Politechniki Gdańskiej (Publishing House of Gdańsk University of Technology). 1994
3. Lee C.Y. : *An Algorithm for Path Connection and Its Applications*. IEEE (Institute of Electrical and Electronics Engineers) Transactions on Electronic Computers, EC-10, 1961
4. Weintrit A. : *The Electronic Chart Systems and Their Classification*. Annual of Navigation, No 3/2001. Polish Academy of Sciences. Polish Navigation Forum, Gdynia 2001

CONTACT WITH THE AUTHOR

Rafał Szłapczyński, M.Sc.,Eng.
Faculty of Ocean Engineering
and Ship Technology,
Gdańsk University of Technology
Narutowicza 11/12
80-952 Gdańsk, POLAND
e-mail : rafal@pg.gda.pl

Conference

SCIENTIFIC SEMINAR OF REGIONAL GROUP *of the Section on Exploitation Foundations*

On 25 November 2004 the Institute of Fluid-Flow Machinery, Polish Academy of Sciences (PAS), Gdańsk, hosted the last-in-the-year scientific seminar of the Regional Group of the Section on Exploitation Foundations, Machine Building Committee, PAS.

During the main part of the seminar 4 papers were presented whose authors came from scientific staff of the Institute, namely :

- *On energy and entropy* – by J. Mikielewicz
- *Plasma techniques applied to modern cars* by J. Mizeraczyk
- *Experimental modal analysis* – by M. Łuczak
- *Structural funds – research projects and their financial assembling* – by J. Kiciński

After interesting discussion on the papers Prof. J.Girtler, the Group's Chairman, presented a proposed plan of seminars for the year 2005.

The seminar was ended by presentation of the up-to-date laboratory facilities of the Institute.

FOREIGN *conference*

PSAM 7 – ESREL'04

On 14-18 June 2004 in Berlin had place the International Conference on :

Probabilistic Safety Assessment and Management

It was organized by two bodies : International Association for Probabilistic Safety Assessment and Management, and European Safety and Reliability Association.

Due to such arrangement its participants took part in a very broad meeeting of scientists, researchers and experts of many branches, which was demonstrated in 594 papers prepared by authors from 40 countries, from all the world around.

Among the gremium were also representatives of Polish scientific research centres who presented the following topics :

- ⇨ *Environmental safety of sea-going ship power plant* – by Brandowski A., and Liberacki R. (Gdańsk University of Technology)
- ⇨ *Risk in the aspect of safety and reliability of autonomous system* – by Drobiszewski J. and Smalko Z. (Warsaw University of Technology)
- ⇨ *New methods of solving multidimensional non-linear optimization problems – application thereof to construct air-line flight schedules* – by Jaźwiński J., Klimaszewski S., and Żurek Z. (Air Force Institute of Technology, Warsaw)
- ⇨ *Availability improvement of port transportation structures and their operation processes* – by Kołowrocki K. (Gdynia Maritime University)
- ⇨ *Incorporation of human and organizational factors into qualitative and quantitative risk analyses*- by Kosmowski K.T. (Gdańsk University of Technology)
- ⇨ *Modelling and uncertainty in system analysis for safety assessment* – by Kosmowski K.T. (Gdańsk University of Technology)
- ⇨ *On reliability improvement of the port grain transport systems* – by Kwiatuszewska – Sarnecka B. (Gdynia Maritime University)
- ⇨ *Reliability model of combined transportation system* by Nowakowski T. (Wrocław University of Technology).

Moreover, Mr J. Górski of Gdańsk University of Technology took part in the activity of *Application Area Co-ordinators*, a 16-person international organizational team, within the area : *Information Technology and Telecommunication*.

Miscellanea

World qualification level for welding engineers

Without doubt welding is one of the most important fields of engineering.

Welding, soldering and bonding – these are engineering processes which play an important role in production, repair and operation of objects. Therefore welding is teached in many engineering schools and universities worldwide.

However it has turned in practice that their graduates – – welding engineers with their qualification do not quarantee to cope with required responsible supervision over realized welding processes, which detrimentally influence quality and durability of produced engineering objects. In order to improve the situation European Federation for Welding (**EFW**) and International Institute of Welding (**IIW**) came to the conclusion that welding processes – from the point of view of quality assurance – require a.o. special surveillance by specially trained engineering personnel.

Education of such personnel is carried out during a special training supplementary to academic studies. It is realized in accordance with the requirements defined in guidelines for unified educational system (Minimum Requirements for Education, Examination and Qualification) issued by the above mentioned organizations. The requirements contain overall description of aims and scope of the education, as well as conditions and criteria for carrying out final examination to obtain the diploma of European/International Welding Engineer (EWE/IWE).

In Poland it was Gdańsk University of Technology which first undertook such a task, where at its Mechanical Faculty the IIW Approved Training Body was organized.

Its activity was initiated during spring semester of 2001/2002 academic year under the authority of Welding Institute, Gliwice, being the IIW Authorized National Body. The first 300-hour course of the education on the highest international level was completed in July 2004. All 24 participants of the course passed the very difficult final exam with the mean grade of "almost very good". This way they entered the group of welding engineers of the highest international level which makes it possible to take up the highest posts in this profession.