computing@tanet.edu.te.ua
www.tanet.edu.te.ua/computing

ISSN 1727-6209
International Scientific Journal of Computing

VEHICLE ROUTING PROBLEM WITH TIME WINDOWS

Vladimir Vacic and Tarek M. Sobh

Department of Computer Science and Engineering
University of Bridgeport, Bridgeport, CT 06604, USA
vladimir@vacic.org, sobh@bridgeport.edu

Abstract: *The topic of this paper is a Genetic Algorithm solution to the Vehicle Routing Problem with Time Windows, a variant of one of the most common problems in contemporary operations research. The paper will introduce the problem starting with more general Traveling Salesman and Vehicle Routing problems and present some of the prevailing strategies for solving them, focusing on Genetic Algorithms. At the end, it will summarize the Genetic Algorithm solution proposed by K.Q. Zhu which was used in the programming part of the project.*

Keywords: *Vehicle Routing Problem with Time Windows, Genetic Algorithms*

1. INTRODUCTION

Vehicle Routing Problem with Time Windows is a variant of one of the most well known problems in contemporary operations research. Informally, the goal of the problem is to determine an optimal route for delivery of packages to customers who have specified when they will be available to receive their packages, taking into consideration vehicle and package size, and possibly some additional constraints. A more formal definition of the problem will be given in Section 5.

Interest in the Vehicle Routing Problem (VRP) grew rapidly after World War II, following the increase in postal traffic and catalog ordering of goods from a remote retailer. VRP falls under a broader category of transportation problems, which also include fleet management, facility location, traffic assignment, air traffic control etc. [5]

Vehicle Routing Problem has a historical and theoretical background in the Traveling Salesman Problem; both of these address the problem of finding a minimal cost route within a predefined set of points, given a set of constraints. Minimal cost route is usually described as the shortest route in terms of Euclidean distance.

A number of solving methodologies will be explored in this paper, with Genetic Algorithms (GA) being examined in greater detail. Finally, a GA solution to the Vehicle Routing Problem with Time Windows proposed by K.Q. Zhu [13] will be presented.

2. TRAVELING SALESMAN PROBLEM

The Traveling Salesman Problem (TSP) can be stated as: given a finite number of nodes (cities) and the cost of travel between them (typically a function of their geographical distance), find the least expensive way to visit all nodes and return to the starting node.

TSP traces its origin to the so-called Icosian Game, invented in the 1880's by the Irish mathematician Sir William R. Hamilton. The goal is to find a way to visit all 20 points of a two-dimensional representation of an icosahedron, without visiting any point more than once.

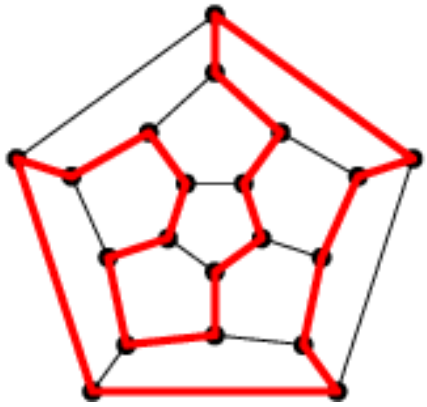

Fig. 1 - A solution to Hamilton's Icosian Game

According to Applegate et al. [1] the largest solved instance of the Traveling Salesman Problem is a tour of 15,112 cities in Germany. The

computation was carried out on a network of 110 processors located at Rice and Princeton Universities. The total computer time used in the computation was 22.6 years, scaled to a Compaq EV6 Alpha processor running at 500 MHz. The optimal tour has a length of approximately 66,000 kilometers.

The Traveling Salesman Problem can be applied to many industrial applications, such as microprocessor manufacturing, transportation and logistics problems, etc.

3 VEHICLE ROUTING PROBLEM

Vehicle Routing Problem (VRP) is one of the most important topics in operations research. It deals with determining least cost routes from a depot to a set of scattered customers. The routes have to satisfy the following set of constraints:

- Each customer is visited exactly once
- All routes start and end at the depot
- Sum of all demands on a route must not exceed the capacity of a vehicle

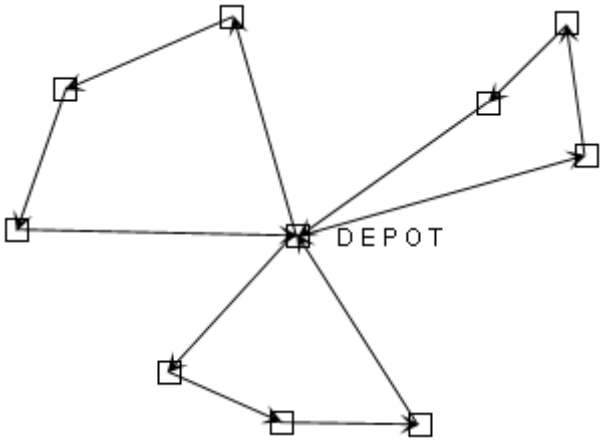

Fig. 2 - An example solution to a Vehicle Routing Problem

VRP is closely related to TSP, and according to Bullheimer et al. [2], as soon as the customers of the VRP are assigned to vehicles, the problem is reduced to several TSPs.

Based on the following two criteria:

- Whether the vehicle is capacitated or uncapacitated (is there a limit on how many passengers or objects a vehicle can take at any given time)
- Whether it has one or more starting points (does the vehicle pick passengers or objects from a central location (i.e. a bus terminal, or a warehouse), or a pick-up can occur on a number of places)

Gendreau and Potvin [5] devised a schematic to classify VRP variants:

	many-to-many	one-to-many
Capacitated	dial-a-ride	feeder system
Uncapacitated	express mail delivery	courier or repair services

Fig. 3 - Vehicle Routing Problem classification

Many-to-many problems with capacity constraints are typically the most difficult ones because pick-up and delivery locations must be located on the same line, and the pick-up point must always precede the delivery location.

Dial-a-ride Problem (DARP), also known as the Stacker Crane Problem, comes in two flavors depending on whether the vehicle is allowed to leave objects on intermediate locations and later pick them up and deliver them. DARP arises in several practical applications [3] such as transportation of elderly or disabled persons, tele-buses and shared taxi services.

Courier and Repair Services [5] should be contrasted to the Traveling Salesman Problem with Time Windows with respect to that the time windows are specified by a central planner in order to minimize the route cost. Another specific of the Repair Services is that the service time is a significant portion of the total schedule time.

4 VEHICLE ROUTING ALGORITHMS

Vehicle routing and dispatching problems are topics of a great deal of ongoing research in the operations research community since the late fifties, which reflects VRP's central role in distribution management.

According to one classification, proposed by Fisher [4], vehicle routing algorithms fall into three categories:

- Simple heuristics based on local search and sweep, developed mostly in the 60's and 70's.
- Mathematical programming based heuristics, which approximate VRP with generalized assignment and set partitioning problems.
- Exact optimization (K-tree, Lagrangean Relaxation) and artificial intelligence methods (Simulated Annealing, Tabu Search, Ant System and Genetic Algorithms).

The algorithm used in this project falls under the Genetic Algorithms category. The following

sections will briefly describe each of the AI methods mentioned.

4.1 Simulated Annealing

Simulated Annealing is a generalization of the Monte Carlo method modeled after the way liquids freeze in the process of annealing. A thermodynamic system goes from a high-energy, disordered state into a "frozen", more ordered one. Using this analogy a thermodynamic system corresponds to the current solution of the combinatorial problem, the energy equation for the thermodynamic system corresponds to the objective function, and the ground state is analogous to the global minimum.

Initially, a thermodynamic system starts at an energy level E and temperature T. While T is being kept constant, the initial configuration is perturbed and changes in energy dE are observed. If the change in energy is negative, new configuration is accepted. If the change in energy is positive, new configuration is accepted with probability:

$$e^{-\frac{dE}{T}}$$ (Boltzmann factor)

The process is then repeated until good sampling statistics are gathered for the current temperature, then the temperature is decreased and the whole process is repeated until the frozen state is achieved.

Simulated annealing is used in various combinatorial optimization problems, in particular in circuit design problems.

4.2 Tabu Search

Glover [6] describes Tabu Search (TS) as a meta-heuristic superimposed on another heuristic. The high level approach is to prevent the algorithm from going in cycles by forbidding or penalizing moves which take the next iteration of the solution to points in the solution space previously visited (those points are declared "tabu").

Tabu method was modeled after observed human behavior – given a similar set of circumstances, humans will act slightly differently on different occasions. This randomness might cause an error, but may also be beneficial and cause an improvement. TS operates in this fashion, except that it does not make random choices, but operates on the premises that there is no point in accepting a poor solution unless it is to avoid a path already investigated. This provides for investigation of new regions of the problems solution space, avoiding local minima and ultimately finding the desired solution.

TS first looks for a local minimum, and to avoid repeating the solutions it already examined, it stores them in one or more Tabu lists. The original intent of the list was not to prevent a previous move from being repeated, but rather to insure it was not reversed. The role of the memory can change during the course of the algorithm execution – at initialization the goal is to make a coarse examination of the solution space (referred to as diversification), but as candidate locations are identified the search is more focused to produce local optimal solutions (referred to as intensification). Different TS methods differ primarily in the size and adaptability of the Tabu memory, having them customized to a particular problem.

The method is still actively researched, and is constantly being improved. Tabu Search in being used in integer programming problems, scheduling, routing, traveling salesman and related problems.

4.3 Ant Systems

Bullheimer et al. [2] describe Ant Systems as a distributed meta-heuristics for solving hard combinatorial optimization problems, first used to solve the TSP. They were introduced by Colorni, Dorigo and Maniezzo, and are based on observed behavior of real ant colonies in search of food. Namely, ants communicate the information about food sources by using pheromones to mark the paths which lead to food. Fellow ants can follow the pheromone trail and while following it they will additionally mark it with new pheromones, thus attracting more ants. In result, paths which quickly lead to rich food sources will be reinforced.

In solving a problem, simulated ants are searching the solution space, the quality and size of the food source correspond to the objective values that are being optimized, and adaptive memory plays a role of the pheromone trail. Artificial ants are heuristically searching through the solution space.

Ant Systems are used in timetabling, production and route scheduling.

4.4 Genetic Algorithms

Genetic Algorithms (GA for short) are a class of adaptive heuristics based on the Darwinian concept of evolution – "survival of the fittest." They have been developed by J. Holland at the University of Michigan in 1975. Solutions to a problem are encoded as chromosomes, and based on their fitness as evaluated by an evaluation function, good properties of a generation of solutions are propagated to a next generation.

Two main aspects of a GA, solution encoding and evaluation function, are problem specific [12]. The most common way to encode a solution into a chromosome is to discretize the variables which we

are trying to optimize on range of a power of 2 and then represent them as bit strings. For example, if the range of values a variable can take is from 0 to 32, we will then have to uses 5 bits to encode this variable. Another popular encoding method is permutation encoding, which is more suitable for ordering problems. An evaluation function is used to estimate how optimal a particular solution is.

Genetic algorithms typically have the following structure:

```
initialize timer
generate a random population
evaluate fitness

while not terminated do
{
        increment timer
    select the fittest parents
        recombine genes of
selected parents
    introduce mutations
        evaluate fitness
    select survivors that will
become the next generation
}
```

A GA will start with a set of chromosomes called the initial population. Each chromosome represents a solution to the problem, and the initial population is either randomly generated (in which case it would take longer time for the algorithm to converge to the solution) or generated using some form of heuristics (in which case the population is already closer to the solution, and would hence take less time to converge).

A selection mechanism will then be used to select the prospective parents based on their fitness, which is computed by the evaluation function. Their offspring will constitute the next generation. Selection mechanisms will be explained better in section 4.4.1. The selected parent chromosomes will then be recombined via the crossover operator to create a potential new population. Some crossover operators will be examined in greater detail in section 4.4.2.

The next step will be to mutate a small number of the newly obtained chromosomes, in order to introduce a level of randomness that will preserve the GA from converging to a local optimum. A mutation is typically a random swap in the gene sequence, or a random negation of a bit if the chromosome was bit-encoded.

Finally, new population will then be selected based on the fitness of the candidate chromosomes.

The genetic algorithm will reiterate through this process until a stoping criterion was met, which can be one of the following:

- predefined number of generations has been produced
- there was no improvement in the population, which would mean that the GA has found an optimal solution
- a predefined level of fitness has been reached.

4.4.1. Selection Mechanisms

Roulette Wheel Selection

Roulette Wheel Selection (RWS) is one of the most common proportionate selection schemes. Man et al. [8] describe it in algorithmic fashion:

1. Sum the fitness of all population members, let's call it F_{sum}.
2. Generate a random number between 0 and F_{sum}
3. Return the first population member whose fitness, added to the fitness of the preceding population members, is greater or equal to the randomly generated number.

To illustrate RWS graphically, in the following Fig. F_{sum} corresponds to the circumference of the Roulette wheel. Chromosome 3 being the fittest occupies the largest interval on the

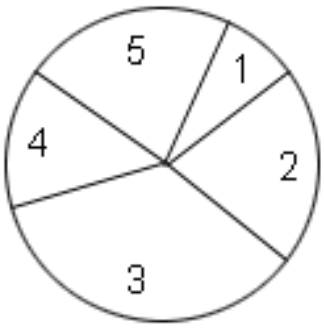

Fig. 4 - Roulette Wheel

circumference. A random number is generated and the chromosome whose segment spans the random number will be selected. This procedure will be repeated until a sufficient number of chromosomes is selected.

Tournament Selection

In Tournament Selection we keep two identical (though differently ordered) copies of the population. In every generation, we compare adjacent chromosomes in one copy of the population pair by pair, and select the chromosome with greater fitness value, and then we proceed with the second copy of the population to select the other half of the selected population. The advantage of this mechanism is that genetically fitter chromosomes are given priority, but the average chromosomes have some chances of being selected if they happen to be compared with a less fit chromosome.

4.4.2. Crossover Operators

Crossover refers to the recombination of genetic information between two chromosomes. Crossover implementation heavily depends on the way the chromosomes were encoded. This section will present some of the most common crossover operators. [8]

One-Point Crossover

This is the simplest crossover operator – a cut-off point is randomly selected, and then the genes after the cut-off point are swapped between the chromosomes.

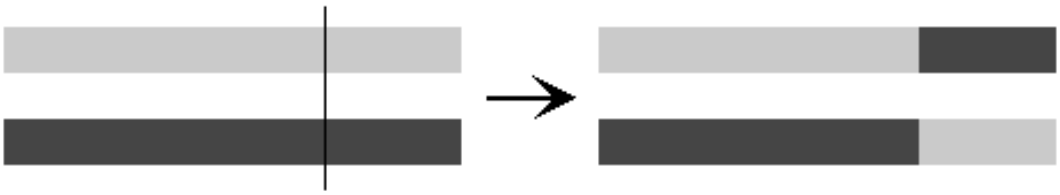

Fig. 5 - One-Point Crossover

Multi-Point Crossover

This is a slight modification of the previous operator, the only difference being that now we have more than one crossover point.

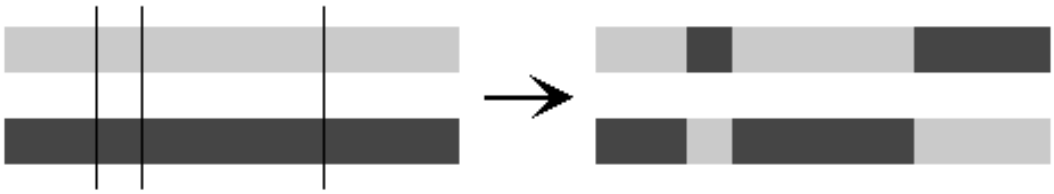

Fig. 6 - Multi-Point Crossover

One and Multi-Point Crossover operators are useful in cases when chromosomes were bit encoded.

Partially Mapped Crossover (PMX)

This crossover method is used when chromosomes are permutation encoded. It preserves the locations of as many genes in both chromosomes as possible. For example:

Parent 1: H K C E | F D B | L A I G J

Parent 2: A B C D | E F G | H I J K L

Two random cut-off points and the segment between them define a mapping (in this case,

$E \rightarrow F$, $F \rightarrow D$, $G \rightarrow B$) which will be applied to the genes of Parent 1 that have not been swapped. This way the duplicate genes will be modified (so the resulting chromosome would still be a permutation), while the remaining ones will stay in place.

Parent 1: H K C D | E F G | L A I B J

Parent 2: A G C E | F D B | H I J K L

Ordered Crossover

This crossover method preserves the relative order of as many genes in both chromosomes as possible. For example:

Parent 1: H K C E | F D B | L A I G J

Parent 2: A B C D | E F G | H I J K L

We select two random cut-off points, determine the order of genes in a chromosome starting with the first gene after the segment which is to be swapped (for example, for Parent 1, that would be LAIGJHKCEFDB), and then from the sequence we will remove the genes which already are in the swapped segment (in this example, those would be EFG, so the remaining sequence would be LAJHKCDB). The resulting offspring would be:

Parent 1: K C D B | E F G | L A I J H

Parent 2: A C E G | F D B | H I J K L

5 VEHICLE ROUTING PROBLEM WITH TIME WINDOWS (VRPTW)

A formal definition of VRPTW can be stated as: let $G=(V, A)$ be a graph with node set $V = V_N \cup \{v_0\}$ and arc set A, where $V_N = \{v_i \in V \mid i = 1, 2, ..., n\}$ stand for customer nodes and v_0 stands for the central depot, where all routes start and end. Each node $v_i \in V$ has an associated demand q_i, service time s_i, service time window $[e_i, l_i]$ and an ordered pair of coordinates (x_i, y_i). Based on the geographical coordinates, it is possible to calculate the distance d_{ij} between every two distinct nodes v_i and v_j, and the corresponding travel time t_{ij}.

Under the time window constraints there may or may not be a transition (an arc) between certain node pairs. The set of arcs can be defined as $A = \{(v_i, v_j) \mid v_i, v_j \in V \wedge t_{0i} + s_i + t_{ij} \le l_j\}$. If the vehicle reaches the customer v_i before the e_i, a waiting time occurs. The route's schedule time is the sum of the travel time, waiting time and the service time.

The objective of VRPTW is to service all customers while minimizing the number of vehicles, travel distance, schedule time and waiting time without violating vehicles' capacity constraints and the customers' time windows.

Some of the most useful applications of the VRPTW include postal deliveries, industrial refuse collection, national franchise restaurant services, school bus routing, and security patrol services.

6 ALGORITHM DESCRIPTION

The algorithm used in the programming part of the project was developed by K.Q. Zhu [13].

A solution to the problem is represented by an integer string of length N, where N is the number of customers which need to be served. All routes are encoded together, with no special route termination characters in between; chromosomes are decoded back into routes based on the customers' demand and the vehicle capacity.

Part of the population is initialized using a heuristic method, and part is initialized randomly. Having a non-random part of the population greatly reduces the time for the solution to converge, and having a random part prevents converging to a local optimum and not exploring the complete solution space.

Candidates for mating are selected using the tournament selection, and two kinds of crossover operators are being used: heuristic and merge.

Heuristic crossover deals with distances between nodes: for example, a random cut was made on two chromosomes, and we will compare the genes immediately after the cuts.

Parent 1: H K C E F D B L A I G J

Parent 2: A B C D E F G H I J K L

Let us say that B is to be the first gene in the child. Gene G has to be swapped with B in Parent 2 to avoid subsequent repetition. After swapping, we have:

Parent 1: H K C E F D B L A I G J

Parent 2: A G C D E F B H I J K L

Now we compare the distance between B and the first two subsequent genes, L and H, and choose the

one which is geographically closer to B. Once again, in Parent 2 we swap the gene which was chosen with the remaining one, to avoid duplication. This process is continued until a new chromosome of length N is formed. A similar heuristic can be created by deleting a gene from a parent, instead of swapping it. We name those crossover operators HeuristicCrossover 1 (H1) and HeuristicCrossover 2 (H2) respectively.

Merge crossover operates on the basis of time precedence, defined by the time windows corresponding to each node. Similarly, the first gene is chosen randomly, and the following gene will be the one whose time window comes earlier. Similar to the heuristic case, once again we can either swap or delete the gene in the other parent, and thus we obtain MergeCrossover 1 (M1) and MergeCrossover 2 (M2).

Heuristic and Merge Crossover operators will produce only one child each. Since geographic distribution is equally important as time sequence, from the same pair of parent chromosomes we will create one child by using the Heuristic Crossover and one child using the Merge Crossover. We now have 4 possible combinations which would produce the offspring: M1H1, M1H2, M2H1, M2H2. Experiments have shown that the H1M2 performs the best out of the four, as was reported in [13].

A probability that a pair of selected parents will mate is called the probability of crossover. When a couple of parent chromosomes is determined not to mate, it will be copied verbatim into the next generation.

Mutation schemes used are swap node and swap sequence.

Zhu uses an adaptive mutation probability scheme, which changes the mutation probability as the standard deviation of the population fitness changes. It is described as

$$P_{mutation} = 0.06 + 0.1 \cdot (5 - \sqrt{\frac{\sum_{j=0}^{N-1}(x_i - \bar{x})^2}{N-1}})$$

where x_i is the fitness of a chromosome, $\bar{x}$ is the mean fitness and N is the population size. We can see that a mutation probability of 0.06 is always guarantied.

Zhu also adds hill-climbing in order to improve the chromosomes obtained through crossover and mutation. Hill-climbing is a scheme for randomly selecting a portion of the population, decoding the chromosomes into solutions and then improving those solutions by a few iterations of removal and reinsertion. To prevent the algorithm from converging to a local optimum, chromosomes are selected for hill-climbing with a probability 0.5.

And, at the end, to additionally improve the quality of the population, the worst 4% of the population will be replaced with the best 4% of the parent population. Note that the percentage of chromosomes to be improved needs to be less than the mutation rate, otherwise no mutated chromosomes will be able to escape the improvement process.

The algorithm was tested on six Solomon's [10] VRPTW Benchmark Problem data sets (R1, C1, RC1, R2, C2, RC2) with 25 customers, as reported in [13].

7 CONCLUSION

Genetic Algorithms fall in the category of methods known as "weak methods," which are robust, but take time to produce a solution. They perform best in situations when it is either not clear how to find a solution, or when there is little information available about which parameters would be good. Blind search which pure GAs offer does not yield best solutions in situations in which we know enough about the problem. Specifically, Tabu Search shows the best performance for VRP, and according to Bullnheimer et al. [2] the superiority of this approach is due to the fact that there was more VRP research involved with Tabu Search than with any other method, and consequently TS solutions have been constantly improved.

Almost by a rule, hybrid genetic algorithms, in which pure GAs are combined with other methods to produce a new approach, give better solutions that strict GA.

However, the strictly GA algorithm devised by Zhu [13] described in this paper offers solutions competitive to the ones obtained by Tabu Search and Simulated Annealing, and in 4 out of 56 cases of Solomon's [10] 100 VRPTW problems, it outperformed or gave results equal to the best results presented by Solomon.

REFERENCES

[1] Applegate D., R. Bixby, V. Chvátal, W. Cook. (2002): *Solving Traveling Salesman Problems.* http://www.math.princeton.edu/tsp/

[2] Bullnheimer B., R.F.Hartl, C. Strauss. (1997): *Applying the Ant System to the Vehicle*

Routing Problem. Department of Management Science, University of Vienna.

[3] Charikar M., Raghavachari B. *The Finite Capacity Dial-A-Ride Problem.* Stanford University and The University of Texas, Dallas.

[4] Fisher, M.L., K.O. Jornsten and O.B.G. Mansen. (1992): *Vehicle Routing with Time Windows*. Working paper.

[5] Gendreau M., J. Potvin. (1998): *Dynamic Vehicle Routing and Dispatching*. NEC Research Institute.

[6] Glover F. (1986): Future paths for Integer Programming and Links to Artificial

Intelligence. *Computers and Operations Research*.

[7] Gray P., W. Hart, L. Painton, C. Phillips, M. Trahan, J. Wagner. (1997): *A Survey of Global Optimization Methods*. Sandia National Laboratories.

[8] Man, K.F., K.S. Tang and S. Kwong. (1999): *Genetic Algorithms: Concepts and Designs*. Springer – Verlag London Limited.

[9] Louis, Sushil J., Xiangying Yin, Zhen Ya Yuan. (1999): *Multiple Vehicle Routing With Time Windows Using Genetic Algorithms*. Genetic Adaptive Systems LAB. University of Nevada, Reno.

[10] Solomon, Marius M. (2000): *VRPTW Benchmark Problems*. http://web.cba.neu.edu/ ~msolomon/problems.htm

[11] Tan, K.C., L.H. Lee and K.Q. Zhu. (1999): *Heuristic Methods for Vehicle Routing Problem with Time Windows*.

[12] Whitley, D. *A Genetic Algorithm Tutorial*. University of Colorado.

[13] Zhu, Kenny Qili. (2000): *A New Genetic Algorithm for VRPTW*. National University of Singapore.

Vladimir Vacic *was born on August 12, 1978 in Belgrade, Serbia and Montenegro. In 1997, he graduated from the School of Mathematics, Belgrade, Serbia, a specialized school whose curriculum emphasizes the natural sciences, Mathematics and Computer Science. The topic of his final thesis was in the area of Projective Geometry.*

In 1998, He enrolled in the University of Bridgeport (Connecticut), and moved to the United States. For a year and a half, he was the webmaster of the University of Bridgeport. After that, he was the Director of Technology for Artxone, Inc., a small B2C and B2B e-commerce company in Westport, Connecticut, specializing in limited edition art gifts and collectibles. He also worked as a Java Developer at Divine Enterprise Portals (formerly Sagemaker) in Fairfield, Connecticut, focusing on J2EE development. He was a President of the Upsilon Pi Epsilon (The Honor Society of the Computing Sciences) Connecticut Delta Chapter at the University of Bridgeport, and a member of the Phi Kappa Phi Honor Society. In 2002, He obtained his Bachelor of Science Degree from the University of Bridgeport, double majoring in Computer Science and Mathematics. In 2004,he obtained his Master of Science Degree from Temple University, Philadelphia. Mr. Vacic is interested in electronic music, reading, traveling, Aikido, 20th century architecture, contemporary visual and conceptual arts, urbanism, cycling, philosophy, Slavic and Norse mythology, dancing, hiking, theatre, magick and cooking.

He maintains www.yutechno.org, a web site dedicated to the Serbian electronic music scene.

Professor ***Tarek M. Sobh*** *received the B.Sc. in Engineering degree with honors in Computer Science and Automatic Control from the Faculty of Engineering, Alexandria University, Egypt in 1988, and M.S. and Ph.D. degrees in Computer and Information Science from the School of Engineering, University of Pennsylvania in 1989 and 1991, respectively. He is currently the Dean of the School of Engineering at the University of Bridgeport, Connecticut; the Founding Director of the Interdisciplinary Robotics, Intelligent Sensing, and Control (RISC) laboratory; a Professor of Computer Science, Computer Engineering, Electrical and Mechanical Engineering; and the Chairman of the Prototyping Technical Committee of the IEEE Robotics and Automation Society. He was the Interim Chairman of Computer Science and Computer Engineering and the Director of External Engineering*

Programs at the University of Bridgeport. He was an Associate Professor of Computer Science and Computer Engineering at the University of Bridgeport from 1995 -- 1999, a Research Assistant Professor of Computer Science at the Department of Computer Science, University of Utah from 1992 -- 1995, and a Research Fellow at the General Robotics and Active Sensory Perception (GRASP) Laboratory of the University of Pennsylvania from 1989 -- 1991.

He was the Chairman of the Discrete Event and Hybrid Systems Technical Committee of the IEEE Robotics and Automation Society from 1992- 1999. His background is in the fields of computer science and engineering, control theory, robotics, automation, manufacturing, AI, computer vision and signal processing. He has published over 100 journal and conference papers, and book chapters in these and other areas.

Dr. Sobh has been awarded many grants to pursue his research.

Dr. Sobh is a Licensed Professional Electrical Engineer (P.E.) and a Certified Manufacturing Engineer (CMfgE) by the Society of Manufacturing Engineers, a member of Tau Beta Pi (The Engineering Honor Society),Sigma Xi (The Scientific Research Society), Phi Beta Delta (The International Honor Society), Upsilon Pi Epsilon (The Computing HonorSociety) and Delta Mu Delta (The Business Adminstration Honor Society). Dr. Sobh was the recipient of the Best Paper Award at the World Automation Congress Conference (WAC 98) in Anchorage, Alaska.