Artificial intelligence applied to rescheduling and optimisation.

James Cunha Werner (wernerjc@sbu.ac.uk)
Terence C. Fogarty (fogarttc@sbu.ac.uk)
SCISM
South Bank University
103 Borough Road
London SE1 0AA

The purpose of this paper is propose a data modelling structure with permits an easy simulation of distributed systems (like data links net and transport systems) associate with artificial intelligence algorithms (Genetic Algorithms) to solve the problem of scheduling and rescheduling applied to settled routes (train/tube/bus transport system).

Introduction and definitions.

Any system under study would be classified in two different classes: continuous and distributed.
Concentrated or continuous parameters are modelled as set of coupled ordinary differential equations, with means transport system quite null and the spatial characteristics would be despised.
Distributed parameters are systems where transport time will not be despised, and the system model is realized as coupled partial differential equations. In this class there are all kind nets and queues problems (see SwedeTrack), usually called logistic. Its main problem is that any action the control system applies will take some time to be realized over all system, with requires the simulation system to be fast, compact and reliable, to be appropriate to apply in real time optimisation with a safe time interval prediction.
This paper addresses a proposal of a pointer structure to retrieve the information necessary to simulate a distributed system, and use it to optimise the scheduling and the rescheduling with genetic algorithms. The main advantage of this optimisation approach is the spread population around several maximum points, which enable a fast convergence when exogenous events occur.

Modelling distributed systems with pointer structure.

To code an efficient structure for information retrieval, we define a structure with indirect pointer with explores the compiler access mechanism. The structure uses the following entities (Figure 1):

- Node: is a connection point between two sectors of the net. It has his owns attributes, like wait time and load (of passengers or commodities).
- Segment: is a trajectory between two nodes, with is divided in slices, called sectors with same constraint and physical characteristics, occupied by just one transport element (link with transport element entity). These constraint deals with system fail safe conditions. Between two sectors, there is a logical switch to implement different strategies of real time events.

- Route: is an origin/destiny trace that the transport element would follow. This entity enlaces data modelling entities.
- Transport element: is the dynamic part of the system, with transfer the load between nodes following the route.

Each entity has its own access methods to the others, resulting in a compact and flexible source code with long pointers references bottleneck. To obtain the name of actual transport element node, we need to point:

```
node[segment[route[transp_elem[i].route].sect[transp_elem[i].trecho_atu]].fim].name
```

The dynamic structure and update of a simulation system with these pointers structure reduce to:

1. Transport element position update by the change of vehicle pointer in sector entity, with means that all net dynamic update is doing by physical properties of element and sector availability.
2. Node load (with passengers or commodities) depends only of feed rate, origin/destine matrix and transfer percentage between connections.
3. Transport flux depends of transport element available capacity and route.

Each structure element contains specific data (for example the characteristics of a segment, like size, maximum velocity, etc) and pointers to establish a relation with other entities.

With this set of definitions, it's possible to define the primitive routines:

1. Feed the entities with initial values and specific information, with the automatic generation of transport elements defined by headway. Special care with conflict manager deal to transport element allocation in already occupied place. The next available position is set taking care with constrains conditions (routes, segments and sectors sequences).
2. Output routines: a simulation of 2 hours updating each 15 seconds would generate lots of listings or a comprehensive excel spreadsheet.
3. System update: it's the software kernel, which is called each execution cycle to: load the nodes and for each transport element update its net position and load.

Using this approach, each primitive routine perform its function updating the information it obtain with mathematical equations or direct input, independently of the others.

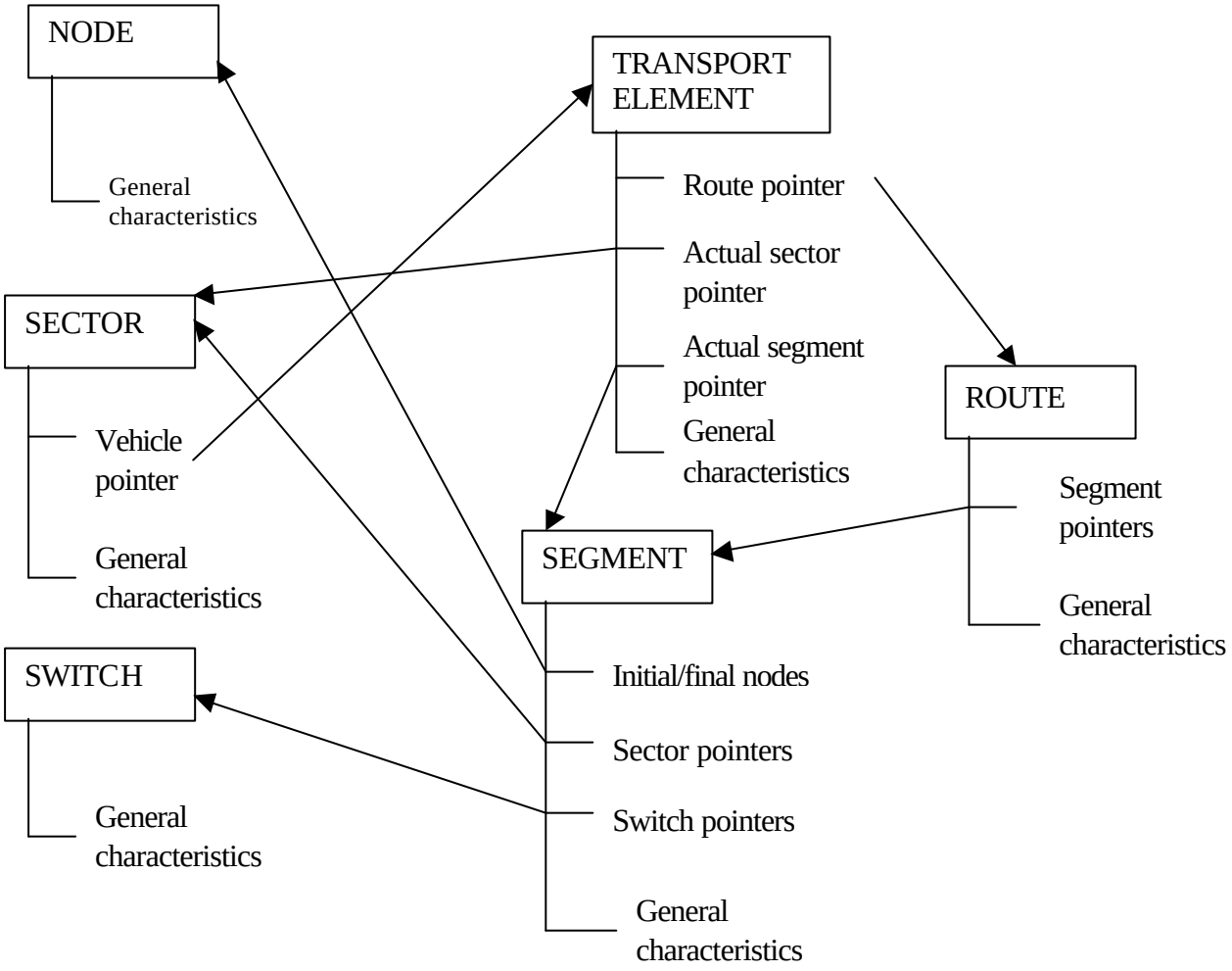

Figure 1. Distributed parameter data modelling.

Genetic algorithms applied to transport optimisation.

Genetic algorithms (GA) mimic the evolution and improvement of life through reproduction, where each individual contributes with its own genetic information to the building of new ones adapted to the environment with higher chances of surviving. This is the basis of genetic algorithms and programming (Holland (1975), Goldberg (1989) and Koza(1992). Specialized Markov Chains underline the theoretical bases of this algorithm change of states and searching procedures.

Each 'individual' of a generation represents a feasible solution to the problem, coding distinct algorithms/parameters to be evaluated by a fitness function.

GA operators are mutation (the change of a randomly chosen bit of the chromosome) and crossover (the exchange of randomly chosen slices of chromosome).

The best individuals are continuously being selected, and crossover and mutation take place. Following a number of generations (fig.2), the population converges to the solution that performs better.

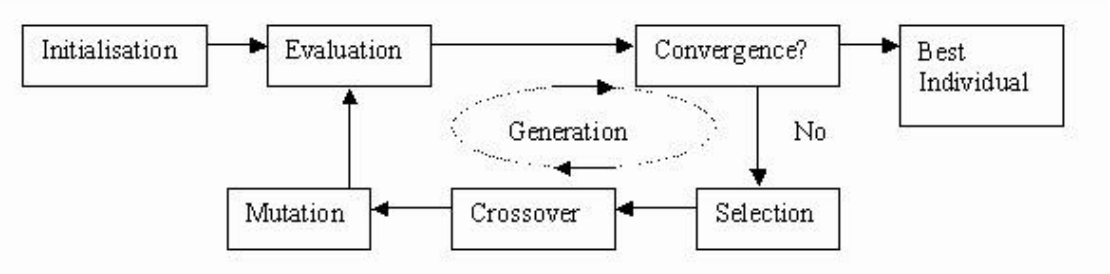

Fig. 2 Genetic algorithm: the sequence of operators and evaluation of each individual.

Genetic algorithm concepts and implementation are structured in the following diagram.

Population initialization

Step 1 : Parent selection.

	First population	Objective value z=f(x,y)
	1100110110101000	3.481746
	0101010110110101	3.668023
==>	1000010100110110	6.261380
==>	1101011111001100	12.864222

Step 2: Crossover

11010	11111001100	------- \		11010	10100110110
10000	10100110110	------- /		10000	11111001100

Step 3: Mutation

1101010100110110	1000011111001100
1111010100100110	1000011111001100
z = 8.044	z = 6.092

Step 4: Reinsertion

	second population	Objective value z=f(x,y)
==>	1111010100100110	8.044
==>	1000011111001100	6.092
	1000010100110110	6.261380
	1101011111001100	12.864222

Step 5: If the result doesn't converge, go to step 1.

Software structure.

The parameters set available to optimisation code a chromosome with should be evaluated with the disperse system simulator. Results accuracy depends of information available to the simulator, and would be executed off-line as presented in figure 3.

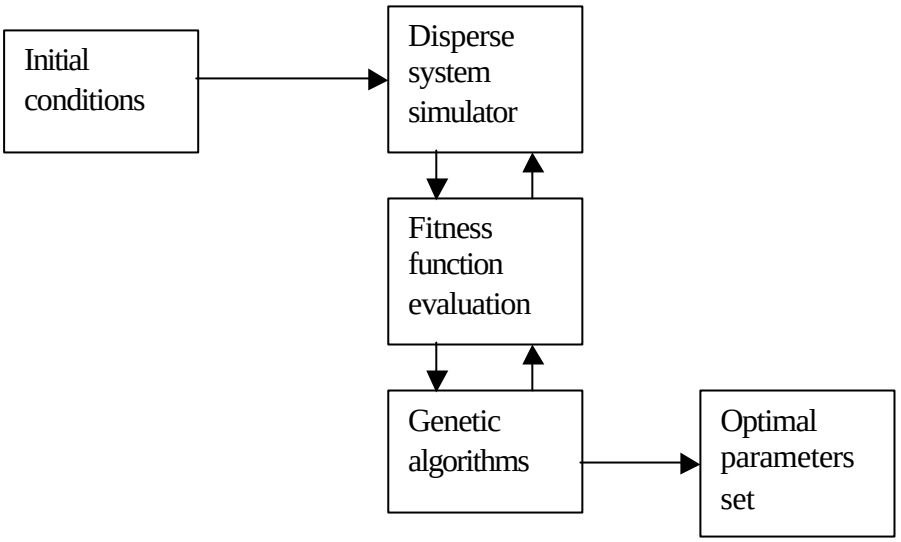

Figure 3. Software optimisation structure.

The simulator first loads the net with vehicles using the headway, wait time in each node, and performance level (a percentage of the maximum velocity), and then the simulation take place with a charged net.

The problem was simulated using three steps. The first optimise the headway for a give net without any exogenous event (not unusual happens). The second simulate the exogenous events with the condition simulated in first step to evaluate the effect of the problem into the evolution of the system.

The last step optimises the system performance level to reduce the effect of the problem. The same time interval is simulated in all steps, and at 500 time unit some sector is blocked, and an alternate route is set for all lines, because this is a distributed system and the some time is necessary to adapt all the system.

Usual genetic algorithms parameters for this step are population of 50 individuals, 30 generations, crossover probability 50% and mutation probability 5%.

The main aim of simulation and optimisation software is predict the behaviour of the system in future, and study the effect of rescheduling into the running system.

Results and outputs.

Two basics route would be found in transport problems: the linear and the circular. An example of each one is first presented to study the optimisation output, and later a mix of both to test its interferences.

The linear route has 5 stations, 6 segments with 31 sectors, and 2 junctions at each extreme with 2 routes. The circular has 6 stations with 1 junction, 16 segments with 43 sectors with 2 routes. The mix configuration is a circular with 2 linear sharing part of circular route, 14 stations, 5 junctions, 91 sectors and 39 segments with 6 routes.

We select the headway as free variable to optimise the system without event, using genetic algorithms. The algorithm found the solution in 2 generations, without any difficulty.

Then, we introduce two sectors interdict after 500 seconds of simulation, and evaluate their effects in passenger number and delay into 5000 seconds of simulation for linear and circular routes and one interdiction to mix route in a share sector of circular and second linear.

The next step was found the optimal value of transport element velocity percentage to optimise the system using genetic algorithm again.
The fitness function had the format:

$$Fitness = \frac{\sum Passenger\_on\_board}{\sum Delays * 0.3 * Max\_Passengers\_on\_board - \sum Passenger\_on\_Plataform}$$

Table 1 presents the total number of transported people and the delay it suffer due to occupied sector.

Table 1. Transport system optimisation with external events.

		Linear		Circular		Mix		
						Circ	Lin	Lin
Optimal headway with 75% velocity	Headway(sec)	58		60		93	95	100
	Total delay(sec)	3,795		3,360		1,575	765	705
	Passenger number (#)	73,791		102,278		84,724	62,116	59,893
Event effect	Number vehicles (#)	12		16		10	8	6
	Total delay (sec)	15,525	409%	6,780	201%	6,585 418%	615 80%	5,985 848%
	Passenger number (#)	68,109	92%	102,256	99%	84,502 99%	62,146 100%	52499 87%
Optimal velocity	%Velocity	47		50		37	39	40
	Total delay (sec)	2,430	64%	1,350	40%	2,010 127%	375 49%	2,010 285%
	Passenger number (#)	63,674	86%	57,608	56%	76,243 89%	45,026 72%	47,919 80%

We can see from results table that an external unpredicted event generates a 2, 4 or 8 times degradation the normal operational value, and with genetic algorithms application its possible to revert its effect with low performance lost (see optimal velocity results and compare with event effect). The percentage together the value is related with optimal headway value without external events.

Future works and conclusion.

Through the association of distributed system simulator and Genetic Algorithms is possible to study the effect of different operational strategies, and the real time rescheduling. It's explores a very particular characteristic of GA, with conserve in its population individuals of different features that in determined moment would be the optimal solution to the problem.

The future work is obtaining the optimal route set, due the application of Genetic Control heuristics (Werner (1999)), with the division of the challenge into the steps:

- Genetic programming to obtain the possible routes to transport some amount of passenger in a given topology.
- Genetic programming to obtain the best action to solve external event.

- Genetic algorithms to adapt operational parameters.

We are looking for transport companies with optimisation concern to apply a beta test version of software to real world problem.

References.

CHAMBERS,L.; "*The practical handbook of Genetic Algorithms*" Chapman & Hall/CRC,2000.

GOLDBERG,D.E. "*Genetic Algorithms in Search, Optimisation, and Machine Learning.*" Reading, Mass.: Addison-Whesley, 1989.

HOLLAND,J.H. "*Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control and artificial intelligence.*" Cambridge: Cambridge press 1992 reedição 1975.

KOZA,J.R. "*Genetic programming: On the programming of computers by means of natural selection.*" Cambridge,Mass.: MIT Press, 1992.

SwedeTrack; Computer Simulation of the Network;
http://www.swedetrack.com/esimul.htm

Werner,J.C.; "*Active noise control in ducts using genetic algorithm*" PhD. Thesis - São Paulo University - São Paulo - Brazil - 1999.